Netra ft™ 1800
User's Guide Update

Solaris 2.6 for Update 01

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. 806-3826-10
December 1999, Revision A

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

Copyright 1999 Sun Microsystems, Inc., 901 San Antonio Road • Palo Alto, CA 94303 USA. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd. For Netscape Communicator™, the following notice applies: Copyright 1995 Netscape Communications Corporation. All rights reserved.

Sun, Sun Microsystems, the Sun logo, AnswerBook, Java, the Java Coffee Cup logo, Netra, Netra ft, OpenBoot, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 1999 Sun Microsystems, Inc., 901 San Antonio Road • Palo Alto, CA 94303 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd. La notice suivante est applicable à Netscape Communicator™ : Copyright 1995 Netscape Communications Corporation. All rights reserved.

Sun, Sun Microsystems, le logo Sun, AnswerBook, Java, le logo Jave Coffee Cup, Netra, Netra ft, OpenBoot, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. [

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Please Recycle

Adobe PostScript

Contents

Figures

Tables

Code Samples

Preface

This book explains in detail, with examples and sample procedures, some of the unique features of the Netra ft™ 1800 system. It describes additional functionality that has been developed since the Netra ft 1800 system was first released, and it supplements information that is available in the original system manuals.

Before You Read This Book

In order to fully use the information in this book, you must have thorough knowledge of the topics discussed in these books:

- *Netra ft 1800 User's Guide*
- *Netra ft 1800 Hardware Reference Manual*

How This Book Is Organized

Chapter 1 describes the functions and use of fault-tolerant networks.

Chapter 2 describes how serial connection is provided in the Netra ft 1800 system.

Chapter 3 describes split mode and how to use it.

Chapter 4 contains a detailed explanation of OSdog and how to use it.

Chapter 5 provides a procedure for hot swapping the motherboards.

Appendix A contains various man pages which can be referred to for detailed descriptions of commands and functions described in this book.

Appendix B lists a script for remotely reporting module failures.

Appendix C describes the automatic firmware update utility.

Appendix D provides information about using the alarms utility.

Appendix E provides a split example for use when patch D is installed.

Using UNIX Commands

This document may not contain information on basic UNIX® commands and procedures such as shutting down the system, booting the system, and configuring devices.

See one or more of the following for this information:

- AnswerBook2™ online documentation for the Solaris™ operating environment
- Other software documentation that you received with your system

Typographic Conventions

TABLE P-1 Typographic Conventions

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output	`% su` `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide.* These are called *class* options. You *must* be superuser to do this.
	Command-line variable; replace with a real name or value	To delete a file, type `rm` *filename.*

Shell Prompts

TABLE P-2 Shell Prompts

Shell	Prompt
C shell	*machine_name*%
C shell superuser	*machine_name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

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 (806-3826-10) of your document in the subject line of your email.

CHAPTER 1

Fault-Tolerant Networks

This chapter describes how a Netra ft 1800 system handles fault-tolerant networks and provides examples of how to configure and unconfigure them. The chapter contains the following sections:

The Netra ft 1800 system supports a number of fault-tolerant Ethernet interfaces. Each interface is a single logical connection, even though it is implemented by two physical connections, a configuration known as *dual-attach*. Two Ethernet controllers are located on each motherboard to drive the physical connections, and additional controllers can be added with PCI modules, either in singles or quadruples.

Overview

By default, a system has four motherboard Ethernet devices that are accessed by two pairs of ports on two CAF modules (one per side), which connect through to the motherboards. The A-side CAF hosts devices `a-net0` and `a-net1`. The B-side CAF hosts devices `b-net0` and `b-net1`.

To achieve fault tolerance through redundancy, a Solaris streams multiplexing device driver, called `pnet`, is used to create logical network interfaces that are composed of two physical device instances. For fault tolerance, each device instance in a `pnet` must be located on a separate side of the system to avoid a single point of failure.

For example, `pnet0` might link together the `a-net0` and `b-net0` devices to form a single fault tolerant network interface. Similarly, `pnet1` might link together `a-net1` and `b-net1` to form another fault tolerant interface. To Ethernet clients, a `pnet` acts as a conventional DLPI 2-compliant driver and so no application software modifications are needed to take advantage of the fault tolerance.

The `pnet` interface can be considered as an Ethernet device that is available for use, and which multiplexes two Ethernet devices below it. FIGURE 1-1 shows one possible configuration, with an application using a TCP socket bound to a `pnet` interface consisting of two underlying devices.

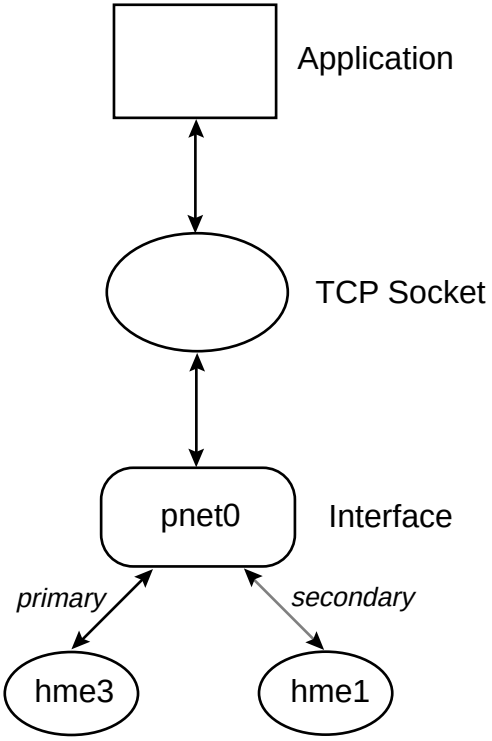

FIGURE 1-1 `pnet` Fault-Tolerant Interface

MAC Address Allocation

Each Netra ft 1800 system comes preloaded with 68 MAC addresses. The MAC addresses are divided into two contiguous blocks, referenced from a BASE address that is fixed for the system. The allocation of these addresses is shown in the following table. In split mode, the MAC address handling is different from that in combined (fault-tolerant) mode.

TABLE 1-1 MAC Address Allocation

MAC Address	OBP Device	Solaris Device
BASE+0	a-net0	A-CAF Net-0
BASE+1	a-net1	A_CAF Net-1
BASE+2	pnet0	(combined or on side A of split machine)
.		
BASE+33	pnet31	(combined or on side A of split machine)
BASE+34	b-net0	B-CAF Net-0
BASE+35	b-net1	B-CAF Net-1
BASE+36	pnet0	(on side B of split machine)
.		
BASE+67	pnet31	(on side B of split machine)

When a system is in OpenBoot™ PROM, it is free to use any of its four Ethernet devices (`[a|b]-net[0|1]`) although it can boot only from `a-net0` (see “Network Booting and Installation” on page 5 for details). When a system is installed and booted, however, it normally uses `pnet0` as its Ethernet interface, which , in turn, uses the BASE+2 MAC address and binds that to `Net-0` on CAF-A (which is `a-net0`) and Net-0 on CAF-B (which is `b-net0`). Similarly, `pnet1` binds BASE+3 to `Net-1` on CAF-A (`a-net1`) and `Net-1` on CAF-B (`b-net1`).

The behavior is similar for Ethernet cards; there is one MAC address for each physical Ethernet device, which is subsumed when it is configured as part of a `pnet` interface.

If you use `ifconfig -a` to list Ethernet interfaces on a system, only the loopback (`lo0`) and the `pnet` interfaces are displayed. For example, for a fully configured system with `pnet0` and `pnet1` operational, the `ifconfig -a` command displays the following:

```
# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232 inet 127.0.0.1 netmask ff000000
pnet0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500 inet 129.144.42.65
netmask ffffff00 broadcast 129.144.42.255 ether 8:0:20:a5:34:f2
pnet1: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500 inet 129.144.42.64
netmask ffffff00 broadcast 129.144.42.255 ether 8:0:20:a5:34:f3
```

There are no references to any `hme` devices.

Fault-Tolerant Operation

When a pair of physical network instances have been formed into a `pnet`, one of the pair is identified as the *primary* instance. The primary instance is responsible for transmitting and receiving the packets on the `pnet` interface. The other instance, termed the *secondary*, is used as a backup in case the primary fails. The primary instance takes the MAC address of the `pnet`, while the secondary instance retains its default address. Hence, in the `pnet0` and `pnet1` examples above, with A-side devices as the primary instances:

- CAF-A Net-0 has BASE+2 MAC address
- CAF-B Net-0 has BASE+34 MAC address (for latent fault checking)
- `pnet0` has BASE+2 address

If the B-side is the primary instance of `pnet0`:

- CAF-A Net-0 has BASE+0 MAC address (for latent fault checking)
- CAF-B Net-0 has BASE+2 MAC address
- `pnet0` has BASE+2 MAC address

The primary always takes the MAC address of the `pnet` interface. The secondary always uses the MAC address of the physical device, for latent fault checking.

If the device is a CAF network device, the MAC address is as shown in TABLE 1-1 ”MAC Address Allocation“ on page 3.

If the secondary is on a PCI plug-in card, the MAC address is programmed into the module EEPROM on the PCI card.

Failure Handling

If a failure of the primary instance occurs—that is, either a device or cable failure—the `pnet` switches over to the secondary instance and reprograms the MAC address of the secondary to be the `pnet` address and return the MAC address of the failed primary to its default.

For example, given `pnet0` A-side primary, CAF-A Net-0 BASE+2 MAC address, CAF-B Net-0 BASE+34 MAC address, if CAF-A Net-0 fails, the `pnet` remaps the addresses to be CAF-A Net-0 BASE MAC address (failed) and CAF-B Net-0 BASE+2 MAC address.

Network Booting and Installation

When booting after an install, the CMS attempts to configure `ft_network 0` and `ft_network 1` automatically. `ft_network 0` will consist of `A Net 0` and `B Net 0`, and have its hostname set to that of the system. `ft_network 1` will consist of `A Net 1` and `B Net 1`, but will not have a hostname.

Network booting and installation are supported, but with the following restriction. You can network boot only from device `a-net0` because only the system BASE+0 MAC address can be used. Hence, the devices `a-net1`, `b-net0` and `b-net1` cannot be used to network boot a system in combined mode.

Latent Fault Checking

If an Ethernet module or controller fails, it is normally detected by the driver. If a cable is removed, this will be detected by the driver and an error is signaled. The driver can also detect when the cable is replaced.

One method for detecting a failure on the network—for example, a broken hub—is for the two instances in a `pnet` to exchange test packets. The packets are MAC packets. If you snoop a system from itself (see CODE EXAMPLE 1-1 ”Using the snoop Command“ on page 8) and examine the traffic related to the `pnet,` you will see that the primary and secondary instances send a packet to each other every two seconds (Ethertype = `0xbabe`).

FIGURE 1-2 shows the Ethernet frame associated with the LFC MAC packet, which is 74 bytes in length.

- `D addr` is set to the destination MAC address for the packet.
- `S addr` is set to the MAC address of the sender
- The type is set to `0xbabe.`
- The data field is zeroed.

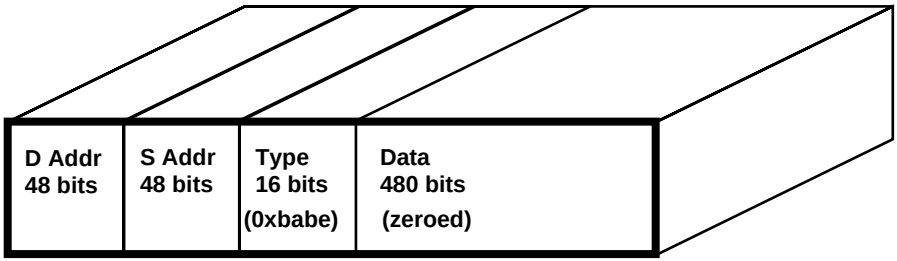

FIGURE 1-2 Ethernet Frame of Test MAC Packet

FIGURE 1-3 displays the basic behavior of latent fault checking. A test packet is sent and a flag is set every two seconds. If the test packet is received, the flag is unset. If the test packet is not received, the flag is left in the set position. However, the first part of the process is to check the status of the flag. A flag that is still in the set position indicates that the previous packet, sent two seconds ago, did not arrive. This suggests that a problem exists with either of the two Ethernet hardware media . It is not always possible to isolate the problem to a specific card, as the fail may be due to another part of the network (for example, an incorrectly configured switch).

Once a problem is recognized, an escalation process begins. The first part of the escalation process is to increase the frequency of the test packet transmission to 2 per second. If packets are still not getting through, the primary instance that has the problem transmits a multicast packet 224.0.0.1 every 30 seconds. This transmission is attempted 50 times, and the primaryship is alternated if they are unsuccessful. This behavior ceases if a reply is received and the standard test packet process is resumed at the normal rate of 2 per second.

If a reply to the multicast packet is not received after 50 attempts, the following message is displayed in the console window:

```
% network connectivity problem
```

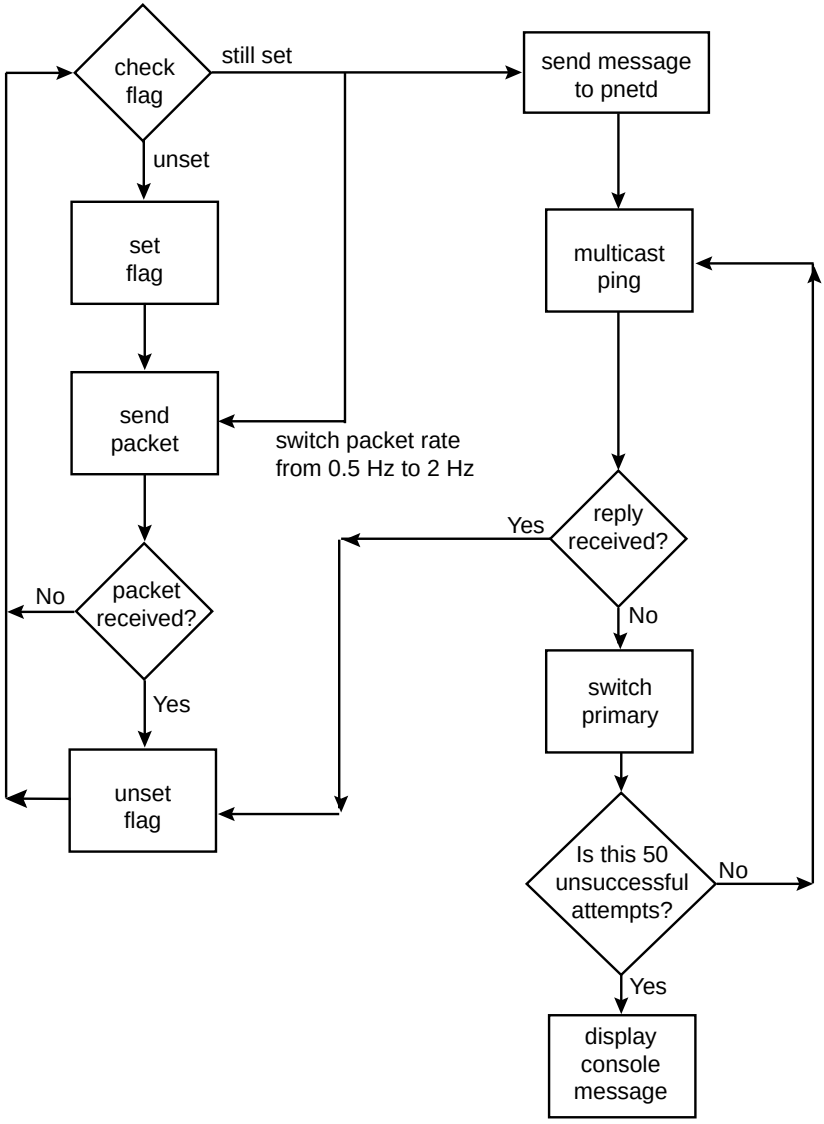

FIGURE 1-3 Latent Fault Checking Flow Diagram

The test packet model assumes that the two instances of a `pnet` are connected via broadcast media. If they are not, the test packets cannot be routed and the `pnet` assumes a failure of the primary instance and takes corrective action. This can mean that the `pnet` enters a cycle of continually switching primaryship in a vain attempt to obtain a working primary.

An example of this can occur when an ft_network has been configured as fault tolerant, but only one of the transceiver interface cables has been connected to the network, or there is a bad connection on one side of the cables.

CODE EXAMPLE 1-1 Using the snoop Command

```
#root@tortoise # snoop -d pnet0
Using device /dev/pnet (promiscuous mode)
           ? -> *       ETHER TYPE=BABE (Unknown), size = 74 bytes
           ? -> *       ETHER TYPE=BABE (Unknown), size = 74 bytes
           ? -> *       ETHER TYPE=BABE (Unknown), size = 74 bytes
           ? -> *       ETHER TYPE=BABE (Unknown), size = 74 bytes
           ? -> *       ETHER TYPE=BABE (Unknown), size = 74 bytes
           ? -> *       ETHER TYPE=BABE (Unknown), size = 74 bytes
```

Consider the following when cabling up a network. Essentially, the two cables are a single network connection, and should be treated as such. The fault-tolerant philosophy can be continued into the network itself; for example, the pnet cables could go to two separate hubs, as long as the hubs are daisy chained together, thus providing a broadcast path for the latent fault-checking test packets.

Caution – Latent fault checking is not supported on a network switch unless the switch ports are configured to see each other's traffic.

CMS Configuration

The pnet devices are configured and managed using the CMS. The CMS object ft_network 0 corresponds to pnet0, and so on.

Using the CMS, you can configure the physical modules that will be used by an ft_network, the ports on those modules, and the host name for the interface.

- An ft_network state can be not_present, online_down or online_up.
- The controllerA_FRU and controllerB_FRU attributes indicate which CAF or PCI module is used.
- The controllerA_Funct and controllerB_Funct attributes indicate the specific port or PCI module device (choice of four for QFE cards, for example).
- The usable_controllers attributes shows which devices are online and available to be used as a primary connection.

- The `controller_in_use` attribute shows the primary controller.
- The `preferred_controller` attribute can be used to force and rotate the primary device in the `pnet`.

One way of disabling a `pnet` is to set the `controllerA_Funct` and `controllerB_Funct` attributes to null.

Note – To ensure a stable platform, disable active routing and ip forwarding.

Note – All generic Solaris utilities, such as `ping`, `netstat`, `ifconfig`, work as normal except that you will be viewing the `pnet` driver and not the `hme` device directly. `netstat` reports a combination of the pair of `hme` statistics.

Files Used by Fault-Tolerant Networks

TABLE 1-2 Files Used by Fault-Tolerant Networks

File	Function
`/platform/SUNW,Ultra-4FT/lib/pnetd`	Daemon (this one is started by default), static linkage
`/platform/SUNW,Ultra-4FT/lib/pnetctl`	Device driver interface utility, static linkage
`/usr/platform/SUNW,Ultra-4FT/SUNWcms/lib/pnetd`	Daemon, dynamic linkage
`/usr/platform/SUNW,Ultra-4FT/SUNWcms/lib/pnetctl`	Device driver interface utility, dynamic linkage
`/etc/init.d/init.pnet`	Start and stop `pnetd`
`/etc/hostname.pnet[0-33]`	Host name for interface, autogenerated by CMS
`/etc/cms.hostname.pnet[0-31]`	Host name for unconfigured interface, autogenerated by CMS
`/dev/pnet`	Linked to `/devices/pseudo/clone@0:pnet` special file
`/etc/link.[a-b]-net[0-1]`	Defines ft_network configuration of motherboard Ethernet devices (used when booting)

Configuring and Unconfiguring ft_networks

A fault-tolerant ft_network consists of two physical network interfaces that are linked together into a virtual network interface. When the ft_network is configured with only one physical network interface, it is said to be *nonfault-tolerant*. When an ft_network is configured (fault-tolerant or nonfault-tolerant) it is listed by the ifconfig -a command as a pnet device.

This section contains the following examples:

Note – When using cmsconfig, the values displayed are updated only when you press Return to request a refresh. When making changes to CMS attributes, you may need to refresh the screen a couple of times to view all the changes taking place. When an ft_network state shows busy (as shown in menu item 0 in the examples), a reconfiguration of the ft_network is in progress. To view the latest values, you should refresh the screen by pressing Return until a new and stable state is reached or until no more attributes are being updated.

▼ To Configure an ft_network to Be Nonfault-Tolerant

In this example you configure the previously unused ft_network 1 to be nonfault-tolerant, comprising the single network interface A-CAF/Net_1.

1. **Type:**

```
# cmsconfig
```

2. **At the prompt, type:**

```
i ft_network 1
```

3. **Page forward by typing** `p` **and Return until** `ft_network 1` **is displayed in the menu list.**

4. **Enter the menu number for** `ft_network 1` **and press Return.**

   The `cmsconfig` screen is displayed:

```
Select: ft_network 1
Item    Name                 Value                                  Page 1 of 1
-------------------------------------------------------------------------------
0       state                not_present
1       description          network multiplexor
2       user_label
3       hostname
4       preferred_controller  none
5       controllerA_FRU      NULL
6       controllerA_Funct    null
7       controllerB_FRU      NULL
8       controllerB_Funct    null
9       info
10      busylock             no
11      devpath
12      link
13      transceiver
14      usable_controllers   null
15      controller_in_use    none

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

5. **Choose** `ControllerA_FRU` **(menu item 5) and set to** `A-CAF`**.**

6. **Choose** `ControllerA_Funct` **(menu item 6) set to** `Net_1`**.**

Note – You must set the `ControllerA_FRU` field followed by the `ControllerA_Funct` field.

---

7. **Wait two seconds to allow the reconfiguration to complete, then press Return.**

   The `link` status is updated; for example `100 Mb full-duplex link up`. The state changes to `online_down`.

8. **Choose** `hostname` **(menu item 3) and set to a valid host name that is defined in** `/etc/hosts`**.**

9. **Wait two seconds to allow the reconfiguration to complete, then press Return.**

   The `state` changes to `online_up`.

```
Select: ft_network 1
Item    Name                 Value                          Page 1 of 1
-------------------------------------------------------------------------------
0       state                online_up
1       description          network multiplexor
2       user_label
3       hostname             goonshow
4       preferred_controller none
5       controllerA_FRU      A-CAF
6       controllerA_Funct    Net_1
7       controllerB_FRU      NULL
8       controllerB_Funct    null
9       info
10      busylock             no
11      devpath              pnet0
12      link                 100 Mbps half-duplex link up
13      transceiver          internal transceiver selected
14      usable_controllers   A (online)
15      controller_in_use    A

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

▼ To Unconfigure a Nonfault-Tolerant `ft_network`

In this example, you completely unconfigure the nonfault-tolerant `ft_network 1` that comprises the single network interface A-CAF/Net_1. This enables this network interface to be re-used in a different configuration.

1. **Run** `cmsconfig`**.**

   If `ft_network 1` is not displayed on the first page, page forward by typing **p** followed by Return.

2. **Choose the menu item for** `ft_network 1`**.**

```
Select: ft_network 1
Item    Name                 Value                          Page 1 of 1
-------------------------------------------------------------------------------
0       state                online_up
1       description          network multiplexor
2       user_label
3       hostname             goonshow
4       preferred_controller none
5       controllerA_FRU      A-CAF
6       controllerA_Funct    Net_1
7       controllerB_FRU      NULL
8       controllerB_Funct    null
9       info
10      busylock             no
11      devpath              pnet0
12      link                 100 Mbps half-duplex link up
13      transceiver          internal transceiver selected
14      usable_controllers   A (online)
15      controller_in_use    A

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

3. **Choose the** `hostname` **(menu item 3) and set it to blank.**
4. **Choose the** `ControllerA_FRU` **field (menu item 5) set it to** `null`**.**
5. **Choose the** `ControllerA_funct` **field (menu item 6) and set it to** `null`**.**
6. **Wait two seconds, then press Return.**

   The `ft_network` state changes to `not_present`.

The final result as viewed in `cmsconfig` is:

```
Select: ft_network 1
Item    Name                 Value                          Page 1 of 1
-------------------------------------------------------------------------
0       state                not_present
1       description          network multiplexor
2       user_label
3       hostname
4       preferred_controller  none
5       controllerA_FRU       NULL
6       controllerA_Funct     null
7       controllerB_FRU       NULL
8       controllerB_Funct     null
9       info
10      busylock              no
11      devpath
12      link
13      transceiver
14      usable_controllers    null
15      controller_in_use     none

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

▼ To Configure an `ft_network` to Be Fault-Tolerant

In this example you, configure the previously unused `ft_network 1` to be fault-tolerant, comprising the network interfaces A-CAF/Net_1 and B-CAF/Net_1.

1. **Type**:

```
# cmsconfig
```

2. **At the prompt, type**:

```
i ft_network 1
```

3. **Page forward by typing** `p` **followed by Return until** `ft_network 1` **is displayed in the menu list.**

4. **Enter the menu number for** `ft_network 1` **and press Return.**

   The `cmsconfig` screen is displayed:

```
Select: ft_network 1
Item    Name                 Value                          Page 1 of 1
-------------------------------------------------------------------------
0       state                not_present
1       description          network multiplexor
2       user_label
3       hostname
4       preferred_controller  none
5       controllerA_FRU       NULL
6       controllerA_Funct     null
7       controllerB_FRU       NULL
8       controllerB_Funct     null
9       info
10      busylock              no
11      devpath
12      link
13      transceiver
14      usable_controllers    null
15      controller_in_use     none

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

5. **Choose** `ControllerA_FRU` **(menu item 5) and set to** `A-CAF`**.**

6. **Choose** `ControllerA_Funct` **(menu item 6) and set to** `Net_1`**.**

Note – You must set the `ControllerA_FRU` field followed by the `ControllerA_Funct` field.

---

7. **Wait two seconds, then press Return.**

   The `link` status is updated; for example `100 Mb full-duplex link up`. The `state` changes to `online_down`.

8. **Choose** `hostname` **(menu item 3) and set to a valid host name that is defined in** `/etc/hosts`**.**

9. **Wait two seconds, then press Return.**

   The `state` changes to `online_up`.

10. **Choose** `ControllerB_FRU` **(menu item 7) and set to** `B-CAF`**.**

11. **Choose** `ControllerB_Funct` **(menu item 8) set to** `Net_1`**.**

12. **Wait two seconds, then press Return.**

    The `usable_controllers` field now shows `A (online) & B (online)`.

    The `ft_network` as viewed in `cmsconfig` appears as:

```
Select: ft_network 1
Item    Name                 Value                          Page 1 of 1
--------------------------------------------------------------------------------
0       state                online_up
1       description          network multiplexor
2       user_label
3       hostname             goonshow
4       preferred_controller none
5       controllerA_FRU      A-CAF
6       controllerA_Funct    Net_1
7       controllerB_FRU      B-CAF
8       controllerB_Funct    Net_1
9       info
10      busylock             no
11      devpath              pnet1
12      link                 100 Mbps half-duplex link up
13      transceiver          internal transceiver selected
14      usable_controllers   A (online)& B (online)
15      controller_in_use    A

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

▼ To Unconfigure a Fault-Tolerant `ft_network`

In this example, you unconfigure the fault-tolerant `ft_network 1` that comprises A-CAF/Net_0 and B-CAF/Net_0. This enables these network interfaces to be re-used in a different configuration.

1. **Type:**

```
# cmsconfig
```

   If `ft_network 1` is not displayed on the first page, page forward by typing **p** followed by Return.

2. **Choose the menu item for** `ft_network 1`**.**

The `cmsconfig` screen is displayed:

```
Select: ft_network 1
Item    Name                 Value                          Page 1 of 1
-------------------------------------------------------------------------------
0       state                online_up
1       description          network multiplexor
2       user_label
3       hostname             goonshow
4       preferred_controller none
5       controllerA_FRU      A-CAF
6       controllerA_Funct    Net_1
7       controllerB_FRU      B-CAF
8       controllerB_Funct    Net_1
9       info
10      busylock             no
11      devpath              pnet1
12      link                 100 Mbps half-duplex link up
13      transceiver          internal transceiver selected
14      usable_controllers   A (online)& B (online)
15      controller_in_use    A

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

3. **Choose the `ControllerB_FRU` field (menu item 7) and set it to `null`.**
4. **Choose the `ControllerB_funct` field (menu item 8) and set it to `null`.**
5. **Wait two seconds, then press Return.**

   The `usable_controllers` attribute changes to `A (online)`.
6. **Choose the `hostname` (menu item 3) and set it to blank.**
7. **Wait two seconds, then press Return key.**

   The `ft_network` state changes to `online_down`.
8. **Choose the `ControllerA_FRU` field (menu item 5) and set it to `null`.**
9. **Choose the `ControllerA_funct` field (menu item 6) and set it to `null`.**
10. **Wait two seconds, then press Return.**

    The `ft_network state` changes to `not_present`.

The `cmsconfig` screen is displayed:

```
Select: ft_network 1
Item    Name                 Value                          Page 1 of 1
-------------------------------------------------------------------------------
0       state                not_present
1       description          network multiplexor
2       user_label
3       hostname
4       preferred_controller  none
5       controllerA_FRU      NULL
6       controllerA_Funct    null
7       controllerB_FRU      NULL
8       controllerB_Funct    null
9       info
10      busylock             no
11      devpath
12      link
13      transceiver
14      usable_controllers   null
15      controller_in_use    none

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

▼ To Configure a Fault-Tolerant `ft_network` to Be Nonfault-Tolerant

In this example, you configure the fault-tolerant `ft_network 1` comprising A-CAF/Net_1 and B-CAF/Net_1 to be a nonfault-tolerant `ft_network` comprising A-CAF/Net_1.

1. **Type:**

```
# cmsconfig
```

If `ft_network 1` is not displayed on the first page, page forward by typing **p** followed by **Return**.

2. **Choose the menu item for** `ft_network 1`**.**

The `cmsconfig` screen is displayed:

```
Select: ft_network 1
Item    Name                 Value                          Page 1 of 1
-------------------------------------------------------------------------------
0       state                online_up
1       description          network multiplexor
2       user_label
3       hostname             goonshow
4       preferred_controller none
5       controllerA_FRU      A-CAF
6       controllerA_Funct    Net_1
7       controllerB_FRU      B-CAF
8       controllerB_Funct    Net_1
9       info
10      busylock             no
11      devpath              pnet1
12      link                 100 Mbps half-duplex link up
13      transceiver          internal transceiver selected
14      usable_controllers   A (online)& B (online)
15      controller_in_use    A

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

3. **Choose the `ControllerB_FRU` (menu item 7) field and set it to `null`.**
4. **Choose the `ControllerB_funct` (menu item 8) field and set it to `null`.**

5. **Wait two seconds, then press Return.**

   The `usable_controllers` attribute changes to `A (online)`.

```
Select: ft_network 1
Item    Name                  Value                          Page 1 of 1
-------------------------------------------------------------------------------
0       state                 online_up
1       description           network multiplexor
2       user_label
3       hostname              goonshow
4       preferred_controller  none
5       controllerA_FRU       A-CAF
6       controllerA_Funct     Net_1
7       controllerB_FRU       null
8       controllerB_Funct     null
9       info
10      busylock              no
11      devpath               pnet1
12      link                  100 Mbps half-duplex link up
13      transceiver           internal transceiver selected
14      usable_controllers    A (online)
15      controller_in_use     A

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

▼ To Set or Change the Preferred Controller

The `cmsconfig ft_network` object enables you to specify the preferred network interface for a fault-tolerant `ft_network`. On system startup, the `ft_network` is configured so that the `ft_network` uses the preferred controller (network interface) as primary.

For example, you could configure half of the `ft_networks` in a system to prefer the controllers on the physical side A of the machine and the other half to prefer the controllers located on physical side B. If one of the sides fails (for example, total power loss to one side), only half of the `ft_networks` would need to swap to the secondary controller, thus minimizing disruption to network traffic.

You can set the preferred controller while the `ft_network` is in any state. If the `usable_controllers` field shows two controllers that are online, the change takes place immediately; otherwise, the change takes place as soon as the controllers reach this state.

1. **Type**:

```
# cmsconfig
```

   If `ft_network 1` is not displayed on the first page, page forward by typing **p** followed by Return.

2. **Choose the menu item for** `ft_network 1`**.**

   The `cmsconfig` screen is displayed:

```
Select: ft_network 1
Item    Name                 Value                          Page 1 of 1
-------------------------------------------------------------------------------
0       state                online_up
1       description          network multiplexor
2       user_label
3       hostname             goonshow
4       preferred_controller none
5       controllerA_FRU      A-CAF
6       controllerA_Funct    Net_1
7       controllerB_FRU      B-CAF
8       controllerB_Funct    Net_1
9       info
10      busylock             no
11      devpath              pnet1
12      link                 100 Mbps half-duplex link up
13      transceiver          internal transceiver selected
14      usable_controllers   A (online)& B (online)
15      controller_in_use    A

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

3. **Choose menu item 4 and enter the menu item for the** `preferred_controller` **(**A**,** B**, or** none**).**

   In the example below, controller A has been specified.

```
Select: ft_network 1
Item    Name                  Value                          Page 1 of 1
--------------------------------------------------------------------------------
0       state                 online_up
1       description           network multiplexor
2       user_label
3       hostname              goonshow
4       preferred_controller  A
5       controllerA_FRU       A-CAF
6       controllerA_Funct     Net_1
7       controllerB_FRU       B-CAF
8       controllerB_Funct     Net_1
9       info
10      busylock              no
11      devpath               pnet1
12      link                  100 Mbps half-duplex link up
13      transceiver           internal transceiver selected
14      usable_controllers    A (online)& B (online)
15      controller_in_use     A

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

If you modify the `preferred_controller` field to be the controller other than the one that is currently in use (indicated by the `controller_in_use` field), the CMS arranges for the physical `controller_in_use` to switch over to the one specified.

4. **After specifying the new** `preferred_controller`**, wait two seconds, then press Return.**

   The display of the `ft_network` object is updated.

```
Select: ft_network 1
Item    Name                 Value                          Page 1 of 1
-------------------------------------------------------------------------------
0       state                online_up
1       description          network multiplexor
2       user_label
3       hostname             goonshow
4       preferred_controller B
5       controllerA_FRU      A-CAF
6       controllerA_Funct    Net_1
7       controllerB_FRU      B-CAF
8       controllerB_Funct    Net_1
9       info
10      busylock             no
11      devpath              pnet1
12      link                 100 Mbps half-duplex link up
13      transceiver          internal transceiver selected
14      usable_controllers   A (online)& B (online)
15      controller_in_use    B

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

Troubleshooting

Error Messages

This section describes the error messages you may encounter and suggests their possible cause.

```
Unable to set address
```

The chosen host name cannot be resolved to an IP address. A likely cause is that there is an entry missing from `/etc/hosts`.

```
Unable to bring interface up
```

The host name clashes with the host name of another network interface on the machine.

```
module port registered to service
```

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

```
Must have at least one controller
```

An attempt has been made to unconfigure the controller that is in use by a nonredundant `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 host name must be set before the state can change to `online_up`. Entering the hostname into the `ft_network` object causes the CMS to trigger an attempt to bring the network up.

Confirming the pnet Connection

You can determine the connectivity and round trip time of the `pnet` using the `ping -s` command from a client. If you then fail the primary instance using `u4ftctl fail`, the `pnet` switches the primaryship and connectivity is restored, with the switch taking about one second (measured).

If, instead, you disconnect the primary Ethernet cable, you observe similar switchover behavior and times.

Note – These experiments are not meant to give accurate failover times and should not be used as a guide for application designers. More accurate experiments reveal that the device failure is detected, and the switchover is effected within tens of milliseconds.

CHAPTER 2

ft_serial

This chapter describes the functionality of `ft_serial` in providing serial connection to the Netra t 1800 system.

The chapter has the following sections:

Accessing the Console

The Netra ft 1800 console can be accessed from multiple physical ports. In the default configuration, the console port on both CAFs provide access to the console. The console port on side A CAF is connected to a separate serial communications controller located on the side A motherboard. Similarly, the console port on the side B CAF is connected to a serial controller on the B motherboard.

This configuration guarantees console availability:

- During motherboard hot swap
- During CAF replacement
- After the failure of one of the serial controllers

During split-mode operation, each serial controller is allocated to its natural side. The consequence of this allocation is that fault-tolerant serial connections via the CAFs are not available when the system is split.

The serial controllers used on the Netra ft 1800 system provide two independent serial channels, with the input and output lines from the first channel connected to the CAF console port, and the lines from the second channel connected to the modem port on the same CAF.

To ensure the console configuration is fault tolerant, only ports on separate CAFs should be combined.

The methods by which different ports can be configured to provide console access are:

- Through the Configuration Management System (CMS)—the preferred method
- By using the `ttymuxadm` (1M) utility

File System Access to the Console

Fault-tolerant serial connections are provided by a separate STREAMS multiplexor driver called `ttymux` (alias name is `u4ftser`). To support booting, OpenBoot PROM creates a device node corresponding to the multiplexor with the path `/u4ftser@0,0`. The minor devices created by the serial multiplexor appear in the Solaris devices directory as:

- `/devices/u4ftser@0,0:[con|ctl]`
- `/devices/u4ftser@0,0:sm[a-f],cu`
- `/devices/u4ftser@0,0:sm[a-f]`

The corresponding device links to these files are:

- `/dev/ttymux:[con|ctl]`
- `/dev/term/[0-5]`
- `/dev/cua/[0-5]`

Each of these multiplexor devices can be associated with one or more real serial devices. If a real serial device is associated with a multiplexor device, real input and output can be performed by opening the multiplexor device. When two or more real serial devices on separate chips are associated with a multiplexor device, the multiplexor provides fault-tolerant service—any output is sent to all associated devices, and any input is multiplexed onto a single input stream and made available for reading.

Real serial devices are associated in a two-stage process:

1. Link the device underneath the ttymux driver
2. Associate the linked device with a multiplexor device

Both steps can be performed together using the `ttymuxadm` utility. Otherwise, a program must be written that issues a STREAMS link ioctl (`I_PLINK`), and then issues a `ttymux` driver-specific ioctl (`TTYMUX_ASSOC`).

A linked lower device cannot be accessed directly. Although you can open such a device, any reads or writes will fail.

Reading the Current Multiplexor Configuration

All linked devices and associations can be read with the `ttymuxadm` utility (using the `-i` option), or by issuing a driver-specific ioctl (`TTYMUX_LIST`). The `ttymuxadm -i` command provides configuration information in the following format:

```
A-CAF:1 20:1 /dev/term/b 31 (166:5 1095586644)
```

The first four fields provide information about a linked device:

- FRU name and port number
- Device number of the linked device
- Device path of the linked device
- STREAMS link identifier for the link

The information in brackets indicates whether the linked device is associated with one of the multiplexor devices (the two fields correspond to the device number of the associated multiplexor device, followed by a user-supplied identifier for the association, which is required when the association is destroyed).

If the lower device number is reported as `-1:-1`, the lower device is not linked. If the multiplexor device number is -1:-1, the linked lower device is not associated with any of the multiplexor devices.

Files

TABLE 2-1 Files Used by `ft_serial`

File	Function
`/kernel/drv/ttymux.conf`	Driver configuration file
`/etc/init.d/ttymuxrc`	Run at rcS to configure any persistent `ttymux` associations
`/sbin/ttymuxadm`	Serial multiplexor administration utility; uses driver-specific `ioctls(2)`
`/dev/term/[0-5]`	Links to the `/devices` directory for opening serial lines with dial-in semantics
`/dev/cua/[0-5]`	Links to the `/devices` directory for opening serial lines with dial-out semantics
`/dev/ttymux:con`	Link to the `/devices` directory for opening a fault-tolerant console
`/dev/ttymux:ctl`	Link to the `/devices` directory for issuing driver-specific ioctls. A specific minor device is required since all other devices may be in use.

Boot Time Configuration of the Console

If the OpenBoot PROM environment variables *input-device* and *output-device* are set to the `u4ftser` node in the PROM, UNIX attempts to configure fault-tolerant consoles during boot. Any failure during boot is written to the system console.

Three message severities are issued:

- Information
- Notices
- Warnings

Warnings should be regarded as serious events that require investigation. The only *normal* message to be expected occurs when a serial device is inaccessible because the hardware is offline, broken or not owned.

In the following, `%d` is a placeholder for a UNIX error code (see `intro`(2)), and `%s` represents a device path.

```
Information: rconsconfig: TCSETS %d
```

Whilst a serial device was being plumbed (configured) underneath the multiplexor, the boot code was unable to enable the receiver on the serial device. The consequence is that input will not be taken from that device until a successful ioctl is issued (any ioctl that modifies the terminal input or control flags, enables the receiver). When the login monitor for the console is configured, the receiver is enabled.

```
Notice: rconsconfig: TCGETS %d
```

Unable to read the `termios` control word from a serial device driver. This error causes the receiver to be left disabled, as in the previous error message.

```
Information: rconsconfig: associate error %d
```

The `ttymux` driver has been unable to associate a plumbed device with the console STREAM. The precise reason for the failure is reported in the system status log (look for lines labeled `ttymux`). If no consoles can be plumbed (configured), the system will still boot. Investigatory and remedial action can be taken when network logins have been enabled.

```
Information: rconsconfig: prop update failed
```

An update of the `console-devices` property of the multiplexor has failed. The impact of this failure is that both software and hardware aborting to OpenBoot PROM will not work until the redirection device (`cn driver`) is opened.

```
Information: rconsconfig: no dev info for mux
```

There is no device node for the multiplexor. The impact is the same as for the previous message.

```
Information: rconsconfig: device at %s is inaccessible
```

A target console device is inaccessible. This can happen if:

- There is no hardware at the path given in the `%s` string
- The hardware is unusable (broken or disabled by the OpenBoot PROM, or in use by the other half of a split system)

```
Information: rconsconfig: bad open on %s
```

A target console cannot be opened. Typically, this can occur if the hardware is broken, or if the hardware fails while the open is initializing the hardware.

```
Information: rconsconfig: bad mux link on %s
```

The `ttymux` driver has been unable to plumb a serial device. The precise reason for the failure is reported in the system status log (look for lines labeled `ttymux`). If no consoles can be plumbed (configured), the system will still boot. Investigatory and remedial action can be taken when network logins have been enabled.

```
Information: rconsconfig: bad close on %s
```

After successfully linking a console device, that device could not be closed. This can result in attempts to take the serial hardware offline to fail because UNIX assumes the device is still in use. Failure to close a device is symptomatic of a software bug and, as such, this message should never be seen.

```
Notice: rconsconfig: invalid console-flags - using 0x%x
```

A corrupt set of console flags has been read, so a default set has been chosen. The only standard way to repair them is from the OpenBoot PROM prompt.

```
Warning: rconsconfig: bad prop lookup
```

The property that specifies which consoles to multiplex, does not specify any consoles. The system still boots, but with no console access.

```
Warning: rconsconfig: autopush not set
```

Unable to configure the `autopush`(1M) facility on the console. This message means that the `ttycompat` and `ldterm` modules will not be pushed onto the console stream when it is opened.

```
Warning: rconsconfig: open %d failed (no console)
```

Unable to open the system console device. This message is generally a result of setting the OpenBoot PROM *input-device* and *output-device* variables to different or invalid paths. If the device corresponding to a given path is inaccessible, the path is considered to be invalid. If a valid console device cannot be found, the system will panic when the redirection console (cn driver) is opened. You must ensure that *input-device* and *output-device* reference a valid console.

```
Warning: rconsconfig: no real consoles plumbed
```

Unable to multiplex any real consoles. This error is reported in conjunction with one or more of the previous messages. It can occur if the list of consoles to multiplex is empty, or if all of the target consoles cannot be plumbed.

Configuring Serial Connections Using the CMS

Redundant (or fault-tolerant) serial connections can be configured using the CMS or the `ttymuxadm`(1M) utility. In fact, the CMS provides a front end to `ttymuxadm`. However, the CMS is the preferred method because it provides information about required devices, such as CAFs, serial chips and motherboards, and it integrates serial functionality into the overall management system.

▼ To Configure a Nonredundant Serial Connection

1. **Determine which file system entry in** `/dev/term` **will provide access to the new serial connection.**

2. **Type**:

```
cmsconfig
```

   If you have chosen **`/dev/term/0`**, the instance number `1` of `ft_serial` is displayed (`/dev/term/1` corresponds to `ft_serial 2`, and so on).

3. **Type the following command to include it in the display**:

```
i ft_serial 1
```

4. **Press return.**

   An entry for ft_serial 1 should be displayed. Enter its Item number. This will display a screen containing a list of the attributes that are relevant to configuring ft_serial serial objects.

   For example:

```
Select: ft_serial 1
Item    Name                    Value                          Page 1 of 1
-------------------------------------------------------------------------------
0       state                   offline
1       description             serial multiplexer
2       user_label
3       port0                   NULL
4       port1                   NULL
5       port2                   NULL
6       port3                   NULL
7       info
8       redundancy              not_present
9       mode_of_use             /dev/term/0
10      ports_in_use
11      working_ports
12      busylock                no

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

Note – If the `ft_serial` object is in the `not_present` state, it must to be configured.

5. **The four port fields correspond to the four possible physical CAF ports. To configure a nonredundant port, select the required CAF port:**
   - port 0 is the console port on the A-CAF
   - port 1 is the modem port on the A-CAF
   - port 2 is the console port on the B-CAF
   - port 3 is the modem port on the B-CAF

   For example, selecting Item 4 displays the following screen;

```
Modify: ft_serial 1 port1
Item    Value                                                  Page 1 of 1
--------------------------------------------------------------------------------
0       NULL
1       A-CAF_Modem

(H)elp, <Number> to set value or (Q)uit ?
```

6. **Enter `1` to configure a nonredundant serial connection (to the modem port on the A CAF) on `/dev/term/0`.**

   If the operation is successful, the `redundancy` attribute changes to `non_redundant` as follows:

```
Select: ft_serial 1
Item    Name                    Value                          Page 1 of 1
-------------------------------------------------------------------------------
0       state                   online
1       description             serial multiplexer
2       user_label
3       port0                   NULL
4       port1                   A-CAF_Modem
5       port2                   NULL
6       port3                   NULL
7       info
8       redundancy              non_redundant
9       mode_of_use             /dev/term/0
10      ports_in_use            A-CAF_Modem
11      working_ports           A-CAF_Modem
12      busylock                no

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

If the operation is unsuccessful (for instance, through trying to configure port2 on the A side of a split system), the `redundancy` attribute remains as `not_present`, and the `info` attribute indicates why the `redundancy` and `state` attributes have not changed:

```
Select: ft_serial 1
Item    Name                    Value                          Page 1 of 1
-------------------------------------------------------------------------------
0       state                   offline
1       description             serial multiplexer
2       user_label
3       port0                   NULL
4       port1                   NULL
5       port2                   NULL
6       port3                   B-CAF_Modem
7       info                    3 (unusable)
8       redundancy              not_present
9       mode_of_use             /dev/term/0
10      ports_in_use
11      working_ports
12      busylock                no

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

If the redundancy level (`not_present`, `non_redundant`, `dual_redundant`, and so on) does not correspond to the required number of configured ports the `info` attribute indicates which ports have failed to configure, and the reason why.

In the above example, port 3 has failed to configure because it is unusable. This is an indication that the associated port object is unusable (for example, its associated CAF may not be enabled).

Configuring a Redundant Serial Connection

Repeat Step 5 of the previous example by choosing a different port, such as port 2. If the operation succeeds, the `ft_serial` object is set to `dual_redundant`.

```
Select: ft_serial 1
Item    Name                    Value                          Page 1 of 1
-------------------------------------------------------------------------------
0       state                   online
1       description             serial multiplexer
2       user_label
3       port0                   NULL
4       port1                   A-CAF_Modem
5       port2                   NULL
6       port3                   B-CAF_Modem
7       info
8       redundancy              dual_redundant
9       mode_of_use             /dev/term/0
10      ports_in_use            A-CAF_Modem B-CAF_Modem
11      working_ports           A-CAF_Modem B-CAF_Modem
12      busylock                no

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

Successfully configuring other ports increases the redundancy level still further.

Configuring Highly Redundant Serial Connections

The maximal level of redundancy that can be configured using the CMS is `quad_redundant`, where all 4 CAF ports have been configured.

The number of devices that can be configured is controlled by one of the `ttymux` driver properties. Further devices can be configured into the redundant set using `ttymuxadm`. For example, if a Serial Asynchronous Interface PCI card is plugged into one of the PCI slots, port entries `/dev/term/a00[0-7]` can be configured into the redundant set using the command:

```
ttymuxadm -a /dev/term/a000 /dev/term/0
```

If more than four ports are configured in this way, the CMS displays the redundancy as highly redundant. However, only CAF ports show as being configured.

Setting Terminal Characteristics on Redundant Serial Devices

Terminal characteristics such as speed, parity, character sizes flow control, and so on, must be set consistently at both ends of a serial connection. Therefore, the terminal settings on the remote terminal device must match the settings on the port to which it is attached. When redundant ports are multiplexed, the only way to modify terminal settings is by means of the multiplexor stream associated with the underlying devices, and the settings will be applied to all connections. If the remote devices do not have the same settings, one of the pair will produce garbled I/O.

In addition, since input streams from redundant ports are interleaved onto a single stream, only character-orientated protocols can run over a redundant serial connection. Message boundaries are not respected; only raw character I/O is supported. If data messages are encoded in the input stream, they are likely to become corrupted with data received on the other input streams.

For example, typing:

```
# eval '/usr/openwin/bin/resize'
```

on a redundant console device fails if all the terminal devices are connected, and responds by returning their current window size.

CHAPTER 3

Split Mode

This chapter provides a detailed description of the function and use of split mode, and concludes with an example of how a Netra ft 1800 system is split into two independent sides and subsequently remerged. The chapter contains the following sections:

Split-Mode Architecture and Components

The Netra ft 1800 architecture enables a system running in fault-tolerant, or *combined*, mode to be split into two independent domains (or sides) running in split mode. 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, you can use one side (the losing side) to test upgrade procedures and services while the other side continues to provide the system services uninterrupted. This enables you to separate the possibly lengthy test and verification cycle from the upgrade procedure, which reduces overall service unavailability. You can also bring the services back on line on the upgraded side with minimal impact on their availability, and merge the two sides together to form a fault-tolerant Netra ft 1800 system running in combined mode.

The split mode software manages the orderly transition of the system from one mode of operation to the other, and coordinates the transfer of system resources between the domains.

When the system is in combined mode, it is composed of a single domain that encompasses all the system resources. In split mode, each independent domain owns resources that can be exclusively accessed by it. Each domain of a split system must contain all the resources required to allow it to function as an independent computing system. In addition, each domain may own other resources, possibly residing on the other side's motherboard, to provide services to the clients of the system.

Split mode should be viewed as a temporary state of the system and used only as a mechanism to minimize service unavailability when the system or application software needs to be upgraded.

High-Level Architecture

The split mode software has four basic components:

- User interfaces
- A daemon
- One or more communication networks
- Configuration files

You interact with the split mode software through commands or an Application Programming Interface (API) to:

- Make requests to change the mode of the system from combined to split or vice versa
- Request transfer of ownership of various resources from one side (or domain) to the other

The commands and libraries, in turn, communicate with the daemon, `u4ftsplitd`, which is responsible for translating your requests into actions that provide an orderly transition from one mode to another, or coordinate the proper transfer of resources among domains in a synchronous manner.

There is a daemon running on each side of a split system and these daemons synchronize their actions by communicating over a network interface. Split-mode software incorporates an Inter-CPUset Network (ICN) driver that supports the connectionless Data Link Provider Interface (DLPI) over the Netra ft 1800 bridges. The ICN driver enables network communication between the two sides of a split Netra ft 1800 system without external cabling. Both the daemon and the ICN driver use configuration files to ensure that they are set up properly. FIGURE 3-1 on page 41 shows the overall architecture of the split-mode software and the relationship between its components.

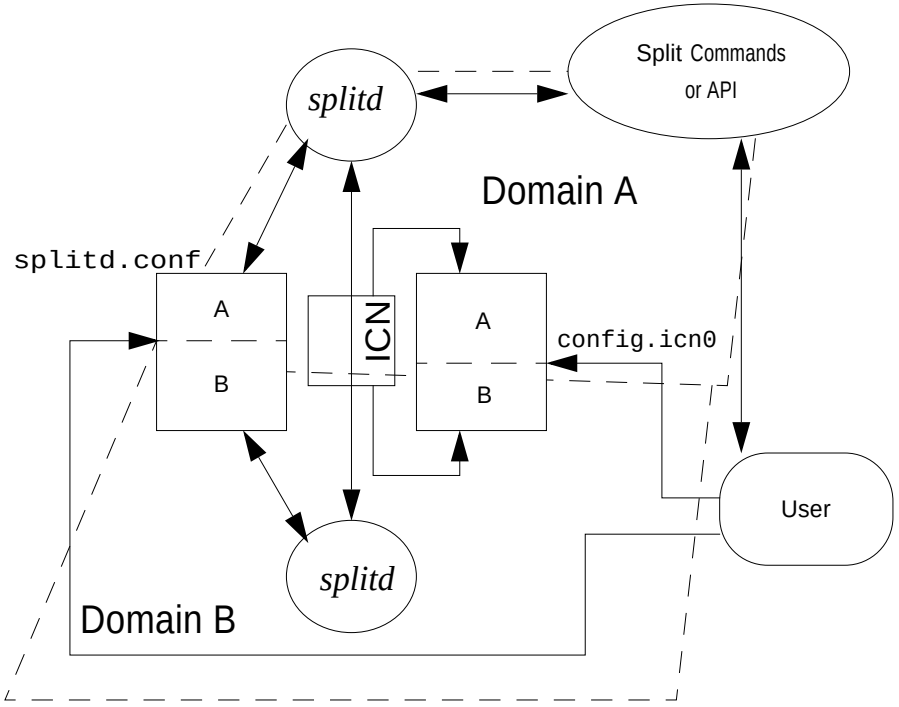

FIGURE 3-1 High-Level Architecture of Split-Mode Software

The operation of split mode can be described in terms of mode transitions and the ownership of system resources. When the system is in combined mode, the system is composed of one domain (or side) that encompasses all the system resources. In split mode, each domain must contain a minimal set of *fixed* (or unmovable) modules that are required to enable it to function as an independent computer. The fixed modules are: CPU, CAF, RMM, MBD, and PSUs. All other modules are considered *movable* and each side can own a set of these resources that you assign to it. Ownership of these modules can be freely transferred from one side to the other.

Fixed modules are statically owned by their *natural* domain when the system is in split mode. Each resource, except the motherboard (MBD), is owned by a single domain. The MBD has shared ownership as each side can own resources residing on the other side's motherboard. When the system is in combined mode, the assignment of ownership of modules is a preparatory step in the transition of the system to split mode. Therefore, in combined mode, the assignment of ownership of a resource to a domain that will exist only after the system is split is taken as prospective ownership of that resource should the system transition from combined to split mode.

Note – While the disk chassis (DSK) is considered a movable resource, each side of a split system must own a boot disk to allow it to operate as an independent computer. Furthermore, only the disk chassis itself is considered a resource—not the disks within the chassis.

Split-Mode Components

There are four basic components in the split-mode software:

- Split daemon
- User interfaces
- Communication networks
- Configuration files

In this section each of these components is described in some detail.

Split Daemon

The split daemon, `u4ftsplitd`, is always present in a Netra ft 1800 system, regardless of the mode (split or combined). The daemon is started in the boot sequence before the Configuration Management System (CMS), and is restarted automatically if it fails.

The daemon is responsible for the correct operation of the split-mode software. It also maintains the mode and ownership information on each side by communicating directly with the daemon running on the other side, if split, and by using EEPROMs and NVRAMs to maintain this information across a reboot. The daemon controls the split, transfer of ownership and merge operations by going through a sequence of actions that maximize system availability.

For split operation, the split daemon communicates with the CMS to disable the resources (other than the CPUset) that will be owned by the other side, and then causes an out-of-sync (OOS) event to move the CPUset to the losing side.

Note the notion of a loser and winner in a split operation. The winner continues to run the same services provided by the combined system, and sees no interruption to those services. Furthermore, the winner continues to run with the same identity as that of the combined system. The loser is placed at the boot prompt and, when booted, has a different identity.

As described above, when the system is in combined mode, the resource ownership is merely remembered for use when the system transitions to split mode. In both combined and split modes, the transfer of ownership of resources is initiated by the split daemon.

When the system is in split mode, transfer of resource ownership is managed by a *master* domain. The daemon running on the master domain is referenced before any transfer or ownership and coordinates the transfer. You can change the master domain so that failure of that domain, or the communication medium between the slave domain and the master domain, will not stop potential transfer of resources. In other words, the split daemon running on the master domain acts as a central coordinator for assigning the ownership of system resources on a split Netra ft 1800 system, and the master domain can be changed upon user's request.

When a request is made to merge two sides of a Netra ft 1800 system and form a fault-tolerant system, the split daemons running on the different sides communicate with each other to change the system synchronously back to a fault-tolerant state. The split daemon running on the merge winner communicates with the CMS running on that side to initiate the Processor Re-Integration (PRI) process, so that the losing CPUset can be brought back in sync with the winning CPUset. The split daemon running in the losing domain prepares its own side for this mode transition by a shutting down and then resetting its own CPUset so that the CPUset can be rebooted in preparation for a PRI process.

Once again, note that there is a concept of a winner and a loser in the merge operation. The winner of the merge continues to operate and offer services without any interruption, and the combined system inherits the identity of the winner. The loser's CPU is reset and waits for PRI to commence. Note also that there must be one winner and one loser in both merge and split operations. Once the system is back to combined mode, and possibly with a new identity, you can recreate the fault-tolerant pairs that existed in the system before it was split.

User Interfaces

The user interface for the split-mode software is composed of administrative interfaces and user libraries. The administrative interface has only three commands:

- `splitinfo`
- `splitconf`
- `splitadm`

The `splitinfo` command is a general information command that reports on the ownership of resources, the domain that is the split master, and the current mode of the system.

The `splitconf` command is used to assign ownership of resources, and changes the system configuration immediately if the system is in split mode. When the system is in combined mode, the `splitconf` command is used to set up a potential configuration for when the system transitions to split mode.

The `splitadm` command is used to request mode changes from the split daemon, and can alter the state of the system from combined to split, or vice-versa.

The split-mode libraries are designed to enable application programs to interact with the split daemon by means of appropriate APIs. There are two libraries: a single-threaded library that supports only the informational functions, and a multi-threaded library that supports all functions.

The informational functions are:

- `get_domain_attributes`
- `get_slot_status`

The function `get_domain_attributes` returns domain related information concerning the master domain, the mode of the system, and the domain on which the program is running.

The function `get_slot_status` returns split related information for a specific slot. This information is the location of the slot, the current owner of the slot, and whether the slot is fixed or movable.

The multi-threaded library has four additional functions:

- `set_domain_attribute`
- `set_slot_owner`
- `split_lock`
- `split_unlock`

The function `set_domain_attribute` assigns a specific attribute to the domains specified by the domain list. The attribute can be a request for a mode transition (to split or combined) or a request for a change to the master domain.

The function `set_slot_owner` sets the ownership of a specific slot (either potential or actual) to the specified domain.

The functions `split_lock` and `split_unlock` are used to synchronize changes to the domain attributes and resource ownerships.

Communication Networks

When the system is in split mode, the daemons running on each side must communicate with each other to synchronize ownership transfers or to make the change from split mode to combined mode. The channel of communication strongly recommended is that provided by the ICN driver. ICN provides standard network functionality (DLPI) across the bridges of a Netra ft 1800 system without the need for external cabling. There are four instances of the driver, one per path, and each instance is paired with its own subnet and host name pairs.

Configuration Files

The split daemon and each instance of the ICN require a configuration file; these are located in `/etc` and are called `splitd.conf` and `config.icn{0-3},` respectively.

The configuration file for the split daemon, `splitd.conf`, must be identical on both sides and needs to be modified (once per OS installation or upgrade) by the user. `splitd.conf` contains information concerning:

- The preferred channel of communication between the split daemons
- The port address of each side
- The timeout value for split mode operations
- The node names of each domain of a split system

One name is specifically reserved for the use of the split mode software, and that name is automatically assigned to the loser of a split operation. Therefore, this node name should not be used for assignment as a host or node name. The ICN can have up to four configuration files, one file per named interface (0 to 3). The corresponding ICN configuration files (the ones used for setting up a particular channel) differ on each side of a split system as each side has a different host name.

Interaction With Other System Components

As can be seen from the above component descriptions, the components of the split-mode software interact not only with each other, but also with other parts of the system, most notably with the OpenBoot PROM, low level system software and CMS.

Interaction with the CMS is organized by the split daemon so that split, merge and transfer of resource ownership happen in a consistent and orderly manner.

Interaction with the low level system software is needed to cause anOOS event. and is limited to direct `ioctl` calls to that layer of software. In addition to the OOS event, the split-mode software can use the `force` option (see “Force Option” on page 57) to request a device that cannot be quiesced to fail so that the resource associated with that device can have its ownership transferred.

Finally, the split-mode software interacts with the OpenBoot PROM so that the mode of the system and resource ownership information persist across a reboot. This is accomplished by writing the information to motherboard EEPROMs and CPUset NVRAMs that are maintained by the split daemon in both split and combined modes.

Split-Mode Operations

The split-mode software enables you to create two independent computers from a fault-tolerant Netra ft 1800 system, and recreate a fault-tolerant system from its two independent domains. However, these split and merge functions are complex operations that require significant preparation by the system administrator. It is assumed that Sun StorEdge Volume Manager (SEVM) is used for mirroring of all storage devices including the boot disk.

In this section the split and merge operations are described and restrictions highlighted. Between the split and merge sections is a description of the operation of a Netra ft 1800 system while it is in split mode and the features that might differentiate it from a standard Solaris system. Finally, the use of the `force` option is considered. However, the `force` option is a last resort and its use is not recommended.

Split

In order to split a fault-tolerant Netra ft 1800 system into two independent computers, the following steps are required:

1. Edit the split daemon and ICN configuration files.
2. Specify the split master.
3. Assign the desired ownership of the modules.
4. Prepare the disks under SEVM control.
5. Quiesce the devices residing on modules that will be owned by the loser of the split.
6. Split the system.

Note – Only after the preparatory steps are completed is it possible to split the system using the appropriate command or library call.

Editing the Configuration Files

Editing the configuration file of the split daemon, `/etc/splitd.conf`, involves identifying:

- The communication channel that will be used by the daemons once the system is split
- the node names of the two sides of a split system

If it is assumed that a fault-tolerant (or combined) Netra ft 1800 system has a *base* or node name, when the system is split for the first time it will have two names, the basename and the basename-2. For example, consider a Netra ft 1800 with the base name of `foo`. When split, the split winner will continue to operate with the same name, `foo`, while the loser, once booted, will operate with the name `foo-2`. The node names are recorded in the `hostnames` section of the split daemon's configuration file.

In addition to identifying the node names of the two sides, you must also identify the primary and, optionally, the secondary interface for communication between the split daemons running on each side of a split system. This is done in the domain address section of the `/etc/splitd.conf` file; the primary interface is identified in the `host_prim` line and the secondary interface is identified in the `host_alt` line.

The names used by the ICN have the suffix `-i`*x* added to the node names, where *x* is an integer ranging from 0 to 3. For example, to identify channel 0 of the ICN as the primary interface for inter-daemon communication, `foo-i0` and `foo-2-i0` are placed in the `host_prim` line of the `/etc/splitd.conf` file. To specify ordinary network interfaces as the alternative, or backup, interface, `foo` and `foo-2`, which are the node names of the two sides of a split system, can be placed in the `host_alt` line of the split daemon's configuration file. This configuration file is shown in CODE EXAMPLE 3-1.

CODE EXAMPLE 3-1 `splitd.conf` File

```
# The domain-address section
    host_prim  foo-i0  foo-2-i0
    host_alt   foo     foo-2
# 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
# The Hostnames section
    hostnames  foo     foo-2
```

`/etc/splitd.conf` must be edited once for each installation or upgrade of the operating system, and that this file must be identical on both sides of a split system.

Keeping the files identical on both sides of a split system is not enforced by the split-mode software. However, violation of this rule will lead to errors in split-mode operations. Therefore, when you use split mode to upgrade the operating system, you must edit the `/etc/splitd.conf` file on the upgraded side so that it is identical to the same file on the other side. Once you have modified the configuration file, either on a combined system or on one side of a split system, you must restart the daemon so that the new configuration can take effect.

ICN is the recommended primary channel for the inter-daemon communication. Assuming that interface 0 is used, you must modify the `HOSTNAME` line in the `/etc/config.icn0` file of the combined (fault-tolerant) system to incorporate the base name of the fault-tolerant system.

In this example, that would mean that the `HOSTNAME` line is modified to `foo-i0`.

Note – The split-mode software automatically modifies the `/etc/config.icn0` file of the losing side so that the `HOSTNAME` of this side is `foo-2-i0`. If you upgrade the operating system, you must restore the `/etc/config.icn0` file so that it contains the correct `HOSTNAME` entry.

Furthermore, note that the `HOSTNAME` entries in the ICN configuration files differ on the two sides of a split system. You must ensure that the configuration files for both the split daemon and the ICN are correct before attempting a split mode operation.

CODE EXAMPLE 3-2 shows this ICN configuration file.

CODE EXAMPLE 3-2 ICN Configuration File `config.icn0`

```
# ident “@(#) config.icn0 1.3 98/11/03 SMI”
# icn config script
PARENT_PATH=/pci@6,2000/u4ftbus@0/u4ioslot@6
DEV_PROPS=”reg=0x82803000,0x00000000,0x58000000,0x00000000,0x04000000”
HOSTNAME=foo-i0
```

Specifying the Split Master

In a split Netra ft 1800 system, only one side of the system can be the split master and you must ensure that this is the case. It is important to specify the split master, and it is recommended that the surviving side be chosen as the split master before a split operation is attempted. The split master— either prospective, if the system is in

combined mode, or actual, if the system is in split mode—can be modified by the `splitconf` command with the `-m` option. You can also view the prospective or actual master domain using the `splitinfo` command with the `-m` option.

Assigning Ownership of Resources

Ownership of slots that contain movable modules can be assigned to either of the sides before the system is split.

- To view the current ownership of all slots in the system, use the command `splitinfo -l all`.
- To change the ownership of a slot, use the command `splitconf` with the `-l` and `-o` options.

Before the system is split, ensure that ownership of resources matches the intended configuration of the split system. The slot occupied by the disk chassis is considered a movable slot. However, there are only two such slots (one per side) in a Netra ft 1800 system and, if each side of a split system is required to boot, each side should own a disk chassis. Note, again, that disks within a chassis cannot be individually owned by different sides—they are owned by the side that owns the chassis.

Preparing the Storage Devices

There are two types of storage device that can affect a split operation: boot and non-boot (or data) disks. The split-mode software requires that the boot disk of a fault-tolerant Netra ft 1800 system is encapsulated and mirrored, and that each mirror is bootable and resides on a different side of the system (in a different disk chassis).

These requirements are necessary, but not sufficient, to ensure that the loser of a split operation will be able to boot. To ensure that the loser of a split operation will be able to act as an independent computer using the Solaris operating environment, the boot disk must not extended beyond one physical disk. In other words, the root disk group, `rootdg`, must be composed of exactly two disks—the boot disk and its mirror—and the mirror must reside on a different side from the boot disk.

In addition, the boot disk and its mirror must contain all the file systems required for the correct operation of the system. These are the `root`, `swap`, `/usr`, `/etc`, `/var`, `/tmp` and `/opt` file systems.

These restrictions imply that each plex on the boot disk must have a valid mirror on the other side. The split-mode software will automatically handle the boot (or root) disk of the split loser and you must not take any action on the mirror of the boot disk before the split operation is attempted. You must, however, follow the restrictions outlined above.

Given that the `rootdg` can contain only the boot disk and its mirror, all other disks must, therefore, belong to other disk groups. These are called the data disks, and before a split operation, you must remove from SEVM control those data disks that will be owned by the loser of the split operation. This is necessary because a disk cannot be disabled while it is in use by SEVM or any other application. In fact, all modules that will belong to the losing side of a split operation must be disabled before the system can be split, and most modules cannot be disabled if they are in use by an application.

The act of discontinuing the use of a device in preparation for disabling it is commonly referred to as *quiescing* that device. In order to remove from SEVM control non-boot disk devices that will belong to the loser of a split, you must first remove the disk from its disk group. This will then allow the disk to be removed from SEVM control and be quiesced.

Quiescing the Devices

To disable resources that will belong to the loser of a split operation, there must be no processes using the devices associated with these resources. Assigning modules that contain open devices to the new system causes the split operation to fail, as these modules will not be disabled and the split operation will time out waiting for them to be disabled.

The split daemon attempts to disable all modules that will be owned by the loser of the split operation before splitting the system. Each disable operation is expected to terminate within a timeout value. The default value is 60 seconds and this can be modified, either by a `splitadm -t` command, or by modifying the split daemon's configuration file where the timeout value is expressed in seconds.

If one or more of the modules cannot be disabled within the specified time, an error message is displayed, identifying the list of modules that failed to disable. The split process then stops. At this point, all the modules that have been successfully disabled remain disabled and the system remains in the combined (or fault-tolerant) mode.

The operation can be retried with a longer timeout so that the devices can be quiesced. If increasing the timeout value fails, there are probably one or more busy devices that cannot be disabled. At this point, you may need to stop the use of these devices by all applications so that the split operation can succeed.

As it may not be possible to quiesce some devices, the split-mode software provides a `force` option. However, use of this option is not recommended, for the reasons described in "Force Option" on page 57, and is only acceptable as a last resort.

Splitting a Netra ft 1800

Once the fault-tolerant system is prepared, it can be split using the `splitadm split` command. Use the command with the:

- `-t` option to adjust the time needed for disabling a module, as discussed earlier
- `-w` option to identify the winning side

By default, the split master side is the winning side. If errors occur following the split command, verify the preparatory steps and issue the command again. If the command fails again, increase the timeout values to give modules more time to disable their devices, then re-issue the command.

If increasing the timeout values also fails, there are probably one or more devices that cannot be quiesced. You can identify these devices by using `cmsconfig`. The devices that cannot be quiesced will go to the loser of the split operation and will be in a `disable_failed` state.

While in Split Mode

Splitting a Netra ft 1800 system does not affect the winning side. This side continues to operate with the same identity as before and offers the same services as the fault-tolerant system. However, the system is no longer fault tolerant. While the system is in split mode, you can use the new side to test and verify upgrade procedures. Once these steps have been completed, you can use the split system to re-establish a (possibly different) fault-tolerant system.

When the split operation has completed, the new system can be booted. Two boots are performed with one `boot` command. During the first boot, you must provide one or two SEVM license keys. On the second boot, the system loads the new identity that was set up in the configuration files. The CMS configuration is inherited from the original combined system. Therefore, modules belonging to the winning side are in the `enable_failed` state. These modules should be disabled on the new side. Modules belonging to the new side are enabled automatically by the CMS initialization process.

Transfer of Ownership

While the system is in split mode, you can transfer the ownership of resources from one side to the other using the `splitconf` command with the `-l` and `-o` options. Provided that it is running, the side that must release the resource automatically disables the modules in it. The operation fails if a module contains devices that are in use. For transfer of ownership to succeed, the module must be in a state in which its devices can be successfully taken offline. Once the new resources have been acquired, you can enable them using `cmsconfig.`

Note – Any attempt to enable modules that belong to the other side will fail. The transfer of ownership is acted upon immediately in a split system, whereas, in a combined system, this is viewed as setting up a potential future split configuration.

When each side of a split Netra ft 1800 system has a running split daemon and the daemons can communicate with each other, either side can initiate a transfer of ownership. However, a failure can occur while the system is in split mode. To minimize the time the service is unavailable, resources can be transferred from one side to another, even if one side is not running. If there is no communication between the two daemons, the side that is not the master refuses any attempt to transfer ownership. If the side requesting the transfer is the master, the transfer is performed and the other side learns about the configuration as soon as communication is re-established.

However, note the following points:

- Mastership can be acquired by the non-master side if the master side goes down.
- 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 after communication has been restored between the sides.
- If both systems believe that they are the master, the system whose split daemon has been running longer is deemed to be the master.
- If the two sides are not communicating and both are running, each may believe that it is the master and attempt to accept the ownership of the same slot. This results in a contradictory state as far as the split daemon is concerned.

You must ensure that there is only one master in the system. You can confirm that the two daemons are communicating by using the `splitconf` command with the `-m` option. To ensure that both daemons are sending and receiving messages, transfer the mastership to the non-master side and back to the original master side. If the sides are not communicating, an error message is displayed. It is often possible to re-establish the communication, particularly when ICN is used, by restarting one or both of the split daemons. Do this by issuing a `/etc/init.d/u4ftsplit stop` command followed by a `splitinfo` or `splitconf` command on each side.

You must also ensure that the two daemons are communicating when a split Netra ft 1800 system is being prepared for a merge operation. If both sides are running Solaris and communication is lost between the two daemons, an inconsistent state can be reached after the merge operation. The inconsistency is due to one side being in split mode while the other side is in combined (or configured fault-tolerant) mode. The inconsistency does not persist across a reboot of the side that believes it is in split mode.

Restrictions and Notes

The following restrictions apply to operations while the system is in split mode:

- After a split operation, the loser of the split must go through a complete reset and reach the `ok` prompt. Until this phase is completed, do not attempt a merge operation. This will lead to serious inconsistencies in the mode of the system and result in service unavailability.
- When a motherboard (MBD) and Power Supply Units (PSUs) are physically located on the other side, none of these modules should be viewed as being owned exclusively by the side on which it resides.

  In addition, consider carefully when any of these modules is to be disabled as some resources that reside on the MBD or draw current from the PSUs may actually belong to the other side. The split mode software does not enforce this restriction, but you must observe it for the correct operation of the system.

Merge

The merge operation changes the system from split mode to combined mode. This operation creates a fault-tolerant system from two independent sides of a Netra ft 1800 system. Either side of a split system can request a transition to combined mode by issuing a `splitadm -w winning_side merge` command or its equivalent API call. The resulting fault-tolerant system has the same identity as the winning side. The winning side continues to run the same services without interruption, while the other side is shut down.

Note – The winning side must be identified as there is no default value.

Daemon Synchronization

Creation a fault-tolerant system, in which the CPU sets on each motherboard run in a lock-step fashion, requires considerable coordination and synchronization, and this is provided by the split daemon, `u4ftsplitd`. While the system is preparing the transition from split mode to combined mode, the daemons running on each side communicate to ensure that this transition happens in an orderly and consistent manner. Therefore, you must ensure that the two daemons are communicating with each other before attempting the merge operation.

The split daemon on the loser of the merge operation:

- Communicates with the other split daemon
- Writes the mode of the system as combined to the local NVRAM
- Shuts down and resets its CPUset

This invokes the OpenBoot PROM, which will notice that the system mode is set to combined and that this side is the loser of a merge operation. The OpenBoot PROM running on the losing side of the merge operation causes the system to wait for the PRI process.

The split daemon on the winner of the merge operation:

- Writes the mode of the system as combined in EEPROMs and the local NVRAM
- Communicates with the CMS to initiate the PRI process and form a fault-tolerant system

---

Note – The winner writes the new mode of the system to both EEPROMs and its local NVRAM, or a majority of persistent storage devices used by the OpenBoot PROM. This prevents a potential inconsistency, if communication between the daemons is faulty, from persisting across the reboot of the system.

Once the system is configured to be in fault-tolerant mode, the modules that previously belonged to the loser of the merge operation can be enabled and brought back online. These modules are now part of the combined system. The system is configured as fault tolerant as soon as the writes to the persistent storage are completed by the split daemon. This means that you can enable such modules and start to re-establish the desired fault-tolerant pairs while the PRI process is in progress.

Furthermore, ownership is no longer applied when the system is returned to combined mode as all resources in a fault-tolerant Netra ft 1800 system belong to the single fault-tolerant domain.

Use the `cmsconfig` utility:

- To enable the modules that used to belong to the loser of the merge operation
- For accurate information regarding the actual state of the system (fault-tolerant or not) and the actual state of the system's resources.

You cannot rely on the output from `split` commands as it only reflects the system view of the split daemon and can be inaccurate. For example, a system that is going through a PRI process can be declared combined by the output from 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
- Slot ownership (either prospective or actual)

Implications for SEVM

To return the system successfully to a fully fault-tolerant state with mirrored disks following a merge operation, you must re-establish the mirrored pairs of disks that existed in the system before the initial fault-tolerant system was split. This requires some preparation on the losing side before you issue the merge command, and some actions following the completion of the merge. These steps are demonstrated below using an example.

The following assumptions are made:

- The original system was split with side A as the winner.
- Side B had its operating system upgraded while side A continued to provide services running on the original system.
- The upgrade was successful and side B was chosen as the winner of the merge.
- The data was updated on side A while the system was in split mode.

The following operations are necessary:

- Add the data disks from the losing side A to the new system and mirror the data on them to disks on side B.
- Mirror the boot disk on side B to the boot disk on side A so that the new system runs the upgraded operating system with mirrored boot disk.

The steps to accomplish this task are:

1. Deport all the data disk groups that belong to the side A.
2. Perform the merge, specifying side B as the winner.
3. Wait until the system comes back into sync.
4. Use the CMS to enable the disk chassis residing on side A (A-DSK) and the disks in it.
5. Import the data disks to the new system.
6. Start the volumes and mount the file systems.
7. Resync the volumes.
8. Remirror the new boot disk (residing on side B) on to the boot disk of side A.

Caution – The boot disk of the loser of the merge operation must be placed under SEVM control before you attempt to reboot the winning side. If you fail to do so, a full re-install will be required, and all the benefits of the split mode operation will be nullified. The reboot of the winning side, however, is not a *requirement* of the merge operation.

System Identity Issues

The system identity can be defined as a set with four elements:

- A hostid
- A node name
- An array of resources (software and hardware)
- An operating system image

This set can be used to define the split and merge operations in terms of system identity.

If a combined Netra ft 1800 system has the set

```
{hostid0, nodename0, resources0, os0}
```

as its identity, the winner of the split operation on that system will have as its identity the set

```
{hostid0, nodename0, resources1, os0}
```

where `resources1` is a subset of `resources0`. The loser of the split operation, on the other hand, will have the identity

```
{hostid1, nodename1, resources2, os1}
```

where the only certain relation between this set and that of the original fault-tolerant system is that `resources2` is a subset of `resources0`.

Note that you can modify the resources array in a system, for example, by adding a new IP address (an example of a software resource) or a new disk (an example of a hardware resource), whereas there is a unique and a one-to-one mapping between a system and its hostid and nodename. Therefore, it is possible to modify the sets `resources1` and `resources2` so that an element of the set `resources1` can be transferred to the set `resources2`; for example, by transferring a network interface card from one side of a split system to the other side and inheriting its associated IP address (an example of both a hardware and software resource transfer).

Note, also, that, in the merge operation, the combined system initially inherits all the system identity of the merge winner. It can modify its resource set after the merge operation by, for example, enabling the network interface cards that belonged to the loser of the merge operation or by completing the PRI process.

Since a Netra ft 1800 system has the potential to be split, two hostids are reserved for the platform. In combined mode, one of the two hostids is used. When a new split side is created, it is assigned the alternative hostid, while the surviving side retains the hostid of the original combined system. The same applies to the node name. When a new split side is created, it must be assigned a node name different from that of the original combined system. The two node names are specified in the line `hostnames` of the `splitd.conf` file.

After merge, the host id and host name of the resulting combined system coincide with the host id and host name of the winner of the merge operation. Depending on the circumstances, the hostid and node name may or may not coincide with the hostid and node name of the original combined system. It is important to consider carefully the identity of the final system and to ensure that the correct surviving side is specified.

For example, if the original combined system had an identity `hostid0`, `nodename0`, and the final merged system has identity `hostid1`, `nodename1`, you can either accept the new identity and take no action or restore both the original `hostid0` and the original `nodename0`.

To restore both the original host id and the original node name, split the system again, select the split winner as the side with `nodename1` and `hostid1`, then merge the two sides again, selecting the merge winner to be the side with `nodename0` and `hostid0`. These operations restore both the host id and node name of the original system.

Use secondary network interfaces to service client applications. These interfaces can be disabled on one side, then transferred and enabled on the other side without implications for the system identity or the need for reboots. View these connections as a system resource that can be transferred from one side of a split system to the other side and start to serve clients before the merge operation is attempted. If you follow this simple advice, your system will take full advantage of split-mode operations and will cause minimal impact to the availability of service when you need to upgrade the system or application software.

Force Option

The `-f` option to the `splitadm` command forces the system to be split, even if one or more modules that belong to the new system fail to disable within the specified timeout.

Similarly, the `-f` option forces transfer of ownership, even if the module specified by the `-l` option fails to disable within the specified timeout.

Caution – Use of this option can cause serious loss of data if you have not previously quiesced all devices in the relevant modules.

You should use the force option only as the last resort for splitting or transferring ownership in a Netra ft 1800 system: the normal procedures for the split and transfer of ownership operations should be used whenever possible. However, if it is not possible to disable some slots by normal procedures because one or more devices on that slot cannot be quiesced, there may 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 the `splitinfo` command and the `cmsconfig` utility.

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 hostid of the combined system, and that the force option was used because the B-PCI0 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 B-PCI0 on side A before merging the system. If this device is needed by the combined system, and the winner was side A, you may have to reboot the combined system before this device can be enabled.

- Transferring Ownership using the Force Option

  Assume you are going to transfer ownership of a module—for example, B-PCI0—from side A to side B using the force option. In this case, the subsequent merge winner must be side B, or B-PCI0 must be disabled on side A if the subsequent merge winner is going to be side A. If this device is needed by the combined system, and the winner was side A, you may have to reboot the combined system before this device can be enabled.

Note that the above restrictions apply individually and in combination. You must plan very carefully to avoid creating an unusable system (which you will need to shutdown and reboot) 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. Finally, note that using the force option for splitting can result in the loser's motherboard being marked as `failed`. This is not a real failure and the motherboard should not be considered to have failed.

Software Upgrades Using Split Mode

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

Assume that the Netra ft 1800 system that is going through an upgrade cycle is:

- Called `foo`
- Running a version of the Solaris operating environment

- Running in combined mode

Furthermore, assume that the operating environment is to be upgraded to the next version, and that the application program developed for this new release of Solaris is also to be upgraded. Now consider the techniques that can be used to minimize the unavailability of service.

The first issue is that of system identity: you need to decide about the final identity of the system. This decision is based on two factors:

- The applications that are running in the system
- How the system is viewed by its external clients

If you intend to minimize the unavailability of service due to software upgrades, you should ensure that portions of the system identity that are unique; that is, its host id and node name, are transparent to the applications running on the system. You should also provide the connections from the servers running on the system to their external clients on secondary interfaces that reside on movable modules. These two key decisions enable you to avoid the system identity issue and to concentrate, instead, on the actual upgrade procedure and minimize the service unavailability. Note that these decisions need to be made when the services of the system are initially set up, which could be long before a need for a software upgrade is apparent.

A second related issue is that of the *type* of the application. An application can be either *stateless* or *stateful*. If an application is stateless, that application does not need to recover its previous state before it can provide a service. Many applications, however, are stateful and require a recovery of their previous state before they can operate correctly. These applications must checkpoint and recover their state if they need to be restarted. Many of these applications already have mechanisms for doing this (for example, database management systems) and you can use such mechanisms to set up the services correctly when such an application is upgraded. The split-mode software does not provide any such facilities and you must ensure that stateful applications are started correctly following an upgrade.

A third area, to be considered during the initial set up of the system, that can impact the usefulness of split mode operation is that of disk layout and volume management. If an application is likely to be going through an upgrade procedure, that application should have its own disk group, in the SEVM sense, so that it can be mirrored independently of other data and applications.

FIGURE 3-2 shows the data layout of the initial fault-tolerant system from the volume management perspective in the example. Note that the root disk group is placed in exactly two (mirrored) disks and that no other data is kept on these two disks.

Similarly, the application software to be upgraded is kept on its own disk group, `dataxdg`, and mirrored on disks A-DSK1 and B-DSK1. The data, which is assumed to be read-write, is also mirrored and is in the `dataydg` disk group.

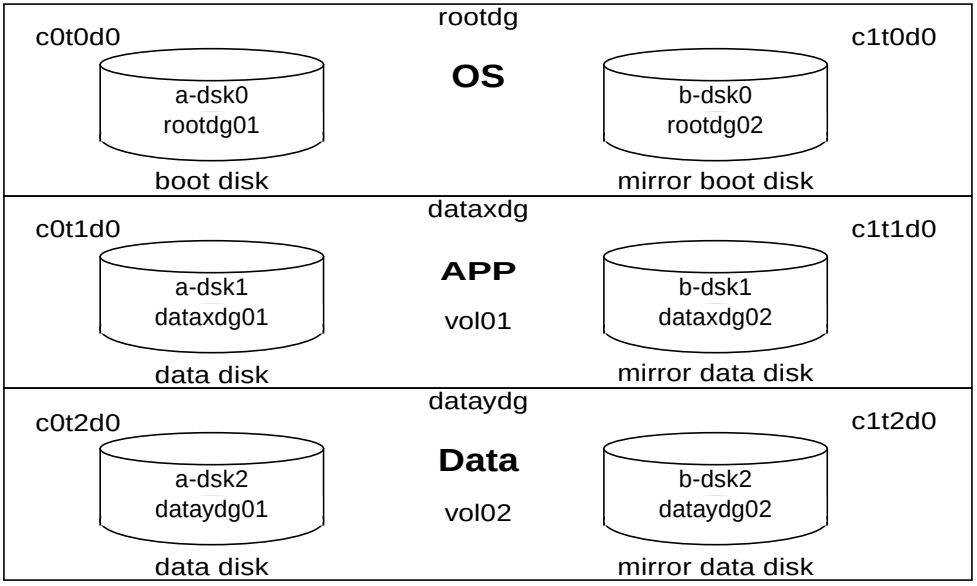

FIGURE 3-2 The SEVM Layout of the Initial Fault-Tolerant System

To minimize service unavailability due to upgrades, at least two fault-tolerant networks are on the original fault-tolerant system, while one is on a movable resource. The primary connection (this is the base name or the node name of the original fault-tolerant system) is called `foo`. An optional secondary connection is called `foo-1` and the mandatory secondary interface is called `foo-3`. Note that both the primary connection, `foo`, and the secondary connection `foo-1` are on the CAF, which is a fixed module in the split-mode sense.

Therefore, `foo-3` is used for the service that the application is providing, as this software resource is on a movable PCI module. The naming convention used here is reflected in the relevant ICN and split daemon configuration files where the name

foo-2 is reserved for the loser of a potential split operation. The system, at this initial stage, is shown in FIGURE 3-3, and the split daemon and ICN configuration files are shown in CODE EXAMPLE 3-1 on page 47 and CODE EXAMPLE 3-2 on page 48.

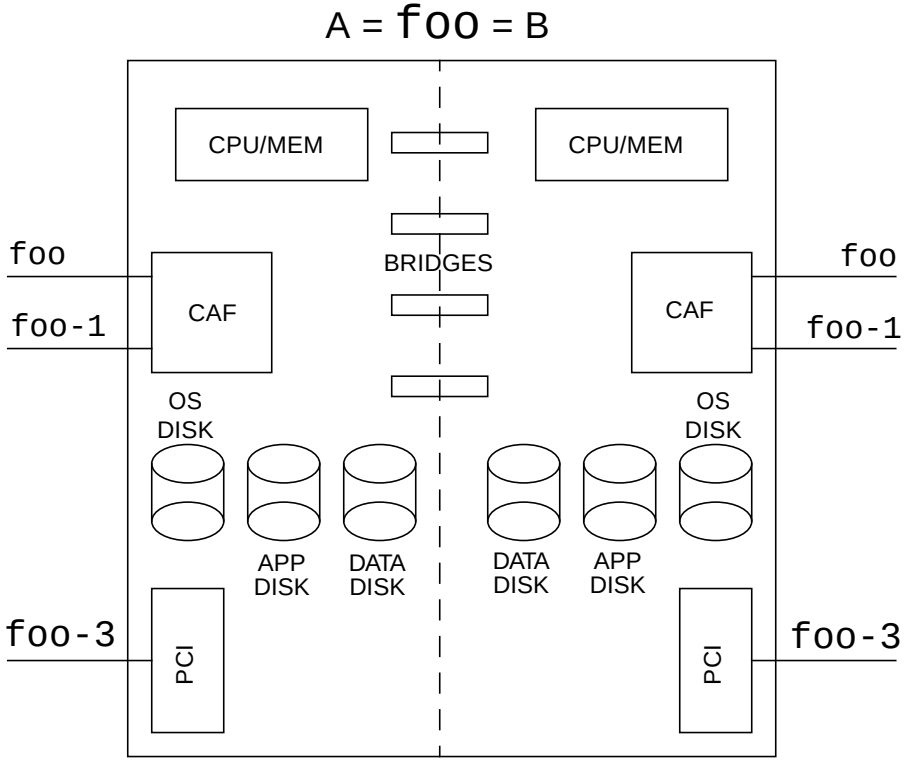

FIGURE 3-3 The Initial Set Up of the Fault-Tolerant System

Recall that you want to upgrade both the operating system and the application. You can split the system into two sides where side A is chosen as the winner of the split operation. Assume that you have taken all the required steps for the split operations and that side A continues to run the service provided by the fault-tolerant system on its secondary interface, foo-3. To minimize unavailability of service, the network foo-3 (the connection of the server to its external clients) is kept as a fault-tolerant network.

When the split operation has completed, you can boot the loser of the split, side B, with its new identity. On this system, called foo-2, you can upgrade both its copy of the application software and its operating system.

Meanwhile, it is possible that the server running on side A has updated the data that will be used by the upgraded server now running on side B. Note that the split daemons running on each side are communicating with each other using channel 0 of the ICN. The system, at this stage of the upgrade procedure, is shown in FIGURE 3-4.

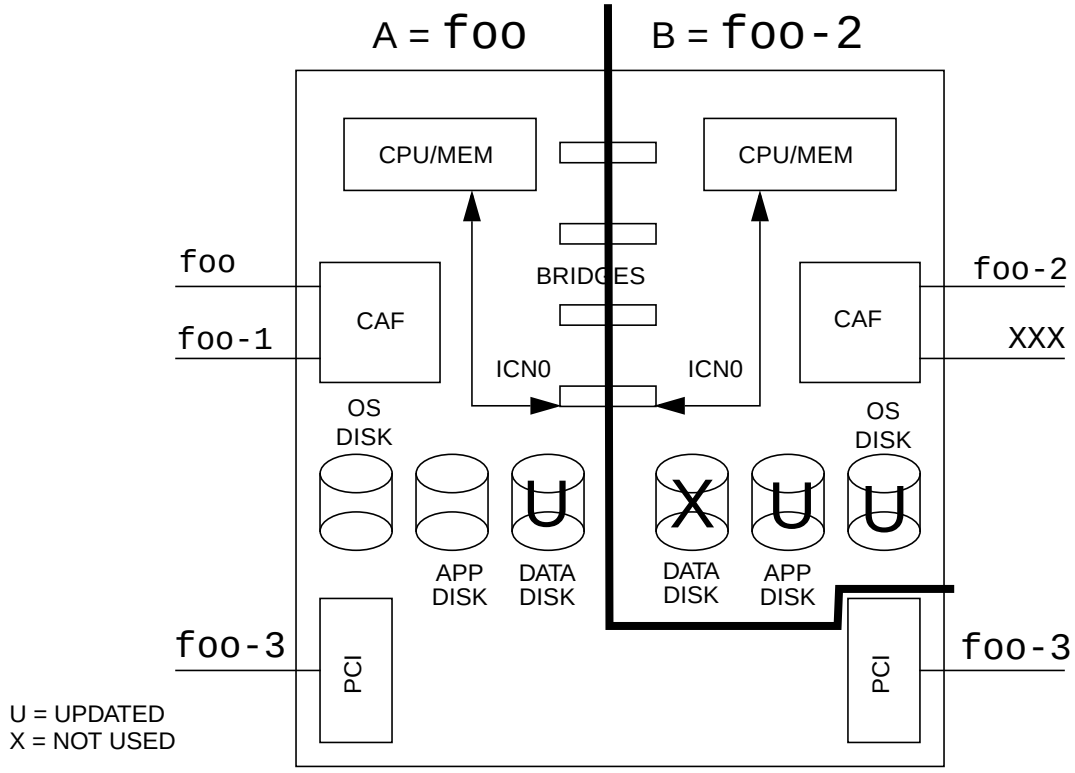

FIGURE 3-4 The System in Split Mode

To minimize service unavailability, you must:

1. Transfer ownership of the backup PCI card (residing on side B in this example) to side B.
2. Enable it using `cmsconfig`.
3. Define it as a component of the ft network interface, `foo-3`, without enabling the `ft_network`.
4. Disable the primary PCI card (residing on side A), and the `ft_network` service that uses it, with the `cmsconfig` utility running on that side.

5. Immediately enable the `ft_network` service using host name `foo-3` on side B, with the `cmsconfig` utility running on side B.

Note – In order to disable the primary PCI card, that card must have been quiesced.

At this point, if the service is stateless, the service can be restored and clients of the service will see the entire upgrade procedure as the loss of their connection to the server. These clients will only need to reconnect to the server (running at the same host, `foo-3`) to take advantage of the upgraded service, which includes both an upgraded application and an upgraded operating system.

You can now merge the system and re-establish the fault-tolerant pairs. The new system will have a different identity (host id and node name) from the original fault-tolerant system, but that is not an issue if the applications, connections, and disk layout are properly thought out when you initially set up system.

While this procedure is sufficient for stateless servers or new services running on the system, the procedure for upgrading stateful applications is more complex and, therefore, more time consuming. This is because check pointing and recovery of state must happen before the service can be correctly restored. The check pointing of the current state must be done before you merge the system and restart the upgraded service.

Note – This should happen after the service on the primary PCI card has been disabled. Once the system is merged, you can recover the state of the server application and restore the service.

FIGURE 3-5 shows the new fault-tolerant system after a successful merge operation where the direction of data mirroring indicates that the data was updated on side A while the operating system and the application software were upgraded on side B. In this figure, the fault-tolerant pairs for the network connections are restored to the original configuration, with the exception of the primary interface, which is now called `foo-2`.

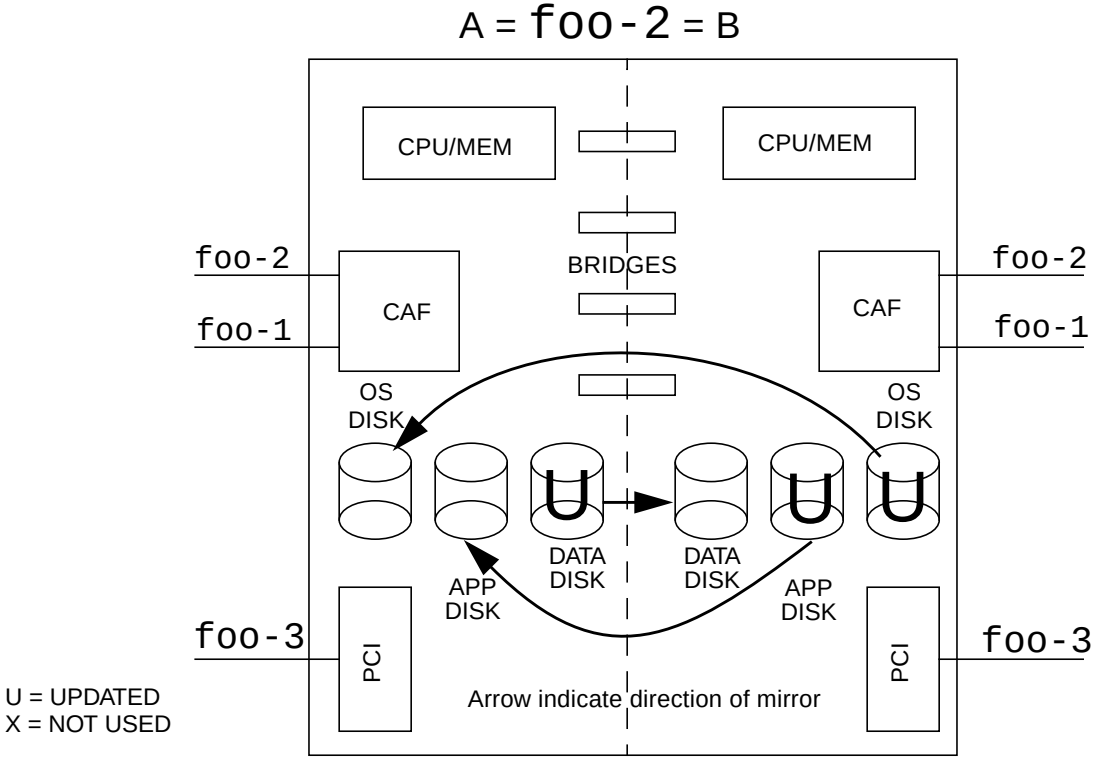

FIGURE 3-5 The Final Fault-tolerant System After the Upgrades

Split Example

The following example demonstrates how to split a Netra ft 1800, upgrade a component such as an software patch, and then remerge the system. Each stage of the split process is documented, with a description of the state of the system at the end of each stage.

The procedure documented here is an instructor-led demonstration of split mode on the Netra ft 18000 system. As such, it is not intended as a tutorial, but provides one example of the use of split mode. The procedure should be used in conjunction with the *Netra ft 1800 User's Guide* (part number 805-4529-10).

This procedure assumes that you have some user-level knowledge of SEVM 2.5.

Assumptions

- If NIS is running, edit the /etc/nsswitch.conf file so that the line:

```
hosts: xfn nis [NOTFOUND=return] files
```

  reads

```
hosts: files xfn nis [NOTFOUND=return]
```

  `files` is now the first option on the line, which will force the system to read the local `/etc/hosts` file first.

- Before the split, the combined system has the host name `seagoon` and IP address 129.156.203.152. After the split, side A will be the split winner with host name `seagoon` and side B will be the loser with the host name `seagoon-2` and IP address 129.156.203.18.

- OBP (PROM) variables:
  - *use-nvramrc?*=true
  - *auto-boot?*=true
  - *boot-device*=a-dsk0 b-dsk0
  - *diag-device*=a-dsk0 b-dsk0
  - *diag-switch?*=false

  Check using the `eeprom` command and, if necessary, set using:

```
eeprom use-nvramrc?=true
eeprom auto-boot?=true
eeprom boot-device=a-dsk0 b-dsk0
eeprom diag-device=a-dsk0 b-dsk0
eeprom diag-switch?=false
```

  The *boot-device* PROM variable is concatenated with an NVRAM variable, *boot-list*, to produce a sequence of disks from which the system tries to boot. The *boot-list* variable is constructed from a motherboard EEPROM variable, which comprises the HDDs in the SEVM root disk group. Setting *boot-device* to the value above ensures that the system will try to boot from A-DSK0 and, if this fails, from B-DSK0. Since these HDDs are mirrored, either should produce a current copy of the boot disk.

- All devices on the losing side should be quiesced, that is. no devices using losing side FRUs should be used by any applications.
- Disk configuration:
  - `rootdisk` is the SEVM representation of `a-dsk0`, which is the disk device `c2t0d0`;
  - `disk01` is the SEVM representation of `b-dsk0`, which is the disk device `c3t0d0`.

Initial Configuration

Machine: 2P 512Mb Netra ft 1800

Operating System: Netra ft 1800 Update 01 CD , supplemental CD , SEVM 2.5 (including patch 105463).

FRUs present: HDD in `A-DSK0` (`/dev/rdsk/c2t0d0`) called `rootdisk` in SEVM, mirrored in `B-DSK0` (`/dev/rdsk/c3t0d0`) called `disk01` in SEVM. No other HDDs are present in the machine. No PCI devices are attached.

Node name: `seagoon` (IP address 129.156.203.152)

Initial configuration (`cmsconfig`): As shown in FIGURE 3-6

```
Item  Name          Fault Loc     State                 Page 1 of 1
-------------------------------------------------------------------------
0     A-MBD 0             A-MBD   enabled
1     B-MBD 0             B-MBD   enabled
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   enabled
8     HDD 0               A-DSK0  enabled
9     HDD 7               B-DSK0  enabled
10    ft_alarm 0                  usable
11    ft_core 0                   enabled    Fault tolerant
12    ft_network 0                online_up  A (online) & B (online)
13    ft_serial 0                 online
14    icn_system 0                disabled   System is not split

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

FIGURE 3-6 Initial CMS Configuration

Notice the presence of:

- two motherboards (MBDs)
- two CAFs
- two CPUs
- two disk chassis (DSKs)
- two hard disks (HDDs)
- an `ft_alarm` (the alarms subsystem)
- an `ft_core` (the fault-tolerant virtual machine)
- `ft_network 0` (a fault-tolerant network)
- `ft_serial` (the serial subsystem)
- `icn_system`

Notice that the `icn_system` entry is disabled and the message states `System is not split`. The `icn_system` enables two sides of a split Netra ft 1800 system to talk to each other. In combined mode, this should be in a `disabled` state as it is here.

Check that the system (`seagoon`) is in combined mode using the command:

```
seagoon# splitinfo -a
attributes=combined
```

Ensure that the Netra ft 1800 combined system is not acting as a router. This is accomplished by the existence of the file `/etc/notrouter`, which can be created by issuing the command:

```
seagoon# touch /etc/notrouter
```

Configuring for Split Mode

This example shows how to split a Netra ft 1800 with nodename `seagoon` into `seagoon` and `seagoon-2`. While `seagoon` is left running on side A, `seagoon-2` will be updated on side B, and then the two sides merged to reform a combined, or fault-tolerant, Netra ft 1800 system.

You may find it useful to have two xterms open, one for each of the following:

- A-side console, to monitor the progress of the split on the A-side
- B-side console, to monitor the progress of the split on the B-side

The following steps are required to prepare to split a system.

1. Edit the `splitd.conf` file
2. Set up ICN
3. Set the split master
4. Add other IP addresses to `/etc/hosts`
5. Configure `ft_network 0`
6. Check SEVM status
7. Check CPUs are in sync
8. Check split daemon status

Editing the `split.conf` File

FIGURE 3-7 shows the contents of `/etc/splitd.conf`, naming `seagoon` and `seagoon-2` as the two host names to be used following the split.

```
# 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 seagoon-i0 seagoon-2-i0
host_alt seagoon seagoon-2

# 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 seagoon seagoon-2
```

FIGURE 3-7 The `/etc/splitd.conf` File

Setting Up ICN

Use `cmsconfig` to set the *hostname* entry in `icn0` to the IP address of the split winner's ICN network. In this case, *hostname* is set to `seagoon-i0`

```
Select: icn 0

Item  Name                          Value                  Page 1 of 1
------------------------------------------------------------------------
0     state                         not_present
1     icn_cmd                       disable
2     description                   ICN controller
3     hostname                      seagoon-i0
4     interface_cmd                 up
5     user_label
6     memory_size                   524288
7     descriptor_limit              255
8     info
9     software_fault                no
10    interface_state               unknown
11    busylock                      no
12    action                        disable

(H)elp, (Number)to modify, (S)elect, (T)op or (Q)uit ?
```

Setting the Split Master

The split master negotiates the transfer of ownership of FRUs from one side to another. Only one side of a Netra ft 1800 can be the master, and this can be changed in combined or split mode. In this case, the following command is used to change the master to side A, which also happens to be the split winner:

```
seagoon# splitconf -m A
master = A
```

Note – The master does not have to be the same as the split winner or merge winner. The split or merge winner is the side that continues running during the split or merge operation, respectively, whereas the master negotiates FRU ownership transfers between sides.

Adding Other IP Addresses to /etc/hosts

Here, entries for both seagoon-2 and host names for both side's ICN 0 network instances have been added to the /etc/hosts file:

```
#
# Internet host table
#
127.0.0.1       localhost       loghost
129.156.203.152 seagoon
129.156.203.18  seagoon-2
192.168.1.1     seagoon-i0
192.168.1.2     seagoon-2-i0
```

In addition, localhost has been set as the loghost to ensure that logging is performed on the local machine after a split. Otherwise, logging would be directed across the network to the other side of the Netra ft 1800.

Configuring ft_network 0

Check that ft_network 0 is operating in fault-tolerant mode, that is, both A and B controllers are online.

```
Select: ft_network 0

Item  Name                        Value                  Page 1 of 1
-------------------------------------------------------------------------
0     state                       online_up
1     description                 network multiplexor
2     user_label
3     hostname                    seagoon
4     preferred_controller        none
5     controllerA_FRU             A-CAF
6     controllerA_Funct           Net_0
7     controllerB_FRU             B-CAF
8     controllerB_Funct           Net_0
9     info
10    busylock                    no
11    devpath                     pnet0
12    link                        100 Mbps half-duplex link up
13    transceiver                 internal transceiver selected
14    usable_controllers          A (online) & B (online)
15    controller_in_use           A
```

The split mode software ensures that when the losing (B) side is rebooted after the split, it will (in this example) take on the host name `seagoon-2`.

Checking the SEVM State

Check the state of the SEVM by issuing the following commands:

```
seagoon# ps -ef -oargs | grep vx
vxconfigd -m boot

seagoon# vxdctl mode
mode: enabled

seagoon# vxdisk list
DEVICE       TYPE      DISK         GROUP        STATUS
c2t0d0s2     sliced    rootdisk     rootdg       online
c3t0d0s2     sliced    disk01       rootdg       online
```

The first command ensures that the SEVM configuration daemon is running.

The second command ensures that the daemon in enabled.

The third command shows the status of the HDDs. In this case there are two HDDs under SEVM control, `rootdisk` and `disk01`, which are both members of the rootdg `disk` group. This SEVM HDD configuration is necessary for split mode to work correctly.

In addition, issue the `vxprint` command to determine the redundant state of SEVM as shown in CODE EXAMPLE 3-3:

CODE EXAMPLE 3-3 StorEdge Volume Manager State

```
seagoon# vxprint
Disk group: rootdg

TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg rootdg       rootdg       -        -        -        -        -       -

dm disk01       c3t0d0s2     -        17538444 -        -        -       -
dm rootdisk     c2t0d0s2     -        17538444 -        -        -       -

v  opt          fsgen        ENABLED  4097331  -        ACTIVE   -       -
pl opt-01       opt          ENABLED  4097331  -        ACTIVE   -       -
sd rootdisk-04  opt-01       ENABLED  4097331  0        -        -       -
pl opt-02       opt          ENABLED  4097331  -        ACTIVE   -       -
 sd disk01-01   opt-02       ENABLED  4097331  0        -        -       -
```

CODE EXAMPLE 3-3 StorEdge Volume Manager State *(Continued)*

```
v  rootvol      root         ENABLED  1106028  -        ACTIVE   -       -
pl rootvol-01   rootvol      ENABLED  1106028  -        ACTIVE   -       -
sd rootdisk-B0  rootvol-01   ENABLED  1        0        -        -       Block0
sd rootdisk-02  rootvol-01   ENABLED  1106027  1        -        -       -
pl rootvol-02   rootvol      ENABLED  1106028  -        ACTIVE   -       -
sd disk01-02    rootvol-02   ENABLED  1106028  0        -        -       -

v  swapvol      swap         ENABLED  1052163  -        ACTIVE   -       -
pl swapvol-01   swapvol      ENABLED  1052163  -        ACTIVE   -       -
sd rootdisk-01  swapvol-01   ENABLED  1052163  0        -        -       -
pl swapvol-02   swapvol      ENABLED  1052163  -        ACTIVE   -       -
sd disk01-03    swapvol-02   ENABLED  1052163  0        -        -       -

v  usr          fsgen        ENABLED  7185591  -        ACTIVE   -       -
pl usr-01       usr          ENABLED  7185591  -        ACTIVE   -       -
sd rootdisk-05  usr-01       ENABLED  7185591  0        -        -       -
pl usr-02       usr          ENABLED  7185591  -        ACTIVE   -       -
sd disk01-04    usr-02       ENABLED  7185591  0        -        -       -

v  var          fsgen        ENABLED  4097331  -        ACTIVE   -       -
pl var-01       var          ENABLED  4097331  -        ACTIVE   -       -
sd rootdisk-03  var-01       ENABLED  4097331  0        -        -       -
pl var-02       var          ENABLED  4097331  -        ACTIVE   -       -
sd disk01-05    var-02       ENABLED  4097331  0        -        -       -
```

Notice that there are five volumes in the `rootdg` disk group, each containing two plexes. All plex KSTATE entries should be in the `enabled` state, and all plex STATE entries should be in the `active` state. These states are required in order for split mode to succeed. Recover any plexes that are not in these states so that they are in the `active` state.

Confirming the CPUs Are In Sync

Enter the command `cmsconfig`:

```
ps -elf -oargs | grep u4ftsplitd cmsconfig

Item  Name          Fault Loc     State                  Page 1 of 1
-------------------------------------------------------------------------
0     A-MBD 0             A-MBD   enabled
1     B-MBD 0             B-MBD   enabled
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   enabled
8     HDD 0               A-DSK0  enabled
9     HDD 7               B-DSK0  enabled
10    ft_alarm 0                  usable
11    ft_core 0                   enabled    Fault tolerant
12    ft_network 0                online_up  A (online) & B (online)
13    ft_serial 0                 online
14    icn_system 0                disabled   System is not split

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

Notice that the `ft_core` object reads `Fault tolerant`, indicating that the CPUs are in sync.

Checking the Split Daemon Status

Ensure that the split daemon, `u4ftsplitd`, is running by issuing the command:

```
seagoon# ps -elf -oargs | grep u4ftsplitd
/usr/platform/SUNW,Ultra-4FT/SUNWcms/lib/u4ftsplitd
```

and noting that the output contains u4ftsplitd. If it is not running, restart it with the command:

```
seagoon# /etc/init.d/u4ftsplit start
```

Quick Checklist

At this stage, confirm the following before proceeding:

- The CPUs are running in sync.
- There are HDDs present in `A-DSK 0` and `B-DSK 0`, which are mirrored using SEVM. In addition, these two HDDs are both members of the SEVM disk group `rootdg`. The SEVM configuration contains exactly *one* disk group (rootdg) containing *one* mirrored pair.
- `/etc/splitd.conf` names both `seagoon` and `seagoon-2` and at least one of the ICN networks.
- `/etc/hosts` contains all IP addresses (for `seagoon`, `seagoon-2` and the ICN networks configured in `/etc/splitd.conf`).
- The ft_network 0 is configured in fault-tolerant mode.
- For each ICN network instance configured in `/etc/splitd.conf`, the `hostname` field of the appropriate object in `cmsconfig` should be set. For example, if ICN network instance 0 is configured, the host name for ICN 0 in `cmsconfig` should be set.
- Ensure that the boot disks are mirrored and that the mirroring process has completed. A fully-mirrored volume has an ACTIVE state for the volume and plex fields in the output from the `vxprint` command.

Splitting the System

You are now ready to split the system, but, before proceeding, note the following caution:

Caution – Do not use motherboard resets (`AAres` and `BBres`) or any other mechanism in which the motherboard is powered off, or perform a power cycle or an operation that results in a power cycle, on either side when in split mode. Doing so will result in undesirable failures occurring in certain FRUs on the other side. This is because some resources on a motherboard may belong to the other side and these resources will be removed from service if the motherboard is reset.

If the system is split naturally, that is, if all the resources residing on the side-A motherboard belong to side A and all resources residing on side-B motherboard belong to side B, or if the resources on the motherboard that needs to be reset are not deemed to be critical for providing service to the clients of the other side, it is possible to use a motherboard reset.

However, to avoid failure of the bridges, which will cause subsequent merge operations to fail, in the CMS first disable the motherboard that is to be reset on the side the side that is *not* going to be reset. this will prevent bridge failures and will enable the system to merge successfully when it is brought back into fault-tolerant operation.

Issue the `split` command from either side:

```
splitadm -w a split
```

Note – This will cause many warning messages from SEVM as it loses sight of its mirror.

FIGURE 3-8 shows the output on the A-side console after the `split` command has been issued from the A-side.

```
seagoon# splitadm -w a split
NOTICE: vxdmp: Path failure on 32/124
WARNING: vxvm:vxio: error on Plex var-02 while writing volume var offset 119388
length 2
WARNING: vxvm:vxio: Plex var-02 detached from volume var
WARNING: vxvm:vxio: disk01-05 Subdisk failed in plex var-02 in vol var
vxvm:vxconfigd: NOTICE: Offlining config copy 1 on disk c3t0d0s2:
        Reason: Disk write failure
vxvm:vxconfigd: NOTICE: Detached disk disk01
vxvm:vxconfigd: NOTICE: Detached plex opt-02 in volume opt
vxvm:vxconfigd: NOTICE: Detached plex rootvol-02 in volume rootvol
vxvm:vxconfigd: NOTICE: Detached plex swapvol-02 in volume swapvol
vxvm:vxconfigd: NOTICE: Detached plex usr-02 in volume usr
vxvm:vxconfigd: NOTICE: Detached plex var-02 in volume var
vxvm:vxassist: ERROR: Cannot allocate space to replace subdisks
vxvm:vxassist: ERROR: Cannot allocate space to mirror 1106028 block volume
vxvm:vxassist: ERROR: Cannot allocate space to mirror 1052163 block volume
vxvm:vxassist: ERROR: Cannot allocate space to replace subdisks
vxvm:vxassist: ERROR: Cannot allocate space to replace subdisks
icn network setupWARNING: Out-of-sync on CPUset B

 (via CMS): icn.keepalive done.
domain = A
attributes = split,master
master = A
```

FIGURE 3-8 A-Side Console After `split` Command

The B-side console display is shown in FIGURE 3-9, which is a copy of the A-side console output before the split causes a reboot of the B-side.

```
NOTICE: vxdmp: Path failure on 32/124
WARNING: vxvm:vxio: error on Plex var-02 while writing volume var offset 119388
length 2
WARNING: vxvm:vxio: Plex var-02 detached from volume var
WARNING: vxvm:vxio: disk01-05 Subdisk failed in plex var-02 in vol var
vxvm:vxconfigd: NOTICE: Offlining config copy 1 on disk c3t0d0s2:
        Reason: Disk write failure
vxvm:vxconfigd: NOTICE: Detached disk disk01
vxvm:vxconfigd: NOTICE: Detached plex opt-02 in volume opt
vxvm:vxconfigd: NOTICE: Detached plex rootvol-02 in volume rootvol
vxvm:vxconfigd: NOTICE: Detached plex swapvol-02 in volume swapvol
vxvm:vxconfigd: NOTICE: Detached plex usr-02 in volume usr
vxvm:vxconfigd: NOTICE: Detached plex var-02 in volume var
```

FIGURE 3-9 B-Side Console After `split` Command

Issuing the `cmsconfig` command on `seagoon` should display information similar that in FIGURE 3-10:

```
seagoon# cmsconfig

Item   Name         Fault  Loc     State                      Page 1 of 1
-------------------------------------------------------------------------
0      A-MBD 0             A-MBD   enabled
1      B-MBD 0             B-MBD   enabled
2      CAF 0               A-CAF   enabled
3      CAF 1               B-CAF   disabled
4      CPU 0               A-CPU   enabled
5      CPU 1               B-CPU   disabled
6      DSK 0               A-DSK   enabled
7      DSK 1               B-DSK   disabled
8      HDD 0               A-DSK0  enabled
9      HDD 7               B-DSK0  disabled
10     ft_alarm 0                  usable
11     ft_core 0                   enabled         Running on A-CPU
12     ft_network 0                online_up       A (online)
13     ft_serial 0                 online          2 (unusable)
14     icn 0                       enabled
15     icn 1                       disabled
16     icn 2                       disabled
17     icn 3                       disabled
18     icn_system 0                enabled

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

FIGURE 3-10 CMS Configuration After Split

Bringing Up the Losing Side

The losing (B) side will attempt a reboot from B-DSK0, finally resulting in a prompt to enter SEVM license keys:

CODE EXAMPLE 3-4 B-Side Reboot

```
Sun Ultra 4FT UPA/PCI(2 X UltraSPARC-II 296MHz), No keyboard
OpenBoot 3.7 [PROTO-P1b-sd_st: Fusion-B2], 256 MB memory installed, Serial #9539444.
Ethernet address 8:0:20:91:8f:74, Host ID: 80918f74.

Running preboot tests:                  SUCCESS
Boot device: b-dsk0  File and args:
kadb: kernel/unix
Size: 283753+68208+74596 Bytes
```

CODE EXAMPLE 3-4 B-Side Reboot *(Continued)*

```
/platform/SUNW,Ultra-4FT/kernel/unix loaded 0x9c000 bytes used
SunOS Release 5.6 Version 108145-10 [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1997, Sun Microsystems, Inc.
WARNING: forceload of drv/scsi failed
NOTICE: Ultra-4FT DDI extensions installed
WARNING: forceload of drv/scsi failed
WARNING: forceload of drv/ssd failed
WARNING: forceload of drv/sf failed
WARNING: forceload of drv/pln failed
WARNING: forceload of drv/soc failed
WARNING: forceload of drv/socal failed
}rconsconfig: device at /pci@6,2000/u4ftbus@0/u4ioslot@4/ebus@4/se@14,400000:b is
inaccessible.
rconsconfig: device at /pci@6,2000/u4ftbus@0/u4ioslot@4/ebus@4/se@14,400000:a is
inaccessible.
VxVM starting in boot mode...
vxvm:vxconfigd: WARNING: Detaching plex rootvol-01 from volume rootvol
vxvm:vxconfigd: WARNING: Detaching plex usr-01 from volume usr
vxvm:vxconfigd: WARNING: Disk rootdisk in group rootdg: Disk device not found
vxvm: NOTE: Setting partition /dev/dsk/c3t0d0s1 as the dump device.
VxVM starting special volumes ( swapvol var )...
dumpvp_setup: Setting partition /dev/dsk/c3t0d0s1 as the dump device.
The / file system (/dev/vx/rdsk/rootvol) is being checked.
/dev/vx/rdsk/rootvol: 3023 files, 77365 used, 441552 free
/dev/vx/rdsk/rootvol: (336 frags, 55152 blocks,  0.0% fragmentation)
The /usr file system (/dev/vx/rdsk/usr) is being checked.
/dev/vx/rdsk/usr: 25591 files, 436045 used, 3099937 free
/dev/vx/rdsk/usr: (2785 frags, 387144 blocks,  0.0% fragmentation)
FIRST LOSER BOOT
YOU MUST ENTER THE VERITAS LICENCE KEYS HERE
Please enter your key: 7141 4164 7746 8669 8047 077

vrts:vxserial: INFO: Feature name: CURRSET [95]
vrts:vxserial: INFO: Number of licenses: 1 (non-floating)
vrts:vxserial: INFO: Expiration date: Sun 14 Nov 1999 08:00:00 (33.9 days from now)
vrts:vxserial: INFO: Release Level: 20
vrts:vxserial: INFO: Machine Class: All
vrts:vxserial: INFO: Key successfully installed in /etc/vx/elm/95.
Please enter your key: 8919 6476 4493 0199 2931 280

vrts:vxserial: INFO: Feature name: RAID [96]
vrts:vxserial: INFO: Number of licenses: 1 (non-floating)
vrts:vxserial: INFO: Expiration date: Sun 14 Nov 1999 08:00:00 (33.9 days from now)
vrts:vxserial: INFO: Release Level: 20
vrts:vxserial: INFO: Machine Class: All
vrts:vxserial: INFO: Key successfully installed in /etc/vx/elm/96.
Licensed features:
 Mirroring
 Concatenation
 Disk-spanning
```

CODE EXAMPLE 3-4 B-Side Reboot *(Continued)*

```
 Striping
 RAID-5
*** REBOOT THE SYSTEM ***
syncing file systems... done
rebooting...
Resetting ...
```

This will be followed by a second reboot of side B.

When side B has successfully rebooted, you can log in to both sides. The winning side (A) retains the identity of `seagoon`, and the losing side (B) has the new identity `seagoon-2`.

Check that the system is actually in split mode by issuing the command `splitinfo` from both sides:

```
seagoon# splitinfo -d -a
domain = A
attributes = split,master
```

```
seagoon-2# splitinfo -d -a
domain = B
attributes = split
```

Issuing `cmsconfig` on side B (`seagoon-2`) should now produce the following:

```
seagoon# cmsconfig

Item   Name         Fault  Loc     State                     Page 1 of 1
-------------------------------------------------------------------------
0      A-MBD 0             A-MBD   enabled
1      B-MBD 0             B-MBD   enabled
2      CAF 0               A-CAF   enable_failed   FRU owned by other side
3      CAF 1               B-CAF   enabled
4      CPU 0               A-CPU   enable_failed   FRU owned by other side
5      CPU 1               B-CPU   enabled
6      DSK 0               A-DSK   enable_failed   FRU owned by other side
7      DSK 1               B-DSK   enabled
8      HDD 0               A-DSK0  enable_failed   Cannot be enabled until DSK 0
9      HDD 7               B-DSK0  enabled
10     ft_alarm 0                  usable
11     ft_core 0                   enabled         Running on B-CPU
12     ft_network 0                online_up       B (online)
13     ft_serial 0                 online          1 (unusable) 0 (unusable)
14     icn 0                       enabled
15     icn 1                       disabled
16     icn 2                       disabled
17     icn 3                       disabled
18     icn_system 0                enabled

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

Notice that the `icn_system` is enabled. To ensure that the sides are correctly communicating, issue the following commands on both sides:

```
seagoon# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
     inet 127.0.0.1 netmask ff000000
pnet0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
     inet 129.156.203.152 netmask ffffff00 broadcast 129.156.203.255
     ether 8:0:20:91:8f:54
icn0: flags=8843<UP,BROADCAST,RUNNING,MULTICAST,PRIVATE> mtu 40945
     inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
     ether 8:0:20:91:0:52
```

```
seagoon# u4ftctl -d /dev/icn status
(icn#0): Online + Exporting + Importing
(icn#1): Not initialised
(icn#2): Not initialised
(icn#3): Not initialised
```

```
seagoon-2# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
     inet 127.0.0.1 netmask ff000000
pnet0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
     inet 129.156.203.18 netmask ffffff00 broadcast 129.156.203.255
     ether 8:0:20:91:8f:76
icn0: flags=8843<UP,BROADCAST,RUNNING,MULTICAST,PRIVATE> mtu 40945
     inet 192.168.1.2 netmask ffffff00 broadcast 192.168.1.255
     ether 8:0:20:91:0:74
```

```
seagoon-2# u4ftctl -d /dev/icn status
(icn#0): Online + Exporting + Importing
(icn#1): Not initialised
(icn#2): Not initialised
(icn#3): Not initialised
```

Note that `icn0` should have `inet` set to 192.168.1.1 on seagoon and 192.168.1.2 on `seagoon-2`. The netmasks should both be `ffffff00`, that is, 255.255.255.0.

Note also that the `u4ftctl` command should show any ICN networks in an `Online+Exporting+Importing` state. If they are not, disable the `icn_system` FRUs on both sides and then re-enable them. Check the output of the above commands again. If the ICN network instances are still not in the `Online+Exporting+Importing` state, stop and restart the split daemon by issuing the command:

```
seagoon# /etc/init.d/e4ftsplit stop
```

This will stop and restart the split daemon.

Check that the split daemon is running on each side:

```
seagoon# ps -efo args | grep u4ftsplitd
/usr/platform/SUNW,Ultra-4FT/SUNWcms/lib/u4ftsplitd
```

```
seagoon-2# ps -efo args | grep u4ftsplitd
/usr/platform/SUNW,Ultra-4FT/SUNWcms/lib/u4ftsplitd
```

If the daemon is not running, start it using the `u4ftsplit` command:

```
seagoon# /etc/init.d/u4ftsplit start
```

```
seagoon-2# /etc/init.d/u4ftsplit start
```

Updating `seagoon-2`

You can now make changes to seagoon-2 as required. In this example, a few files are created on seagoon-2 that should survive the merge.

```
seagoon-2# touch /toto /usr/toto /var/toto /opt/toto /etc/toto
```

You can also install a new operating system, new packages and patches, or upgrade an application on the split loser.

Merging the Systems

The steps required for merging the two systems are as follows:

1. Check the split daemons on both sides are communicating with each other.
2. Issue the `merge` command.
3. Bring the losing side FRUs back online.
4. Wait for the CPUsets to recombine.
5. Remirror the new boot disk.

Confirming the Split Daemons are Communicating

Before a merge can occur, the split daemons on the two sides of the machine must be communicating with each other over the chosen network (set up in `/etc/splitd.conf`), which is typically ICN. Confirm this by attempting to change split mastership on both sides:

```
seagoon# splitconf -m <side>
master = A
```

<*side*> is dependent on the existing master: if the existing master is A, <*side*> should be B; if the existing master is B, <*side*> should be A.

- If the change of mastership proceeds without producing an error and the `splitinfo` command shows <*side*> as the master on both sides, the split daemons are communicating.
- If this is not the case, stop the split daemon on each side using the command **`/etc/init.d/u4ftsplit stop`**, which should result in the split daemons stopping and then restarting. If necessary, recheck split daemon communication using the above procedure.

Issuing the `merge` Command

You can issue the `merge` command from either side. In this case, it is issued from the merge winner:

```
seagoon-2# splitadm -w B merge
icn network shutdown (via CMS): icn.keepalive done.
domain = C
attributes = combined
master = A
```

This will force the losing side (A) to reboot. and the side A console displays the following:

```
icn network shutdown (via CMS): icn.keepalive 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
Killed download daemon.
Oct 11 11:26:33 snmpdx: received signal 15
The system is down.

INIT: failed write of utmpx entry:"s6"

INIT: failed write of utmpx entry:"rb"
syncing file systems... done
rebooting...
Resetting ...
```

At this point the side A console may go dead because the newly-combined system has the A-CAF disabled.

Waiting for the CPUsets to Become Fault Tolerant

Check by running `cmsconfig` and noting that *ft_core* has the value `Fault tolerant`. This step is optional; other tasks can be executed in parallel.

Bringing the Losing Side FRUs Back Online

Issue the `cmsconfig` command on the merge winner (`seagoon-2`, side B):

```
seagoon# cmsconfig

Item  Name         Fault  Loc     State                      Page 1 of 1
-------------------------------------------------------------------------
0     A-MBD 0             A-MBD   enabled
1     B-MBD 0             B-MBD   enabled
2     CAF 0               A-CAF   disabled        WARNING: Power off command
3     CAF 1               B-CAF   enabled
4     CPU 0               A-CPU   busy
5     CPU 1               B-CPU   enabled
6     DSK 0               A-DSK   disabled        WARNING: Power off command
7     DSK 1               B-DSK   enabled
8     HDD 0               A-DSK0  disabled        WARNING: Power off command
9     HDD 7               B-DSK0  enabled
10    ft_alarm 0                  usable
11    ft_core 0                   enabled         Waiting for A-CPU to be ready
12    ft_network 0                online_up       B (online)
13    ft_serial 0                 online          0 (unusable)
14    icn_system 0                disabled

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

Remirroring the New Boot Disk

After the merge, you can check the state of the SEVM using the `vxdisk list` command, as follows:

```
seagoon-2# vxdisk list
 DEVICE       TYPE      DISK         GROUP        STATUS
 c2t0d0s2     sliced    -            -            online
 c3t0d0s2     sliced    disk01       rootdg       online
 -            -         rootdisk     rootdg       failed was:c2t0d0s2
```

The boot disk used by `seagoon-2`, containing the updated operating system, is `c3t0d0`, known to SEVM as `disk01`. The original disk used by `seagoon` before the upgrade is `c2t0d0`, known to SEVM as `rootdisk`, which has been failed by SEVM.

The merge process results in SEVM losing sight of `rootdisk` because the CMS tells it that it is disabled (which occurs because the merged system inherits its CMS state from the merge winner in which `rootdisk` was disabled).

The procedure now is to inform SEVM that `rootdisk` is still present in the Netra ft 1800. To do this, use the `vxdiskadm` command, main menu option 5 (Replace a failed or removed disk). The sequence of screens is as follows.

```
seagoon-2# vxdiskadm

Volume Manager Support Operations
Menu: VolumeManager/Disk

 1      Add or initialize one or more disks
 2      Encapsulate one or more disks
 3      Remove a disk
 4      Remove a disk for replacement
 5      Replace a failed or removed disk
 6      Mirror volumes on a disk
 7      Move volumes from a disk
 8      Enable access to (import) a disk group
 9      Remove access to (deport) a disk group
 10     Enable (online) a disk device
 11     Disable (offline) a disk device
 12     Mark a disk as a spare for a disk group
 13     Turn off the spare flag on a disk
 list   List disk information

 ?      Display help about menu
 ??     Display help about the menuing system
 q      Exit from menus

Select an operation to perform: 5
```

```
Replace a failed or removed disk
Menu: VolumeManager/Disk/ReplaceDisk

 Use this menu operation to specify a replacement disk for a disk
 that you removed with the "Remove a disk for replacement" menu
 operation, or that failed during use.  You will be prompted for
 a disk name to replace and a disk device to use as a replacement.
 You can choose an uninitialized disk, in which case the disk will
 be initialized, or you can choose a disk that you have already
 initialized using the Add or initialize a disk menu operation.

Select a removed or failed disk [<disk>,list,q,?] list
```

```
Disk group: rootdg

DM NAME       DEVICE       TYPE     PRIVLEN  PUBLEN   STATE

dm rootdisk   -            -        -        -        NODEVICE

Select a removed or failed disk [<disk>,list,q,?] rootdisk

Select disk device to initialize [<address>,list,q,?] list
```

```
DEVICE       DISK         GROUP        STATUS
c2t0d0       -            -            online
c3t0d0       disk01       rootdg       online

Select disk device to initialize [<address>,list,q,?] c2t0d0
```

```
This disk device is currently listed as in use by another host.
If you are certain that the other host is not using the disk, you
can choose to clear the use status. To use the disk the use status
must be cleared.
Output format: [Device_Name,Disk_Access_Name,Hostid]

[c2t0d0,c2t0d0s2,seagoon]

Clear use status? [y,n,q,?] (default: n) y
```

```
The following disk you selected for use appears to already have
been initialized for the Volume Manager.  If you are certain the
disk has already been initialized for the Volume Manager, then you
do not need to reinitialize the disk device.
Output format: [Device_Name]

c2t0d0

Reinitialize this device? [y,n,q,?] (default: y) y
```

```
Replacement of disk rootdisk in group rootdg with disk device
c2t0d0 completed successfully.

Replace another disk? [y,n,q,?] (default: n) n
```

Check the state of SEVM using the `vxdisk` list command:

```
seagoon-2# vxdisk list
DEVICE       TYPE      DISK         GROUP        STATUS
c2t0d0s2     sliced    rootdisk     rootdg       online
c3t0d0s2     sliced    disk01       rootdg       online
```

Note that there are two devices, both of which are online and mapped to two SEVM disks. At this point, SEVM knows about `c2t0d0` and has assigned `rootdisk` to it, as before. The mirroring from `disk01` to `rootdisk` can now proceed.

Check for the existence of a process `vxrecover`, which resynchronizes the boot disk mirror:

```
# rip# ps -ef | grep recover
root  9764  4250  0 17:13:10 console  0:00 grep recover
root  9173     1  0 17:12:00 ?        0:00 vxrecover -sb -g rootdg rootdisk
```

If it doesn't exist, execute it by issuing the command `vxrecover`. The progress of this remirroring can be followed using either `vxprint`, or the X-windows graphical interface, `vxva`.

If SEVM is unable to find the disk device `c2t0d0` for `rootdisk`, add the disk and initialize it before you select option 4.

This is achieved by selecting option 1 from the main menu of `vxdiskadm`.

Recovery Procedures

This section describes some recovery procedures specifically for use when operating in split mode.

Power-Cycling a Domain When in Split Mode

In split mode, each side has access to both the local motherboard and the opposite side's motherboard. By default, on each side `cmsconfig` shows that the state of A-MBD and B-MBD is `enabled`. If one side needs to be power-cycled, via an RCP command or equivalent operation, the other side is likely to be affected in the same

way as if it were in combined mode. Normally, the result is a failure of the power-cycled motherboard. Depending on the machine configuration, this can cause data corruption and/or loss of service.

Therefore, on a split system, power-cycle operations should be treated as in combined mode; that is, before performing the power cycle, the relevant motherboard must be disabled.

For example, suppose that you want to power-cycle side B. Before doing this, use `cmsconfig` on side A to check the status of B-MBD. If B-MBD is enabled, you must disable it.

Note – To disable a motherboard, you must first disable all its dependent FRUs (not the PSUs). `icn_system` is a dependent of both motherboards and should also be disabled.

Once B-MBD is disabled, you can proceed with the power-cycle operation on side B, and side A will not be affected. When side B is powered on again and has completed its reset sequence, B-MBD can be enabled again on side A. When B-MBD is successfully enabled, all its dependent FRUs owned by side A can be enabled. `icn_system` should also be enabled.

Fixing a Motherboard Failed When in Split Mode

On a split side, the other side's motherboard may be reported as `failed` if, for example, the other side is power-cycled without following the procedure described above.

Note – Even if no FRUs are owned or used on the opposite side, the failure of the motherboard must be corrected to achieve a successful merge operation.

For example, if side A reports B-MBD as `failed`, the following procedure should be followed:

Use `cmsconfig` on side A to disable B-MBD.

Note – To disable a motherboard, you must first disable all its dependent FRUs (not the PSUs). `icn_system` is a dependent of both motherboards and should be disabled too.

Once B-MBD is disabled, it can be enabled again. When B-MBD is successfully enabled, all its dependent FRUs owned by side A can be enabled. `icn_system` should also be enabled.

Fixing a Motherboard Failed as a Result of a Merge Operation

If a motherboard is reported as `failed` soon after the completion of a merge operation, it is likely that the reported failure is not a genuine hardware fault but a spurious error caused by a software malfunction. If this happens, the failed motherboard should not be replaced. Instead, the following procedure should be applied

Use `cmsconfig` to disable the failed motherboard.

Note – To disable a motherboard, you must first disable all its dependent FRUs (not the PSUs).

Once the motherboard is disabled, use `cmsrepairfru` to delete the failure record on its EEPROMs. For example, if you are fixing A-MBD, type:

```
# cmsrepairfru -l A-MBD
```

Once repaired, use `cmsconfig` to enable the motherboard, and then all the FRUs dependent on it.

CHAPTER 4

OSdog

This chapter describes the functionality and use of the OSdog facility.

The OSdog is the last line of defense that the system provides against a hung machine, whether caused by a software or hardware failure. Only a subset of total failures result in a hang (where the machine is not scheduling threads sufficiently quickly); some can result in a crash from which the machine will recover with a reboot.

The OSdog helps to maintain availability by providing a limit to the recovery time for such cases. The OSdog does not attempt to catch all system failures. It assumes that your application is capable of determining if there is sufficient I/O throughput, memory, and computing resources for its needs. The availability of thread scheduling enables you to monitor these and take appropriate action when required.

The chapter contains the following sections:

Operation

Hardware

The OSdog protects the system from hangs that can be caused by either hardware or software failure. Each system motherboard contains a hardware timer that increments until either:

- The timer is reset to zero (the OSdog is *patted*), or
- The timer reaches an expiry value, at which point an OSdog reset is issued

Under normal operating conditions, the software regularly pats the OSdog timer, indicating that the system is operating correctly. If the system hangs, the pats do not occur, and the timer causes an OSdog reset event, which allows the system to reboot and recover normal operation.

The effect of an OSdog reset is dependent upon the system mode of operation (fault-tolerant mode or split mode). In fault-tolerant mode, an OSdog reset event causes the appropriate side of the system—CPUset, CAF, disks and PCI cards—to be power cycled. In split mode, only the appropriate side's CPUset is reset due to the potential of resource sharing across sides.

There are two OSdog devices (one per motherboard), and these are associated with that motherboard's side of the system. Each OSdog must be patted individually (this is transparent to the user) to stop an OSdog reset occurring. The OSdog is only enabled on CPUsets that are performing useful work. In particular, the OSdog is not enabled on a secondary CPUset (in fault-tolerant mode) because that CPUset is performing system monitoring operations and it is not appropriate to enable the OSdog at that point. When the secondary CPUset is brought into sync, the OSdog will be enabled for that side.

Solaris Software

The Solaris operating environment provides a multiple stage OSdog operation that simplifies the diagnosis of hangs. In addition to the hardware OSdog protection, if the software detects a system hang that would cause the hardware OSdog to trigger, it panics the system to cause a reboot and to produce a core dump. This makes it possible to diagnose a large number of system hangs.

The principle of operation behind this feature is similar to the Solaris kernel deadman.

Design Principles

The OSdog feature has been developed with the following in mind:

- The OSdog triggers when the system is hung.
- The OSdog does not trigger when the system is not hung.
- All OSdog events are reported.
- The OSdog only runs on a side of the system whose CPUset is doing useful work.
- OSdog protects as much code as possible.
- A latent fault checker performs periodic checks for possible OSdog faults.
- Wherever possible, diagnostic information is saved for fault analysis.

Configuration

The OSdog does not have to be used to protect a system. You can alter the configuration of each motherboard's OSdog, either to enable or disable it. This allows for situations where the expected system operation exceeds the bounds set for OSdog operation (for instance, debugging).

The configuration information is stored in the motherboard configuration EEPROMs at locations `EE_MBD_OSDOG_A` and `EE_MBD_OSDOG_B`. Note that both locations should contain the same value. This information is read during system initialization and is used to determine the initial state of the OSdog. After this time, the OpenBoot PROM caches the information in the NVRAM device on the CPUset and makes it available as variables *osdog-a* and *osdog-b*. It also makes it available as properties *osdog-a* and *osdog-b* in the `/u4ft-options` device node.

Note – Both motherboard EEPROMs should contain the same data. An update of the OSdog configuration involves changing the EEPROMs on both motherboards.

The only supported method of updating the OSdog configuration is using `set-conf-osdog` in the OpenBoot PROM. This command must be used immediately after a system reset (and must be followed by a system reset).

To enable the OSdog, type:

```
ok setenv auto-boot? false
ok reset-all
ok h# 4f set-conf-osdog
ok reset-all
ok setenv auto-boot? true
```

To disable it, type:

```
ok setenv auto-boot? false
ok reset-all
ok h# 0 set-conf-osdog
ok reset-all
ok setenv auto-boot? true
```

OSdog Software

u4ftdog - OSdog Driver

The u4ftdog driver is a multi-threaded, multi-instance non-STREAMS driver that is responsible for controlling the operating system watchdog. It exports an ioctl(2) interface that enables applications to register that they can report system availability. Once registered, the application is responsible for periodically reporting (via the ioctl(2) interface) that the system is available. Failure to do this is classed as a system failure and the u4ftdog driver then initiates a system reboot using the cmn_err (9F) CE_PANIC function.

The u4ftdog driver also enables control of the Netra ft 1800 OSdog hardware, which provides a hard failure recovery mechanism. Failure to regularly pat the OSdog hardware causes a system power cycle/reset to be initiated.

Hardware Interface

The Netra ft 1800 hardware OSdog functionality is provided by a number of registers on each motherboard. These registers implement a timer that counts upwards. In normal system operation, this counter is reset periodically by the

u4ftdog driver software writing to the pat register. If the register is not patted within a specified time limit, the counter reaches a trigger value, which causes the motherboard hardware to initiate an OSdog timeout.

The actions associated with an OSdog timeout are dependent upon the system's operating mode: in a combined system, the OSdog timeout causes a system power cycle of the side whose OSdog timed out; in split mode, the OSdog timeout causes a CPUset reset on the OSdog's side. The differences in actions are due to the possibility of resource sharing in a split system where one side may be using resources on the other side. If the side is power cycled, the other side's resources are reset.

The OSdog hardware has two possible timeout values:

- Normal (10.74 seconds), which is generally selected when the Solaris operating environment is running
- Extended (85.90 seconds), which is generally selected when system firmware is running

Driver Summary

The u4ftdog driver comprises three instances—a control instance and two hardware instances.

- The control instance represents the upper layer of the driver. It handles the user patting ioctl interface and performs the patting of the hardware OSdogs.
- The hardware instances represent the OSdog hardware on each motherboard and enable the hardware to be brought online or taken offline in response to CPUset integration, out-of-sync events, and hot swapping of the motherboards. The hardware instances have private interfaces, which are not intended for use by end users.

The driver performs continuous latent fault checking on the OSdog hardware once the OSdog has been enabled. The driver checks monitor bits in the hardware registers and ensures that they are pulsing correctly.

Configuration

The u4ftdog driver provides a hardware configuration file, u4ftdog.conf, which contains various properties that are used during driver initialization. You should change only the channel_x_limit properties; none of the other properties is user-configurable. The properties recognized by the u4ftdog driver are:

- name
  A string property that identifies the driver. It is the same for all instances.

- `parent`
  A string that identifies the parent of a particular instance. This property encodes the absolute path of the parent node. For the control instance, this property is denoted as *pseudo* because it is not responsible for driving any hardware.
- `node`
  A single character string property that defines the instance of the current node. `A` and `B` represent the OSdog hardware on the respective sides, and `C` represents the control instance.
- `reg`
  A 5-tuple integer property that describes the address space that will be mapped by the hardware instances.

The following configuration file properties are common to all three `u4ftdog` instances:

- `channel_`*x*`_limit`
  A number of integer properties, one per channel (replace *x* with the channel number), that describe initial limits (in seconds) for each pat channel. This is the interval between initialization of the kernel and triggering of the OSdog. These properties are used to provide OSdog protection during the initial stages of the boot. They should be set to a value that represents the time that you want the system to wait before concluding that a particular channel patter has failed to start. If this limit is reached, the software OSdog triggers. You can update this property to reflect individual requirements.
- `no-config`
  An integer property that tells the framework that the hardware represented is non self-identifying and will provide a configuration file
- `non-prom`
  An integer property that tells the framework that the OpenBoot PROM does not recognize the device; that is, it is not a hardware node
- `u4ft-aware`
  An integer property that tells the framework that the instances will provide their own routines to process the `online`/`offline` state change requests
- `trace`
  A string property that encodes the trace flags that cause certain trace messages to be output to the message log

`ioctls`

The `u4ftdog` driver provides `ioctls` that enable applications to configure and perform user OSdog patting, and enable system management functions to configure the OSdog.

The following system calls are not supported by the `u4ftdog` driver:

- `read`(2)
- `write`(2)
- `mmap`(2)

During initialization, the control instance creates a number of device nodes (one per pat channel):

`/dev/u4ftdog:pat`*x*

where *x* is the pat channel number from 1 to 8.

In addition, each instance creates a device node `/dev/u4ftdog:[ABC]`, which is used by system control functions. These devices should not be used.

User Pat Channel `ioctls`

The user pat channels enable you to define a custom system hang detection scheme that monitors aspects of the system appropriate to your needs (see `u4ftdogpat`(1M) for an example).

By default, the user pat channels are active only when a process has the channel device special file open and has configured the limit of the channel. This provides a safety net against problems with the monitoring application that cause it to terminate abnormally and thus cause the OSdog to trigger. This behavior can be overridden using the `LINGER_ON_CLOSE ioctl` described below.

A typical use of a user pat channel is as follows:

- Open the channel device; set the timeout limit to an appropriate value
- Loop around, monitoring for hangs, and patting the OSdog if the system is not hung: you must pat more frequently than the limit period
- If you want to close the channel, first set the limit to zero (disable the channel) then close the channel device

The following `ioctls` are applicable to the pat channel devices.

`U4FTIOC_OSDOG_SET_LIMIT`

This enables or disables a user patting channel. The argument is a pointer to `uint32_t`, which represents the desired limit (in seconds) for that pat channel. A pat channel is enabled when a non-zero limit is specified and disabled when a zero limit is specified.

Note – Care should be taken when selecting a limit to ensure that it is appropriate for the expected pat process. In particular, small values (less than 10) are generally inappropriate.

U4FTIOC_OSDOG_PAT

Pat the channel, thus informing the `u4ftdog` driver that the patting process is active and that the driver should reset the pat channel's OSdog timer. The `ioctl` takes no argument (supply a NULL pointer).

The patting must be performed often enough to ensure that the OSdog does not trigger. Take care to ensure that any scheduling delays (if permissible) are accounted for.

U4FTIOC_OSDOG_LINGER_ON_CLOSE

Specify that the driver should not automatically disable the pat channel if the channel is closed. This is provided to enable system shutdown operations to be OSdog protected.

The `ioctl` takes a `uint32_t` parameter whose value is 1 to enable it and 0 to disable it.

Note – If a device is closed and then re-opened, the `ioctl` will be disabled.

Diagnosing OSdog Timeouts

OSdog timeouts are generated in response to hangs detected by either the OSdog hardware or software. These timeouts cause the system to take corrective action by rebooting. The behavior of each type of hang is described below.

Hardware Event

A hardware OSdog timeout is indicative of a number of faults including:

- Hardware connectivity failure between the CPUset and motherboard
- Motherboard failure
- Gross software failure, that is, the system is hung with interrupts locked out

There is independent OSdog hardware on each side of a system. It is responsible for power cycling and resetting its own side if the OSdog is not patted for any reason. This structure means that, in some circumstances, the OSdog can trigger on only one side of a system, which often indicates a hardware or connectivity problem on that side. There is a small time window during CPUset re-integration operations when a hang would cause the OSdog to fire on only one side of the system. This is due to the software conventions used by the CMS. In these circumstances, the system recovers on one side and the CMS initiates recovery on the second side.

Hardware events are reported by the system in the following locations:

- Message on the console during initialization of theOpenBoot PROM

  This reports that the reset was due to an OSdog event:

```
Last reset was caused by OSdog expiry
```

- The value of *reset-reason* in the OpenBoot PROM device node

  If the reset was due to OSdog, the property has the value `OSDOG`. To examine this device node, type the following:

```
{0} ok cd /
{0} ok .properties
initial-rcp-status   13 46 c6 e2 26 da b7 1b bc e8 12 ad 66 16 b3 15
reset-reason         OSDOG
energystar-v2
...
{0} ok
```

The OpenBoot PROM reports *reset-reason* has the value OSDOG.

- A message is placed in the status log during boot, as follows:

```
[380c3a04.123bdd30]     M       0       <u4ftcmd#0>     ()
    OBP reports reset-reason as “OSDOG”
```

Software Event

A software OSdog timeout occurs as a result of an event that causes the system to hang. Specific causes include:

- High level interrupt failure, which causes level 14 soft interrupts to be blocked

  This failure actually results in a hardware OSdog event because the OSdog software cannot monitor the system and pat the hardware OSdog. The type of failure occurs only when interrupts on all processors are blocked.

- High level interrupts locked out (levels 11 through 13)

  This stops the OSdog from patting the hardware, and the deadman (level 14) interrupts see that patting has stopped and cause the system to panic with the following message:

```
panic[cpu0]/thread=0x3002be80: OSdog patting is too infrequent or
stopped (OSdog hard hang)
stopped at:
Syslimit+0x94:  ta      %icc,%g0 + 125
kadb[0]:
```

  A typical cause of this type of hang is a driver high level interrupt routine that becomes stuck in a non-terminating loop; for example, waiting for a mutex that is never released by the current owner.

  It can also occur as a result of a check for a hardware condition that never occurs. That is, the code can look for the completion of a command in a device register, but the register is broken and is returning false values.

- Clock thread stopped (level 10 interrupts not being delivered)

The OSdog monitors the *lbolt* kernel variable, which is incremented once every ten milliseconds. If this is not incremented for 20 seconds (configurable), the OSdog reports a hang:

```
panic[cpu0]/thread=0x3002be80: OSdog detected system hang, pat
channel=0, limit=20
stopped at      Syslimit+0x94:  ta      %icc,%g0 + 125
kadb[0]:
```

Note – This type of hang is always reported as `pat channel 0` (zero).

This type of hang is typically caused by scheduler problems, including excessive scheduling load.

- User pats not delivered within the specified limit time

  If a user pat channel is enabled and the appropriate pat daemon has not performed a pat ioctl within the specified time limit, the system panics and reports an OSdog hang. This produces a message similar to that in previous case, but the pat channel will be the one that has not been patted.

```
panic[cpu0]/thread=0x3002be80: OSdog detected system hang, pat
channel=1, limit=10
stopped at Syslimit+0x94:  ta      %icc,%g0 + 125
kadb[0]:
```

Causes of this type of hang are user defined, although it can also result from an excessive boot time, where the pat counter reaches the initial timeout limit before the user patting daemon has started.

Similarly, a timeout can be caused by a shutdown that hangs or takes too long (for example, as a result of NFS unmounting problems).

Software events are reported in a number of ways:

- The most immediate report is the system panic message that appears on the system console. This reports one of the messages (shown above), which indicates the possible causes of a hang.
- The same message is also reported in the CPUset NVRAM log. This is obtained from the CPUset during reboot and stored in the NVRAM log file.

```
[380c490d.51c6f43b]  F 0 <u4ftdog#4> ()
  OSdog detected system hang, pat channel=1, limit=10
```

After a software OSdog timeout, the OpenBoot PROM does not report that the reset was due to an OSdog event. This is because the OpenBoot PROM reports the state of the hardware and sees the software OSdog timeout as a soft reset.

Debugging the Operating System

The OSdog can cause difficulties when you debug the operating system. In particular, breakpoints set in either `kadb` or OpenBoot PROM bypass normal methods of entry into the debugger and cause the hardware OSdog to trigger. To prevent this from happening and facilitate debugging, `u4ftdog` reduces the level of system protection when a debugger is being used. This means that, in this case, the OSdog is always patted when entering `kadb` or OpenBoot PROM.

The OSdog driver detects that a debugger is loaded by the presence of either `kadb` or the `misc/obpsym` kernel module. Under all other circumstances, the OSdog is patted only in OpenBoot PROM, if you enter this using a console hardware or software break sequence, and when the Solaris operating environment is shutdown to run level 0.

Note – It is essential, therefore, that you do not run with a debugger enabled unless absolutely necessary for debugging purposes.

Handling Panics

Operating system panics need special treatment by the OSdog software to allow time for the file systems to sync and for panic dumps to be made. This is essential to minimize file system check time and to aid fault diagnosis.

When a panic is initiated, the `u4ftdog` driver switches into panic mode, where it:

- Sets the OSdog hardware to the extended timeout setting
- Relaxes the usual checks for system operation

The latter is achieved by placing a time limit on panic processing. If this time limit is exceeded, the driver invokes a secondary OSdog panic with the message

```
panic[cpu1]/thread=0x64587b80: panic OSdog timeout
```

and then no longer pats the OSdog hardware. This combination of activities provides another opportunity for the panic code to continue, but limits any remaining time to the hardware OSdog timeout period (that is, 85.90 seconds). If the panic has not initiated a system reboot within this time, the hardware OSdog triggers, causing a power cycle and a reboot.

Note – In certain circumstances, particularly when the panic is originated by the interrupt subsystem, it may not be possible to pat the OSdog at all during the panic. In these cases, the OSdog hardware resets the machine in 85.90 seconds unless the panic succeeds within that time. Normally, the panic processing would not succeed.

You can tune the timeout period for panic handling by setting the `u4ftdog` *max_panic_time* variable in `/etc/system`, that is:

```
set u4ftdog:max_panic_time=300
```

The variables value is the number of seconds the system waits before declaring that the panic has failed to complete. The default time for this is 160 seconds.

Panic and the Debugger

A panic provides an opportunity to debug the kernel and determine the cause of the hang. You can perform this either immediately, if the kernel debugger (`kadb`) is loaded, or later using the core file, which is saved as part of the panic.

Note – Check that `savecore` is enabled (see `/etc/rc2.d/S20sysetup`).

In production systems, you may not want the system to enter the kernel debugger. In this case, set the kernel variable *nopanicdebug* to 1 by placing an entry in `/etc/system` as follows:

```
set nopanicdebug = 1
```

Note – The system must be rebooted for this setting to take effect.

User Patting Description and Example

A sample user patting configuration is provided on the system by default. You can tune this configuration to provide cover in a way that is appropriate to your needs. The patting is provided by the `u4ftdogpat` program, which pats the OSdog at a specified time interval. This enables you to monitor thread scheduling for specific user priorities.

The configuration is split into a number of areas:

- **Initial boot** When the u4ftdog driver comes online, a default OSdog timeout period is initialized and the pat channel is enabled. The `/etc/init.d/u4ftdog_boot` script is then responsible for starting an initial instance of the patting program, which starts patting the OSdog. This guards against hangs in kernel initialization that stop the boot from getting underway.
- **Normal running** Normal running mode is entered once Solaris has checked disks and mounted the `/usr` partition. In this mode, the OSdog is reconfigured to provide appropriate coverage for the system and `cmsmonitord` is used to monitor the `u4ftdogpat` program and ensure that it is always running. If the daemon dies for any reason, it will be restarted by `cmsmonitord`.

  Entry into normal running mode is controlled by the `/etc/init.d/u4ftdog` script. This responds to parameters *start* and *restart* (and also *stop*, but see the following bullet point). The *start* version is executed during the latter stages of `/etc/rcS.d/` scripts and this starts the dynamically-linked version of `u4ftdogpat` with the appropriate configuration parameter, and then kills the statically-linked version.

  restart is used by `cmsmonitord` if it detects that `u4ftdogpat` has died. In this case, there is no attempt to kill off the statically-linked version.
- **Shutdown** Shutdown may involve delays whilst file systems are being unmounted, for example, syncing file systems. This can take longer than the standard timeout period for user patting. This is compounded by the fact that processes are also killed during shutdown causing patting to stop. To protect against hangs at this stage, the pat channel is reconfigured with a longer timeout period and with the `LINGER_ON_CLOSE` option set to ensure that the channel remains active once the pat process is killed.

The default pat configuration uses pat channel 1. If you want to provide monitoring support for your application, use a different channel and leave the existing monitoring in place.

Developing a Custom User Pat Daemon

The OSdog user patting API has been developed to enable you to specify a criteria for satisfactory system operation and to develop a monitor daemon that checks that this criteria is being met. A suggested algorithm for doing this is as follows:

CODE EXAMPLE 4-1 User Pat Daemon

```
#include <sys/types.h>
#include <fcntl.h>
#include <sys/u4ftdog_io.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
    int fd;
    int period = 5;       /* Interval between checks on system */
    uint32_t limit = 40;  /* Channel timeout value */
    int shutdown = 0;     /* do shutdown? - maybe set by a signal */
    int system_ok = 1;    /* Set by status monitor function */

    /* initialise system monitor application (insert your code) */

    /* Open pat channel */
    fd = open("/dev/u4ftdog:pat2", O_RDWR);

    /* Configure appropriate limit */
    ioctl(fd, U4FTIOC_OSDOG_SET_LIMIT, &limit);

    /* Perform loop until shutdown is requested */
    /* A signal handler may be used to do this */
    while (!shutdown) {

        /* check system/application status (insert your code) */

        /* system_ok = sys_status(); */
        if (system_ok) {
        ioctl(fd, U4FTIOC_OSDOG_PAT, NULL);
        } else {
            printf("System needs attention\n");

            /* take recovery action (insert your code) */

            /* possibly pat the osdog during recovery period */
            /* (insert your code) */
```

CODE EXAMPLE 4-1 User Pat Daemon *(Continued)*

```
        }
        poll(NULL, 0, (period * 1000));
    }
    /* Orderly close requested */
    /* Disable osdog */
    limit = 0;            /* Limit of zero means disable */
    ioctl(fd, U4FTIOC_OSDOG_SET_LIMIT, &limit);

    /* Close pat channel */
    close(fd);

    /* stop system monitoring (insert your code) */
    exit(0);
}
```

Note – Insert your own tested and proven program at the points indicated in the algorithm.

Configuring the Default User Patting

Adjust the configuration for the default user patting by modifying file settings as follows:

- Initial timeout between OSdog coming online and the user pat daemon starting

  In the file `/platform/SUNW,Ultra-4FT/kernel/drv/u4ftdog.conf`, change the setting of *channel_x_limit* to a different value. The value specified is the initial timeout limit in seconds.

Note – Replace *x* with the number of the channel which is to be changed.

- Boot timeout

In the file `/etc/init.d/u4ftdog_boot`, change the *-l* parameter for `u4ftdogpat` to the desired timeout limit.

- Normal running timeout

  In the file `/etc/init.d/u4ftdog`, change the *-l* parameter for `u4ftdogpat` to the desired timeout limit.

Note – Change only the parameter for the `u4ftdogpat` invocation that is part of the *start* and *restart* option.

- Shutdown timeout

In the file `/etc/init.d/u4ftdog`, change the *-l* parameter for `u4ftdogpat` to the desired timeout limit.

Note – Change only the parameter for the `u4ftdogpat` invocation that is part of the *stop* option.

Adjust other parameters as specified in the `u4ftdogpat` manual page.

Obtaining OSdog Settings Using `cmsfruinfo`

You can obtain the current OSdog settings stored in the motherboard using the `cmsfruinfo` command.

- When the OSdog is disabled:

```
# cmsfruinfo -1 A-MBD EE_MBD_OSDOG
EE_MBD_OSDOG=
    EE_MBD_OSDOG_0=0
    EE_MBD_OSDOG_1=0
```

```
# cmsfruinfo -1 B-MBD EE_MBD_OSDOG
EE_MBD_OSDOG=
    EE_MBD_OSDOG_0=0
    EE_MBD_OSDOG_1=0
```

- When the OSdog is enabled:

```
# cmsfruinfo -1 A-MBD EE_MBD_OSDOG
EE_MBD_OSDOG=
    EE_MBD_OSDOG_0=79
    EE_MBD_OSDOG_1=79
```

```
# cmsfruinfo -1 B-MBD EE_MBD_OSDOG
EE_MBD_OSDOG=
    EE_MBD_OSDOG_0=79
    EE_MBD_OSDOG_1=79
```

System Clock Thread Monitoring

You can adjust the timeout for system clock thread monitoring by setting the variable *u4ftdog_lbolt_timeout* in `/etc/system`.

For example:

```
set u4ftdog:u4ftdog_lbolt_timeout=40
```

The value is the number of seconds to wait before causing the OSdog to trigger. Reboot the system for this change to take effect.

Note – This timeout limit should be smaller than the timeout limits used for user pat monitoring because the user processes are generally dependent upon the clock for process wake up.

Troubleshooting

- The initial OSdog timeout period is too short and you get an OSdog timeout before you can get to a prompt to correct matters.

  Boot to single-user mode (using the *-s* option to boot). When a single-user boot is used, the initial timeout period is not set. Once you are at the single-user mode prompt, you can alter the settings appropriately.

- Problems with initial boot

  Use `boot -s`

CHAPTER 5

Motherboard Replacement

This chapter describes the procedure for hot swapping the Netra ft 1800 motherboards. You may also wish to refer to the *Netra ft 1800 User's Guide*. The main sections of the chapter are:

The upper motherboard (A-MBD) and the lower motherboard (B-MBD) are removed and replaced in almost exactly the same way.

Note – The procedure described below assumes that you have installed the Update 01 software. Do not use this procedure with any other level of software.

Caution – Motherboard replacement must be carried out as a hot swap to ensure that all system identity (base Ethernet address and host id) is copied to the new motherboard module's EEPROM. Failure to do this will result in a system motherboard without a unique host id and a base Ethernet address of zero.

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.

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 5-1).

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.

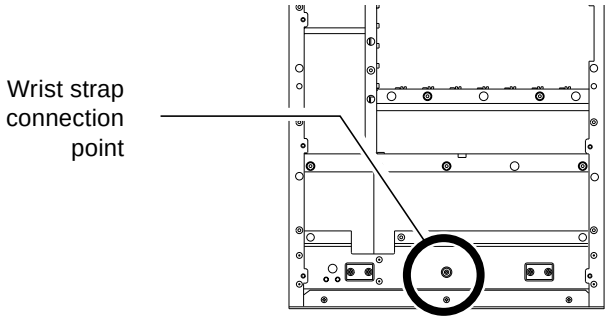

FIGURE 5-1 Wrist Strap Connection Point

Removing an Existing Motherboard

Refer to this section for details of how to disable modules connected to the motherboard due for replacement, and how to remove the motherboard from the chassis.

▼ To Disable Connected Modules

1. **Log on as root using the console on the side of the system that is to remain running.**

   You can also `rlogin` as root, assuming the `CONSOLE` line in `/etc/default/login` is commented out.

2. **Perform the following actions prior to disabling the modules:**

   a. **Ensure that all devices on all modules connected to the faulty motherboard are relinquished by applications and comms stacks that are using them directly.**

b. Eject any CD-ROM from the RMM module using the `eject` command.

c. If the system is mirrored, use SEVM to ensure that all disks associated with the faulty motherboard are taken offline. Use the following commands:

```
# vxdisk list
# vxdg -g diskgroupname -k rmdisk diskname
# vxdisk offline cCtTdDs2
```

An example is given in Step i through Step iv.

i. Run `cmsconfig`. At the prompt, type `v Disk` and note the disk device names associated with the motherboard you are replacing:

```
# v Disk

Item Name          Fault Loc    State                    Page 1 of 2
-------------------------------------------------------------------------------
0    A-MBD 0             A-MBD  enabled
1    B-MBD 0             B-MBD  enabled
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  enabled
8    HDD 0               A-DSK0 enabled      /dev/rdsk/c1t0d0 ( online )
9    HDD 1               A-DSK1 enabled      /dev/rdsk/c1t1d0 ( online )
10   HDD 2               A-DSK2 enabled      /dev/rdsk/c1t2d0 ( online )
11   HDD 6               B-DSK0 enabled      /dev/rdsk/c2t0d0 ( online )
12   HDD 7               B-DSK1 enabled      /dev/rdsk/c2t1d0 ( online )
13   HDD 8               B-DSK2 enabled      /dev/rdsk/c2t2d0 ( online )
14   PCI 1               A-PCI1 enabled
15   PCI 2               A-PCI2 enabled
16   PCI 3               A-PCI3 enabled
17   PCI 4               A-PCI4 enabled
18   PCI 5               A-PCI5 enabled
19   PCI 9               B-PCI1 enabled

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

In this example, disks `clt0d0`, `clt1d0` and `clt2d0` are connected to A-MBD, and disks `c2t0d0`, `c2t1d0` and `c2t2d0` are connected to B-MBD.

To replace the side B motherboard (for example), you must first use SEVM to take offline all the disks connected to B-MBD; that is, `c2t0d0`, `c2d1t0` and c2t2d0.

ii. Type `vxdisk list` to list the associations between the SEVM disk group, disk names and disk device name:

```
# vxdisk list
DEVICE       TYPE      DISK         GROUP        STATUS
c1t0d0s2     sliced    roota        rootdg       online
c1t1d0s2     sliced    dataxa       datax        online
c1t2d0s2     sliced    dataya       datay        online
c2t0d0s2     sliced    rootb        rootdg       online
c2t0d0s2     sliced    dataxb       datax        online
c2t0d0s2     sliced    datayb       datay        online
```

iii. Remove the disks from the disk group using vxdg -g:

```
# vxdg -g rootdg -k rmdisk rootb
# vxdg -g datax -k rmdisk dataxb
# vxdg -g datay -k rmdisk datayb
```

iv. Take the disks offline using `vxdisk offline`:

```
# vxdisk offline c2t0d0s2
# vxdisk offline c2t1d0s2
# vxdisk offline c2t2d0s2
```

3. Check for faulty FRUs.

Also confirm that the remaining motherboard and required connected modules are all enabled. Rectify any faults prior to continuing with motherboard replacement.

Refer to section 4.2, The `cmsconfig` Utility, of the *Netra ft 1800 User's Guide*.

4. Disable the modules connected to the motherboard you are replacing. Ensure that they are removed in the following order:

a. PCI cards

b. HDD modules

c. DSK module

d. RMM module

e. CPUset module

f. CAF module

5. **Disable the motherboard.**

▼ To Remove a Motherboard from the Chassis

1. **Unlock the ejector slides on all disabled modules.**

   The red warning dots will show.

2. **Open the external power breakers associated with the PSUs on the motherboard to be replaced.**

3. **Unplug all the modules that are now disabled.**

Caution – You must remove completely 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 slot, or to remove blanking panels from unused slots.

The *Diag* LED on the remaining CPUset will flash slowly during this procedure.

4. **Loosen the four screws that secure the mid cover.**

   Refer to FIGURE 5-2. Lift off the cover and place it out of the way of the work area.

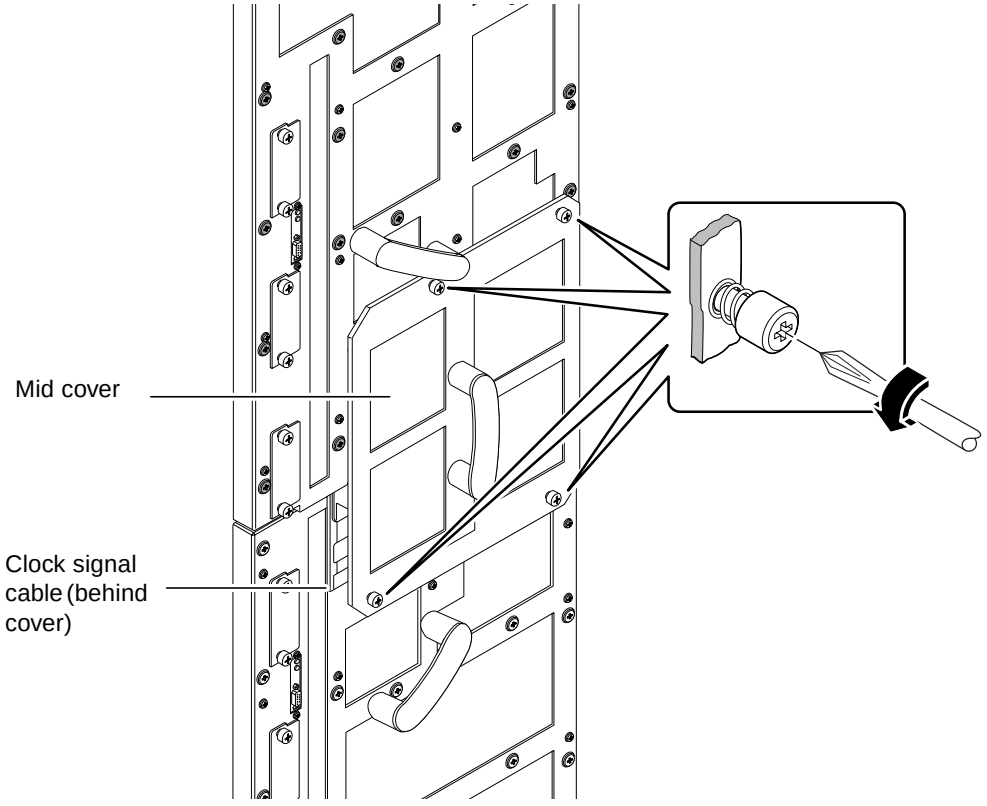

FIGURE 5-2 Location of Mid Cover Securing Screws and Clock Signal Cable (Rear of Chassis)

5. **Gently disconnect the brass connector that secures the clock signal coaxial cable.**

   Refer to FIGURE 5-3.

   Ensure the brass connector does not come into contact with the motherboard.

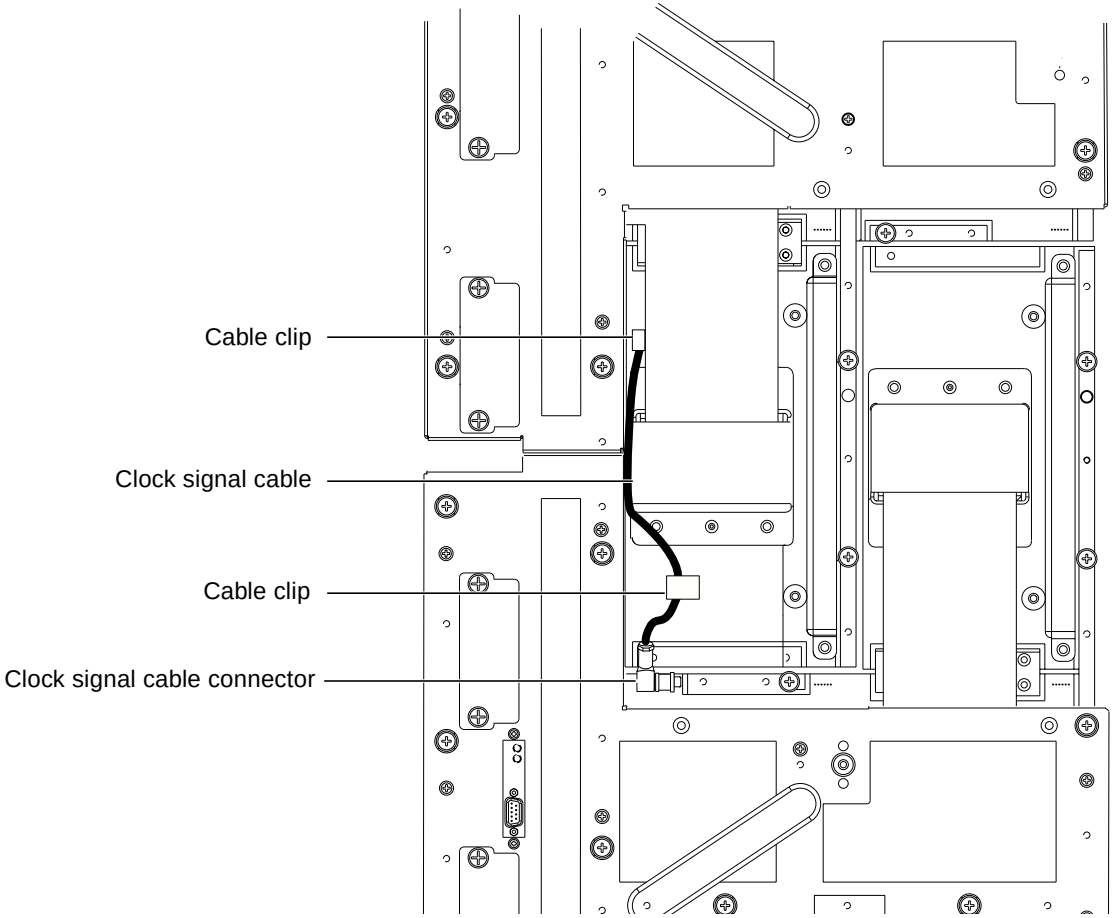

FIGURE 5-3 Clock Signal Cable Connector Location

6. **Secure the cable using the cable clip provided, as shown in** FIGURE 5-4**.**

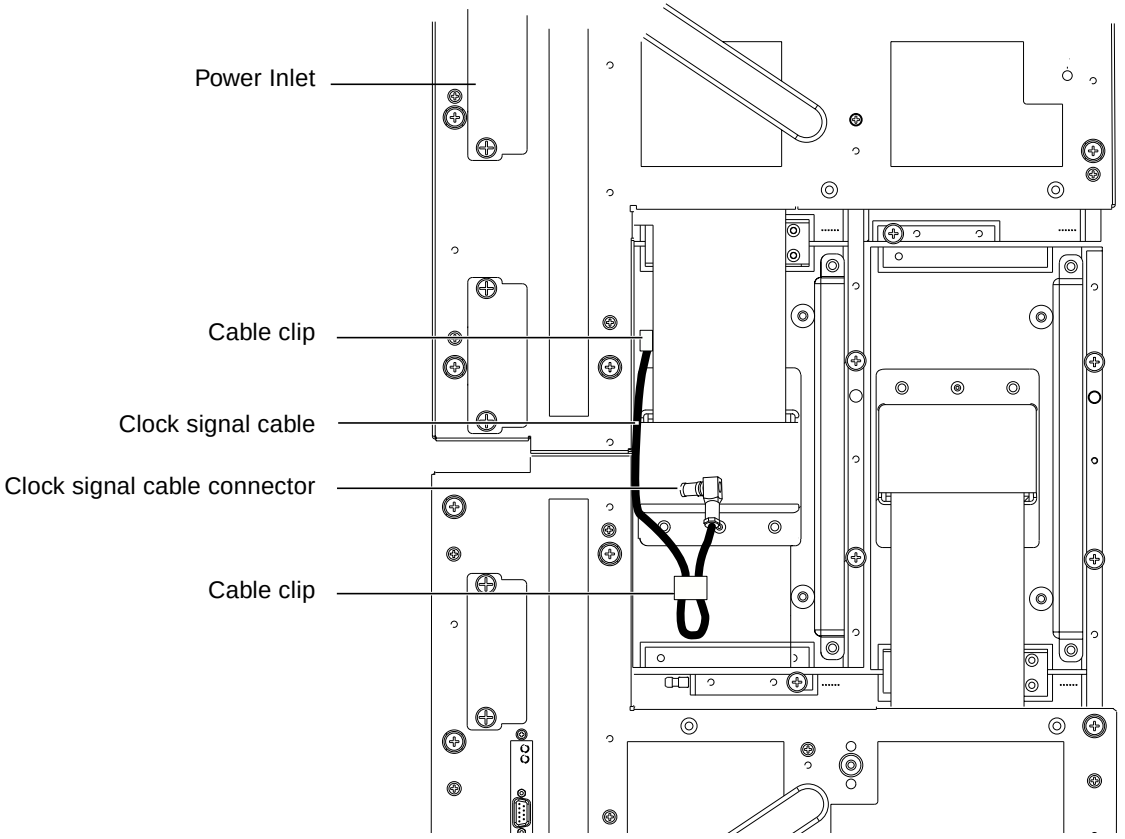

FIGURE 5-4 Securing the Clock Signal Cable

Caution – Take care to secure the connector well away from the motherboard.

7. **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.

8. **Insert two of the CPUset module locking tools into the holes provided.**

   Refer to FIGURE 5-6. Hand-tighten the tools to secure the remaining CPUset module to the motherboard that is to remain in the chassis.

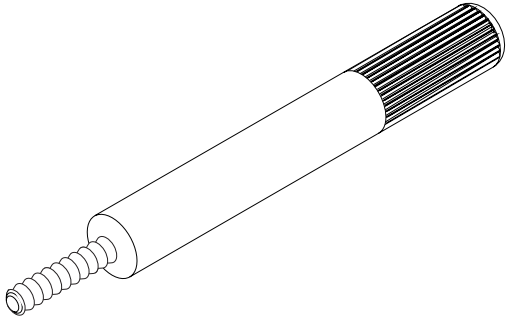

FIGURE 5-5 CPUset Module Locking and Motherboard Ejection Tool

---

Note – The tools have knurled handles to prevent them being overtightened.

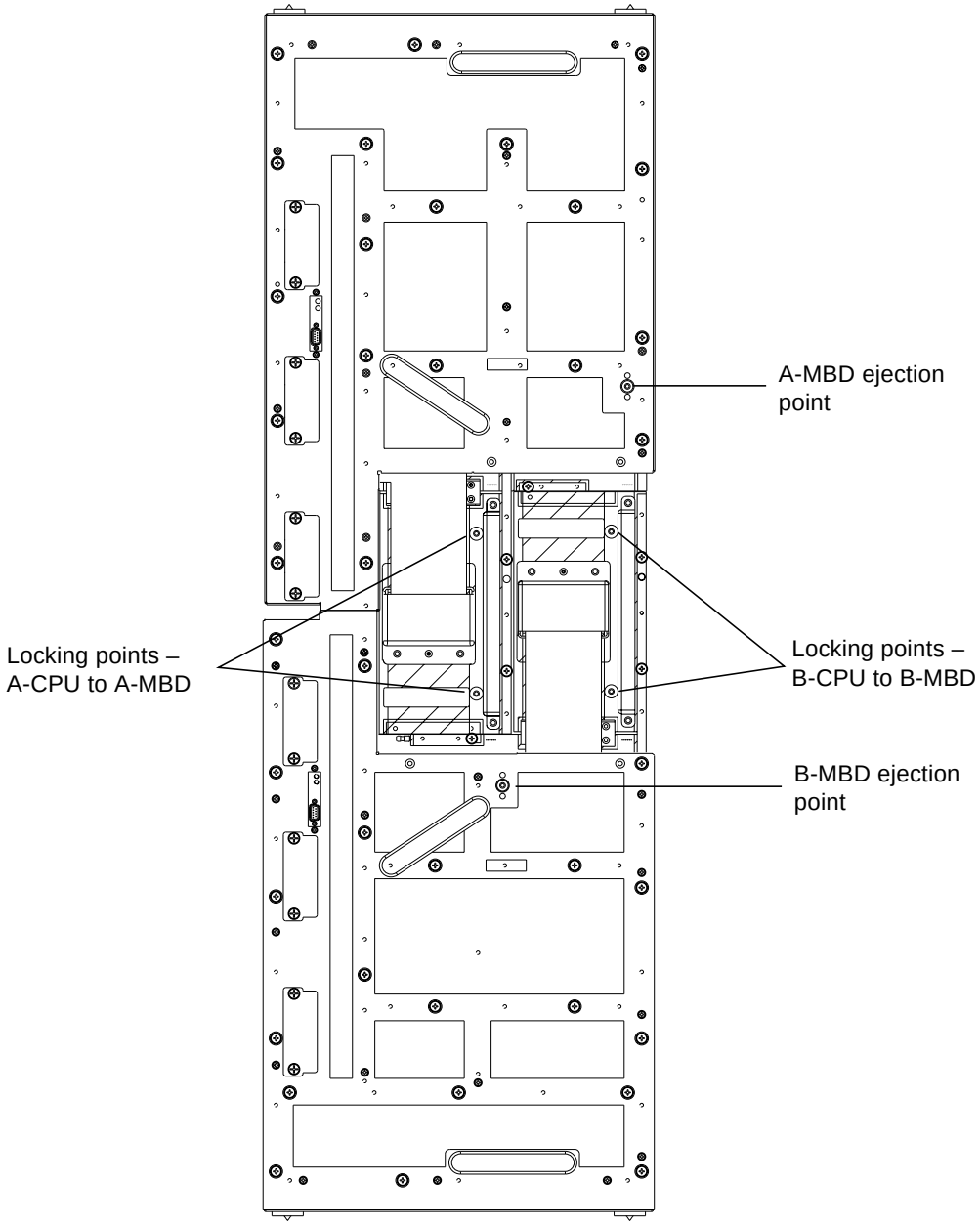

FIGURE 5-6 Location of CPUset Module Locking and Motherboard Ejection Points

9. **Loosen all 22 (A-MBD) or 23 (B-MBD) captive screws securing the motherboard to the chassis (see** FIGURE 5-7**).**

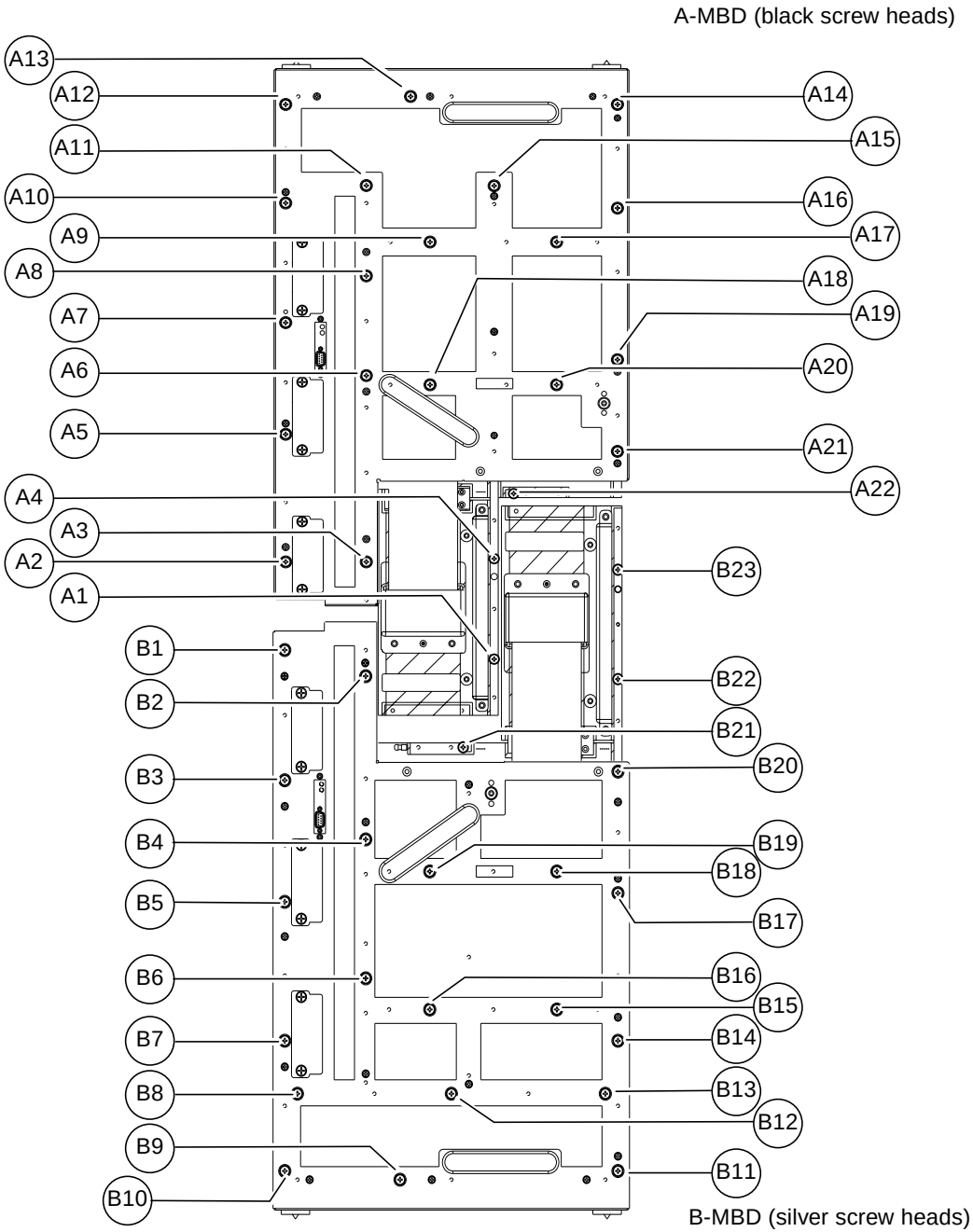

FIGURE 5-7 Motherboard Securing Screws

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.

10. **Insert the motherboard locking tool into the appropriate ejection point.**

    Refer to FIGURE 5-6 on page 120. Gently tighten the tool to lift the motherboard away from the working CPUset module.

Note – The tool has a knurled handle to prevent it from being overtightened.

11. **Using the handles provided, pull the motherboard gently away from the chassis and off the guide pins.**

    Ensure that all cables are kept clear of the motherboard prior to completing this step.

Caution – The motherboard weighs 11.34 kg (23 lb.). Make sure that there is a clear area to which you can transfer the motherboard once it is removed from the chassis.

12. **Remove the ejector tool and replace it in the clip on the mid cover.**

Installing a New Motherboard

Refer to this section for details of how to insert a motherboard in the chassis and then enable the associated modules.

▼ To Insert a Motherboard in the Chassis

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. **Locate the three guide pins which position the motherboard.**

   Refer to FIGURE 5-8.

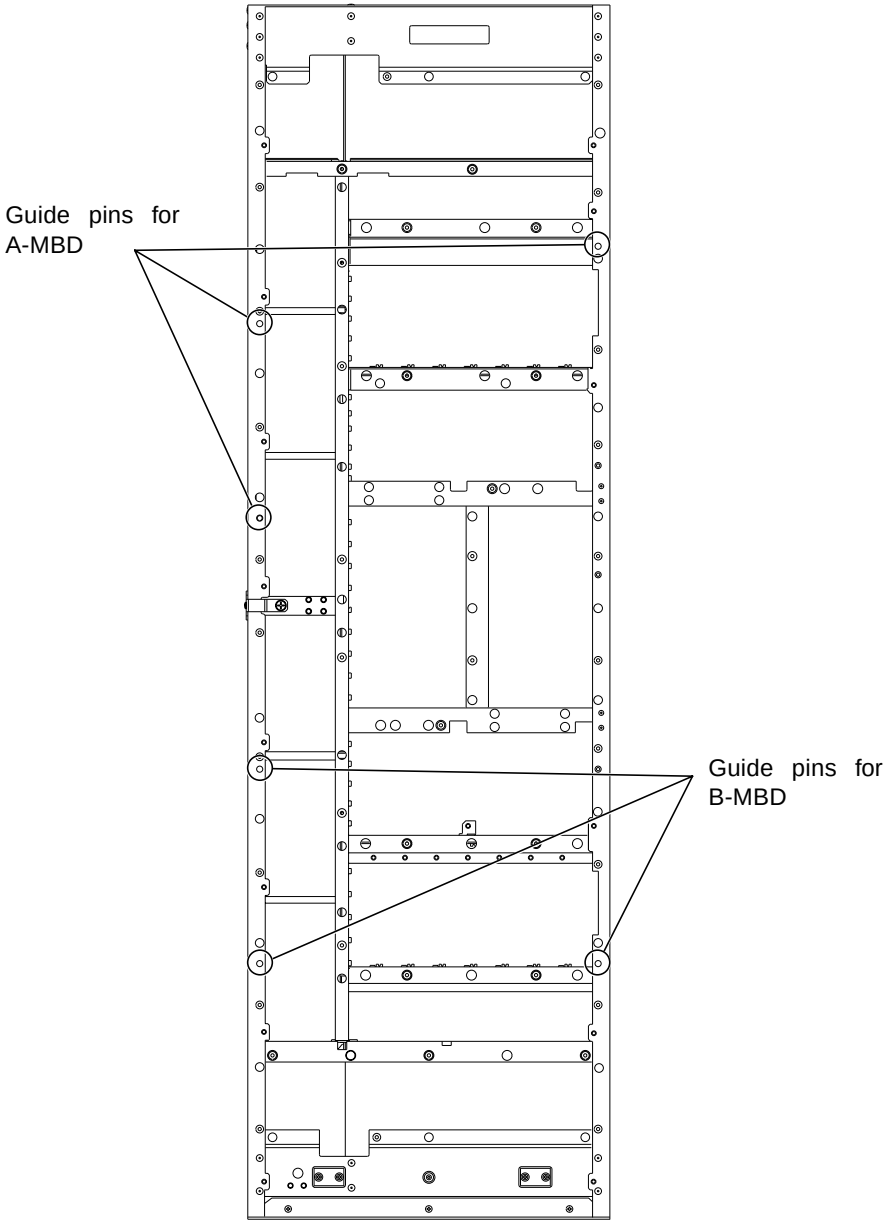

FIGURE 5-8 Location of Motherboard Guide Pins

3. **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.

4. **Push the motherboard gently but firmly home.**

5. **Tighten the captive securing screws.**

   Tighten one of the uppermost screws first so that the motherboard is held in position, then loosen the motherboard lock tool and tighten the screws around the CPUset module. Do not overtighten the screws: maximum torque permitted is 5.4 Nm (4 lb/ft).

6. **Remove the motherboard lock tool.**

7. **Connect the clock signal cable, pushing the connector home firmly.**

   Ensure that the cable itself is routed such that it will not be trapped by the mid cover.

8. **Remove the two CPUset module locking tools and replace them in their clips on the mid cover.**

9. **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).

10. **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).

11. **Re-insert the modules that were previously withdrawn and close the injector latches.**

Caution – Take great care when re-inserting the CPUset module. Failure to accurately align the module may result in damage to the connection pins on the motherboard.

---

12. **Close the external circuit breakers associated with the re-inserted PSUs.**

Enabling a Replacement Motherboard

Enabling a replacement motherboard involves the following stages:

1. Updating the replacement motherboard with the system identity
2. Enabling the connected modules

▼ To Update With the New System Identity

Once the new motherboard has been physically installed, its module EEPROM must be updated with the system identity. Use the `cmsintroduce_mbd` utility to copy the base Ethernet address and the host id to the new motherborard EEPROM. If this step is omitted, the system motherboard will have an Ethernet address of zero.

1. **Update the EEPROM on the new motherboard.**

   If you have replaced motherboard A, type:

```
# /usr/platform/SUNW,Ultra4-FT/SUNWcms/sbin/cmsintroduce_mbd A-MBD
```

   If you have replaced motherboard B, type:

```
# /usr/platform/SUNW,Ultra4-FT/SUNWcms/sbin/cmsintroduce_mbd B-MBD
```

▼ To Enable Connected Modules

1. **Use `cmsconfig` to enable the new motherboard.**

   Refer to section 4.2, The `cmsconfig` Utility, of the *Netra ft 1800 User's Guide*.

2. **Use `cmsconfig` to enable the connected modules in the following order:**

   a. CAF

   b. CPUset

   c. RMM

   d. DSK

   e. HDD modules

   f. PCI cards

   Refer to section 4.2, The `cmsconfig` Utility, of the *Netra ft 1800 User's Guide*.

3. **If the system is mirrored, use SEVM to ensure that all disks associated with the new motherboard are placed online. Use the following commands:**

```
# vxdisk online CCtTdDs2
# vxdg -g diskgroupname -k adddisk diskname=cCtTdD
# vxrecover&
```

An example is given in Step a through Step c.

a. Bring the disks online using `vxdisk online`:

```
# vxdisk online c2t0d0s2
# vxdisk online c2t1d0s2
# vxdisk online c2t2d0s2
```

b. Add the disks to the diskgroup using `vxdg -g`:

```
# vxdg -g rootdg -k adddisk rootb=c2t0d0
# vxdg -g datax -k adddisk dataxb=c2t1d0
# vxdg -g datay -k adddisk datayb=c2t2d0
```

c. Type:

```
# vxrecover &
```

APPENDIX **A**

Manpages

This Appendix contains the following manpages:

- u4ftdog - Netra ft 1800 OSdog driver
- u4ftdogpat - user OSdog patting daemon
- split – introduction to Split mode
- splitadm – to split or merge domains
- splitconf – change the split Master or the ownership of a resource
- splitd.conf – configuration file for u4ftsplitd system split daemon
- splitinfo – displays domain-related information

NAME

u4ftdog - Netra ft 1800 OSdog driver

SYNOPSIS

u4ftdog@*bus_address*:*port_name*

DESCRIPTION

The **u4ftdog** driver is a multi-threaded, multi-instance, non-STREAMS driver that is responsible for controlling the operating system watchdog. It exports an **ioctl**(2) interface that enables applications to register their ability to report system availability. Once registered, the application is responsible for periodically reporting (via the **ioctl**(2) interface) that the system is available. Failure to do this is classed as a system failure and the **u4ftdog** driver then initiates a system reboot via the **cmn_err** (9F) CE_PANIC function.

The **u4ftdog** driver also allows control of the Netra ft 1800 OSdog hardware, which provides a hard failure recovery mechanism. Failure to regularly pat the OSdog hardware causes a system power cycle/reset to be initiated.

HARDWARE INTERFACE

The Netra ft 1800 hardware OSdog functionality is provided by a number of registers on each motherboard. These registers implement a timer count upwards. In normal system operation this counter is reset periodically by **u4ftdog** driver software writing to the “pat” register. If the register is not patted within a specified time limit, the counter reaches a trigger value, which causes the motherboard hardware to initiate an OSdog timeout.

The actions associated with an OSdog timeout are dependent upon the systems operating mode. In a combined system, the OSdog timeout causes a system power cycle of the side whose OSdog timed out. In split mode, the OSdog timeout causes a cpuset reset on the OSdog’s side. The differences in actions are due to the possibility of resource sharing in a split system where one side may be using resources on the other side. If the side was power cycled, the other side’s resources would be reset.

The OSdog time has two possible timeout values: normal (10.74s), which is generally selected when Solaris is running, and extended (85.90s), which is generally selected when system firmware is running.

OVERVIEW

The **u4ftdog** driver comprises three instances; a control instance and two hardware instances. The control instance represents the upper layer of the driver. It handles the user patting ioctl interface and performs the patting of the hardware OSdogs.

The hardware instances represents the OSdog hardware on each motherboard and allows the hardware to be brought online/taken offline in response to cpuset integration and out-of-sync events. The hardware instances have private interfaces, which are not intended for end-user use.

The driver performs continuous latent fault checking on the OSdog hardware once the OSdog has been enabled. The driver checks monitor bits in the hardware registers and ensures that they are pulsing.

CONFIGURATION

The **u4ftdog** driver provides a hardware configuration file, **u4ftdog.conf** which contains various properties that are used during driver initialization. These properties are user configurable and are presented below.

name - a string property that identifies the driver. It is the same for all instances.

parent - a string that identifies the parent of a particular instance. This property encodes the absolute path of the parent node. For the control instance, this property is denoted as **pseudo** since it is not responsible for driving any hardware.

node - a single character string property that defines the instance of the current node. "A" and "B" represent the OSdog hardware on the respective sides and "C" represents the control instance.

reg - a 5-tuple integer property that describes the address space that will be mapped by the hardware instances.

The following configuration file properties apply to all three **u4ftdog** instances:

channel_x_limit - a number of integer properties (one per channel - replace x by the channel number) that describe initial limits (in seconds) for each pat channel. These properties are used to provide OSdog protection during initial boot. They should be set to a value that represents the **no-config** - an integer property that tells the framework that the hardware represented is non self-identifying and will provide a configuration file.

non-prom - an integer property that tells the framework that the OBP does not recognize the device, i.e., it is not a hardware node.

u4ft-aware - an integer property that tells the framework that the instances will provide their own routines to process the online/offline state change requests.

trace - a string property that encodes the trace flags that cause certain trace messages to be output to the message log.

IOCTLS

The **u4ftdog** driver provides ioctls that enable applications to configure/perform user OSdog patting and ioctls that allow system management functions to configure the OSdog. Note that the latter should not be used by users and are not described below.

The following system calls are not supported by the **u4ftdog** driver:

read(2), write(2), mmap(2).

During initialization, the control instance creates a number of device nodes (one per pat channel) - **/dev/u4ftdog:pat***?* where '?' is the pat channel number from 1 to 8.

In addition, each instance creates a device node **/dev/u4ftdog:***[ABC]*, which is used by system control functions. These devices should not be used by users.

USER PAT CHANNEL IOCTLS

The user pat channels allow the user to define a custom system hang detection scheme which monitors aspects of the system appropriate to their needs (see **u4ftdogpat**(1M) for a suitable example).

By default, the user pat channels are active only when a process has the channel device special file open and has configured the limit of the channel. This provides a safety net against problems with the monitoring application that cause it to terminate abnormally and thus cause the OSdog to trigger. This behavior can be overridden using the LINGER_ON_CLOSE ioctl described below.

A typical use of a user pat channel is as follows: open the channel device, set the timeout limit to an appropriate value, loop around monitoring for hangs and patting the OSdog if the system is not hung (you must pat more frequently than the limit period), and if you wish to close the channel first set the limit to zero (disable the channel) then close the channel device.

The following ioctls are applicable to the pat channel devices.

U4FTIOC_OSDOG_SET_LIMIT

enable / **disable** a user patting channel. The argument is a pointer to the a **uint32_t** which represents the desired limit (in seconds) for that pat channel. A pat channel is enabled when a non-zero limit is specified and disabled when a zero limit is specified.

Note that care should be taken when selecting a limit to ensure that the limit is appropriate for the expected pat process. In particular, small values (less than 10) are generally not appropriate.

U4FTIOC_OSDOG_PAT

Pat the channel to inform the **u4ftdog** driver that the patting process is active and that the driver should reset the pat channel's OSdog timer. The ioctl takes no argument (supply a NULL pointer).

The patting must be performed often enough to ensure that the OSdog does not go off. Take care to ensure that any scheduling delays (if permissible) are accounted for.

U4FTIOC_OSDOG_LINGER_ON_CLOSE

Specify that the driver should not automatically disable the pat channel if the channel is closed. This is provided to allow system shutdown operations to be OSdog protected.

The ioctl takes a **uint32_t parameter** whose value is 1 to enable linger-on-close and zero to disable it.

Note that if a device is closed and then re-opened, the linger-on-close attribute will be disabled.

ERRORS

An open() will fail if:

ENXIO
The driver is not installed in the system.

EINVAL
The device minor being opened is the wrong type.

An ioctl() will fail if:

EFAULT
A bad user-space address was specified.

EINVAL
A non-existent control command was requested or invalid parameters were supplied.

EIO
An I/O error occurred.

ENXIO
The driver is not installed in the system or the required hardware instance(s) were not online.

FILES

/platform/SUNW,Ultra-4FT/kernel/drv/u4ftdog # device driver module.
/platform/SUNW,Ultra-4FT/kernel/drv/u4ftdog.conf # hardware configuration file.
/dev/u4ftdog:A # side A hardware device /dev/u4ftdog:B # side B hardware device /dev/u4ftdog:C # generic control device /dev/u4ftdog:pat? # pat channel devices

SEE ALSO

u4ftdogpat(1M), **cmn_err**(9F), **driver.conf**(4),

NAME

u4ftdogpat - user OSdog patting daemon

SYNOPSIS

u4ftdogpat -p *period* **-l** *limit* **-c** *channel* [**-i**] [**-o** *monitoring options*]

DESCRIPTION

The **u4ftdogpat** daemon provides a software operating system watchdog (OSdog). It communicates with the **u4ftdog**(7D) driver periodically to pat a specific OSdog channel, thus demonstrating that the system is running processes. If a channel is not patted within a specified time limit, the **u4ftdog** driver initiates a operating system panic (see **cmn_err**(9F)) causing the machine to reboot. The panic dump can then be analyzed at a later point in time to determine the cause of the panic.

OPTIONS

–c *channel*

The software OSdog channel to use to protect the system. The **u4ftdog**(7D) driver provides a number of different channels enabling different hang conditions to be monitored.

–i Specify that the daemon should not detach itself from the controlling terminal. This is generally used to allow the daemon to be run as a "respawn" entry in /etc/inittab (see **inittab**(4)).

–l *limit*

The time (in seconds) that the **u4ftdog** driver will wait for a pat command before concluding that the system is hung and, hence, invoking an operating system panic.

–p *period*

Used to specify the period between OSdog pats. This is the time (in seconds) that the daemon waits before issuing a pat command to the **u4ftdog**(7D) driver.

monitoring options

Specify monitoring options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

bind=*processor_id*

This option specifies the processor_id (see **processor_bind**(2)) that the daemon process should be bound to. This allows the daemon to monitor a particular processor for hangs. By default, the daemon inherits its parent’s binding.

bindset=*pset*

This option specifies the processor set (see **pset_bind**(2)) that the daemon process should be bound to. This allows the daemon to monitor a particular processor set for hangs. By default, the daemon inherits its parent’s binding.

RT=*priority*

This option specifies the priority (see **priocntl**2)) of the patting daemon as being in the real-time class with the specified numeric priority.

TS=*priority*

This option specifies the priority (see **priocntl**2)) of the patting daemon as being in the timesharing class with the specified numeric priority.

unlock

Specify that the daemon’s address space should not be locked down in memory (see **mlockall**(3C**)**). By default, the daemon’s address space is locked down in memory.

EXAMPLES

The following example uses software OSdog channel 1 to pat the OSdog every 5 seconds with a time limit of 20 seconds before the OSdog is triggered. All scheduling priorities and processor bindings are inherited from the invoking process.

u4ftdogpat -c 1 -p 5 -l 20

The following example uses software OSdog channel 4 to pat the OSdog every 5 seconds with a limit of 10 seconds. The process runs with real-time priority 59.

u4ftdogpat -c 4 -p 5 -l 10 -o RT=59

The following example uses software OSdog channel 1 to pat the OSdog every 10 seconds with a limit of 60 seconds. The daemon does not detach from the controlling terminal. This example is suitable for use in a /etc/inittab entry.

u4ftdogpat -c 1 -p 10 -l 60 -i

SEE ALSO

u4ftdog(7D), **cmn_err**(9F), **inittab**(4), **mlockall**(3C,**)** **priocntl**(2), **processor_bind**(2), **pset_bind**(2).

NAME

split – introduction to Split mode

DESCRIPTION

The Netra ft 1800 fault-tolerant platform is capable of running in two modes: as a single fault-tolerant system (combined mode), and as two domains, each acting as an independent system (split mode). The split-mode support software manages the orderly transition from one mode of operation to the other and coordinates the proper transfer of system resources between the domains.

When the system is in combined mode, the system is composed of a single domain that encompasses all the system resources. In combined mode, the system has a *combined* attribute alluding to its fault-tolerant nature. In split mode, each independent domain has a *split* attribute implying that it is not part of a fault-tolerant system.

In split mode, each domain contains at least a CPU, MBD, CAF and PSUs, and, possibly, some other modules. Each module is owned by one domain. Ownership can be changed, but can never be shared. Certain modules (CAF, RMM, MBD, CPU, and PSUs) are considered fixed (unmovable) and are statically owned by their "natural" domain when the system is in split mode. All other modules are movable and can be freely assigned to either domain. When the system is in combined mode, the assignment of ownership of modules is meant as a preparatory step for the system to transition to split mode. Therefore, in combined mode the assignment of ownership of a resource to a domain that will exist only after the system is split is taken as potential future ownership of that resource should the system transition from combined mode to split mode.

One domain is defined as the split master. While the system is in combined mode, the split master refers to the future master should the system become split. While the system is in split mode, the split master can always initiate the transfer of ownership of a module from one domain to another. Other domains can initiate a transfer of ownership only if they can successfully negotiate such a transfer with the split master. Note that the split master can be changed by the user, both in combined and split modes.

Mode (or attribute), split master, and module ownership persist across the reboot of any domain.

USER INTERFACE

The split mode support software provides both a command line interface and an application programming interface (API). Both the commands and the API enable management of the mode and resources as well as enquiry functions to get information about the current mode, domain, mastership and ownership of resources.

Commands
*/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/***{splitinfo,splitadm,splitconf}**

The split library
*/usr/platform/SUNW,Ultra-4FT/SUNWcms/lib/***{libu4ftsplit.so**,**libu4ftsplitmt.so**}
The split API is defined in */usr/platform/SUNW,Ultra-4FT/SUNWcms/include/***split_api.h**

SPLIT SUPPORT

The software subsystem that implements split mode support consists of:

The split daemon
u4ftsplitd, which executes on each system while split and on the combined system when in fault tolerant mode. It is the daemon that controls all the domain attribute operations and resource ownership. One copy of the daemon runs on each domain of a multi-domain system and the daemons communicate with each other using sockets. The daemons maintain the domain and ownership information for all the resources in the system. This information is kept consistently in several locations. There are in-memory and persistent copies.

The InterCpuset Network driver

icn, used in split mode as the basis of communication between the two daemons.

SEE ALSO

splitadm(1M), **splitconf**(1M), **splitinfo**(1M), **set_slot_owner**(3), **set_domain_attribute**(3), **get_slot_status**(3), **get_domain_attributes**(3), **split_lock**(3), **split_unlock**(3), **splitd.conf**(4)

NAME

splitadm – to split or merge domains

SYNOPSIS

/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/splitadm [**-f**] [**-t** *timeout*] [**-w** *domain*] [**-d** *domain...*] **split -w** *domain*

/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/splitadm [**-d** *domain...*] **merge**

DESCRIPTION

splitadm is used to split a domain into a number of independent sub-domains, or to merge a number of sub-domains into a single domain.

All the domain names used must be valid and specified in **splitd.conf**.

This command must be executed with super-user privileges.

Split a domain

splitadm [**-f**] [**-t** *timeout*] [**-w** *domain*] [**-d** *domain...*] **split**

When a domain splits, a set of independent sub-domains is created, each one owning a subset of the original domain's resources. One sub-domain (called the winner or the surviving domain) may have the system identity of the original domain and may keep running the existing processes. All other sub-domains are given a new system identity.

The following options may be supported for splitting:

-t *timeout*
overrides the default timeout (see **splitd.conf**(4)) within which all the implied (if any) operations on the resources involved in this transition must complete. *timeout* must be expressed in seconds.

-f enforces the success of the command even if some operations on the involved resources fail or do not complete within the timeout.

-w *domain*
specifies the winner domain, which inherits the system identity of the original domain.

-d *domain...*
specifies a list of resulting sub-domains.

The configuration, as reported by **splitinfo**(1M) before splitting, is realized if the command succeeds.

Merge a set of domains

splitadm –w *domain* [**-d** *domain...*] **merge**

This command merges a set of domains into a single one, which owns all the resources of the original sub-domains. The resulting domain inherits the system identity of one of the original sub-domains (the winner) and keeps running the existing services; all other merged domains are stopped and their resources added to the surviving domain.

The following options may be supported for merging:

-w *domain*
specifies the winner domain.

-d *domain...*
specifies a list of sub-domains to merge together.

Netra ft 1800

On this platform, only two system configurations are supported: one single domain, or two independent domains. When the system is operating in combined mode, a single domain exists called

C. When the system is operating in split mode, two sub-domains exist **A** and **B**. Furthermore, the system identity of the combined domain will be the same as the winner sub-domain.

The only valid domain names, specified in **splitd.conf**, are **A**, **B** and **C**.

On this platform, the **-d** option is ignored since the only valid subdomains are **A** and **B**.

Splitadm(1M) is responsible for the mode transitions on this platform.

Splitting

When the combined domain splits, two independent sub-domains are created (**A** and **B**), each one owning a subset of the total system resources. One domain (the winner or surviving) has the system identity of the combined domain and keeps running the existing processes. The other domain can be rebooted by the user, and will come up with a new system identity.

At first the command tries to unconfigure some modules in the locations belonging to the new domain. The command fails if any disable process fails or does not complete within a certain timeout (unless the **-f** option is used).

If the option **–w** is not specified, the winner (surviving) domain defaults to the split master, as reported by **splitinfo -m.**

Merging

The merge operation moves the system from split to combined mode: the losing domain is reset and its CPU is reintegrated in the combined domain. The specified winner (surviving) sub-domain must be **A** or **B**.

EXAMPLES

Split a domain, surviving domain is split master:

```
example# splitadm split
domain = A
attributes = split,master
master = A
```

Split a domain, surviving domain is B:

```
example# splitadm -w B split
domain = B
attributes = split
master = A
```

Split a domain, surviving domain is A (fails because B-CAF cannot be disabled):

```
example# splitadm -t 30 -w A split
Error: cannot disable B-CAF, operation initiated but not completed
```

Split a split domain:

```
example# splitadm split
already split
```

Merge a domain (surviving domain will be A):

```
example# splitadm -w A merge
domain = C
attributes = combined
master = B
```

Merge a combined domain:

```
# splitadm -w A merge
already merged
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWspltu
Interface Stability	Unstable

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

splitconf(1M), **splitinfo**(1M), **splitd.conf**(4), **split**(5)

NAME

splitconf – change the split Master or the ownership of a resource

SYNOPSIS

/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/splitconf **-m** *master_domain*
/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/splitconf [**-f**] [**-t** *timeout*]
-l *location* **-o** *owner_domain*

DESCRIPTION

splitconf is responsible for controlled changes in resources ownership and split mastership.

Any domain is allowed to change the split master with no restrictions.

Resource ownership changes can be initiated from any domain. All the domains involved in the operation should have the ability to communicate each other (see **split**(5)) in order to negotiate and realize the change, and to keep the same view of the system configuration.
If, for any reason, the communication between the domains involved in the transition stops, only the split master is allowed to initiate a configuration change.

This command must be executed with super-user privileges.

Assign the split master

splitconf -m *master_domain*

Elects *master_domain* as the split master.
master_domain must be a valid domain name, listed in **splitd.conf**.

Assign ownership to a location

splitconf [**–f**] [**-t** *timeout*] **-l** *location* **-o** *owner_domain*

Assign the ownership of the resource in *location* to *owner_domain.*.

The following options are supported:

-t *timeout*
overrides the default timeout (see **splitd.conf**(4)) within which all the implied operations (if any) on the resource involved must complete. *timeout* must be expressed in seconds.

-f enforces the success of the command, even if some operations on the involved resource fail or do not complete within the timeout.

-l *location*
specifies the location of the resource whose ownership is changed.

-o *owner_domain*
specifies the domain to which the resource must be assigned. A sub-domain may be specified, before splitting a domain, in order to prepare a target configuration before a split operation (see **splitadm**(1M) for details).

If the timeout expires before the required operations (if any) on the resource are completed, or if some of the operations fail, the command fails and the ownership of the resource is not altered. Otherwise, on success, *location* is assigned to *owner_domain.*

The command always refuses to change owner of Fixed locations.

Netra ft 1800

When in combined mode, the command simply assigns *location* to *owner_domain* (**A** or **B**). The options **-f** and **-t** are ignored.

When in split mode, the command tries to unconfigure the module in *location* before transferring ownership.

On this platform, the allowed resource locations are specified in **cms_location**(5).

EXAMPLES

Assign mastership to B:

```
example# splitconf -m B
master = B
```

Assign A-PCI2 to B:

```
example# splitconf -l A-PCI2 -o B
A-PCI2: B Movable
```

Assign A-CAF to B (fails because A-CAF is a fixed location):

```
example# splitconf -l A-CAF -o B
splitconf: Error: A-CAF Fixed, cannot change owner to a Fixed slot
```

Assign A-PCI2 to B (fails because A-PCI2 didn't disble within the timeout):

```
example# splitconf -t 10 -l A-PCI2 -o B
splitconf: Error: cannot change owner, split daemon operation timeout
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWspltu
Interface Stability	Unstable

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

splitinfo(1M), **splitadm**(1M), **splitd.conf**(4), **split**(5), **cms_location**(5)

NAME

splitd.conf – configuration file for u4ftsplitd system split daemon

SYNOPSIS

/etc/splitd.conf

DESCRIPTION

The file **/etc/splitd.conf** contains information used by the split daemon, **u4ftsplitd**. This information is used to initialize the daemon's operation timeout, enable logging, and initialize the IP addresses and port numbers used by the daemons running on multiple domains.

The file can contain an unlimited number of one-line entries. Each entry can consist of up to two space-separated fields:

key value

The recognized keys and their values are as follows:

domain
A character string with the name of a domain. This is used to check for validity of domain operations.

host_prin
Two host names, one for each domain, representing the interface used by the two split daemons to exchange messages with each other. On a Netra ft 1800 system, it is recommended that you specify icn interfaces here.

host_alt
Two alternative host names, one for each domain, representing the interface used by the two split daemons to exchange messages with each other in cases when the primary interface defined in **host_prin** malfunctions.

port_address
A character string that is composed of a 4-tuple separated by space. The 4-tuple appear in the following order: source_domain, which is the name of the domain that will act as the sender for this port, destination_domain, the port_number, and a flag that indicates if this is a bidirectional port or not. Note that several ports may be specified for each pair of domains to be used by the split daemons on various domains to communicate with each other. On a Netra ft 1800 system the split daemon uses unidirectional ports and binds itself to the first available one in the configuration file.

dolog
Used to enable split daemon logging. The value is *yes* or *no.*

timeout
The value is an integer which is the operation timeout in seconds.

hostnames
The two host names for the domains of a split system. These host names should be included in the */etc/hosts* file or available from NIS.

There must be only one identical copy of this file, i.e., split daemons on all domains use identical configuration files. The daemons obtain the initial configuration parameters from the configuration file **splitd.conf**. The value of *timeout*, if specified by a command or API option, overrides the default value.

EXAMPLE

In this example, a Netra ft 1800 system called *treeton* can be split into two nonfault-tolerant systems called *treeton* and *treeton-2*. The two split daemons running on each side use an icn interface as their primary copmmunications path. An alternative communication path is defined on an Ethernet interface.

```
# The domain-address section.
# Add the primary and alternative host names
# (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 treeton-i0 treeton-2-i0
host_alt treeton treeton-2

# The port-number section.
port-address a b 3501
port-address b a 3502
port-address a b 3502
port-address b a 3503

# The misc-section
timeout    60
dolog      yes

#The hostnames section.
# Add the host names of the two sides here.
# Example:
# hostnames thishost thishost-2
#host_alt thishost thishost-2
hostnames treeton treeton-2
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWspltr
Interface Stability	Unstable

NAME

splitinfo – displays domain-related information

SYNOPSIS

/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/splitinfo [**–d**] [**–a**] [**–m**] [**–l** *location...*] [**–o** *domain*]

DESCRIPTION

This command retrieves information concerning domains and their attributes, as well as the resources on the system.

OPTIONS

The following options are supported:

–d prints the current domain name.

–a prints a list of attributes associated with the current domain.

–m prints the current split master.

–l *location...*
prints the owner and status (Fixed or Movable) of all *location* in the list. Use **-l all** to print the info on all the locations in the system.

–o *domain*
prints the resources (list of the locations) owned by *domain.*

splitinfo with no arguments does **–dam –lall**.

Netra ft 1800

On this platform, in combined mode domain **C** owns all the resources. Therefore, **splitinfo** always returns as the owner of a given resource the (actual or prospective) sub-domain **A** or **B**.

The allowed resource locations are specified in **cms_location**(5)

EXAMPLES

Print the current domain:

```
example% splitinfo –d
domain = A
```

Print the owner (actual or perspective) and status of A-PCI3:

```
example% splitinfo -l A-PCI3
A-PCI3: A Movable
```

Print the current split Master:

```
example% splitinfo –m
master = B
```

Print the current domain, attributes and Master:

```
example% splitinfo –dam
domain = B
```

```
attributes = split,master
master = B
```

Print all the locations currently owned by A:

```
example% splitinfo –o A
A-PCI0
A-PCI7
A-CAF
A-DSK
A-RMM
A-CPU
A-MBD
A-PSU0
A-PSU1
A-PSU2
B-PCI0
B-PCI1
```

Print a complete split information:

```
example% splitinfo
domain = A
attributes = split
master = B
A-PCI0: A Movable
A-PCI1: A Movable
A-PCI2: A Movable
 .
 .
 .
B-PSU1: B Fixed
B-PSU2: B Fixed
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWspltu
Interface Stability	Unstable

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

splitadm(1M), **splitconf**(1M), **split**(5), **cms_location**(5)

APPENDIX **B**

The cmsphonehome Script

cmsphonehome is a user-configurable script that is called by the CMS when a module fails.

The path is:

/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/cmsphonehome

CODE EXAMPLE B-1 cmsphonehome Script

```
#!/bin/sh
# @(#)cmsphonehome.sh      1.5    99/10/01 SMI
# /* Copyright (c) 1998, by Sun Microsystems, Inc.
# /* All rights reserved
# /*=========================================================================
# cmsphonehome - report a failure by phoning home
#
# SYNOPSIS
# --------
# 'cmsphonehome' 'module' 'module_no' 'location' 'old_state' 'new_state'
#                'faulty' 'info'
#
# DESCRIPTION
# -----------
# 'cmsphonehome' is a script called by the CMS when any module fails. It is
# intended that this script can be altered to report module 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       module type as known to the CMS
#            module_no    module no as known to the CMS
#            location     location of the module
#            old_state    previous state of the module
#            new_state    current state of the module
#            faulty       this event implies the module needs replacing
```

CODE EXAMPLE B-1 cmsphonehome Script *(Continued)*

```
#             info          information string
#
# '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 and
# trigger an alarm but these are commented out.
#
# SEE ALSO
# --------
# cmsconfig(1MFT)
# ------------------------------------------------------------------------*/

MOD_NAME=$1
MOD_NUM=$2
LOCATION=$3
OLD_STATE=$4
NEW_STATE=$5
FAULTY=$6
INFO="$7"
TMP=/tmp/phonehome$$

MACHINE=`/usr/bin/uname -n`
DATE=`date`
# cat > $TMP << EOF
#
# Configuration change report
# ---------------------------
#
# date:                             $DATE
# machine name:                     $MACHINE
# module:                           $MOD_NAME $MOD_NUM
# location:                         $LOCATION
# information string:               $INFO
# implies module needs replacing:  $FAULTY
# EOF
# cat $TMP | mailx -s "$MACHINE:$LOCATION:$INFO" root
# [ $FAULTY = "yes" ] && /usr/platform/SUNW,Ultra-4FT/sbin/u4ftalctl alarm1=on
# rm -f $TMP

# Ensure it returns success
exit 0
```

APPENDIX **C**

Automatic Firmware Update

Occasional upgrading of the firmware on an ft1800 system may be necessary to add new functionality, fix bugs, and so on. New firmware will be delivered using the standard patch and install mechanisms. The upgrade procedure for installing new firmware runs automatically and should require no user intervention.

Note – A firmware update is a time consuming operation (of the order of minutes) and it should be scheduled to allow it to proceed to conclusion; interruption will cause system corruption.

Note – The system must be reset and/or power cycled for the newly-upgraded firmware to take effect. This is handled automatically at boot time or when the module concerned is enabled through the CMS.

As the firmware upgrade proceeds, appropriate messages are displayed, either on the console or in `cmsconfig`.

Operation

The system functions automatically at discrete points in time as follows:

Preboot Checks

The PROM partially validates the firmware as part of its preboot sequence and this may trigger various reset actions to ensure that newly loaded firmware is activated.

Boot Checks

Every time the system boots, a check is made to ensure that the system is running firmware supported by the installed version of the operating system. If this check is successful, the boot proceeds as normal.

If the check fails, one or more pieces of firmware are incompatible with the operating system and an attempt will be made to rectify the situation by installing new version(s) of the firmware if they are available. During the update, a message is displayed on the console. Once the firmware has been installed, the system will automatically reboot.

If the firmware installation fails, an appropriate message will be printed on the console and the boot will proceed as normal.

Firmware updated during boot includes:

- Boot side PROM
- Boot side motherboard FPGA
- Boot side RCP

If the system is configured for fault tolerant operation, the following additional firmware may be updated:

- Nonboot side motherboard FPGA
- Nonboot side RCP

Enabling the Other CPUset Module

Since the non-boot side PROM cannot be checked or upgraded automatically when the system boots, the compatibility of that PROM is established when the other CPUset module is enabled by the CMS.

If an upgrade is required, it will be performed automatically. However, this process will delay the enabling of the FRU by CMS. An informative message is displayed by `cmsconfig` will the upgrade occurs. It is important that the process is not interrupted otherwise the PROM will be corrupted.

Regardless of the success of the upgrade operation the CPU FRU will still enable as normal unless another problem is detected.

Motherboard Hot Swap

If a motherboard is hot swapped, the firmware on the replacement board may not be compatible with that in the running system. Therefore, it is also checked when the motherboard FRU is enabled in the CMS.

If the firmware is out of date, the new firmware is downloaded (with an informative message). However, there are then two possible outcomes that depend upon the power state of the motherboard.

- If the power was OFF, after the update the enable will proceed as usual with no further user interaction.
- If the power was ON, the enable of the motherboard will FAIL and its state in CMS will be set to `enable_failed`. This situation can be cleared only by using the CMS to first disable and then re-enable the upgraded motherboard. This action is required because new firmware downloaded to a motherboard is not activated until the next power-on.

Firmware Updates and Module EEPROMS

Each time a firmware update takes place, the part number of its firmware image is recorded in the module's EEPROM. You can read these part numbers using the `cmsfruinfo`(1M) utility.

For example:

```
# cmsfruinfo -1 A-CPU EE_CPU_PROM
    EE_CPU_PROM_PARTNO=2587354
    EE_CPU_PROM_DASH=10

# cmsfruinfo -1 B-MBD EE_MBD_BRIDGE_FWARE
    EE_MBD_BRIDGE_FWARE_PARTNO=2587134
    EE_MBD_BRIDGE_FWARE_DASH=08
```

APPENDIX **D**

Alarms Utility

Access to the Netra ft 1800 alarms is provided by the `u4ftalctl`(1M) utility, which enables you to toggle the alarms and UNIX-running watchdog, and report their status.

You can implement `u4ftalctl` utility directly from the command line using the following format. You can also incorporate these commands into a shell script.

- To set or clear alarms:

```
% u4ftalctl alarm=[off|on]
```

- To set or clear watchdog:

```
% u4ftalctl watchdog=[off|on]
```

- To reset alarms or UNIX-running watchdog:

```
% u4ftalctl reset
```

- To report status of alarms and UNIX-running watchdog (in fault-tolerant mode):

```
% u4ftalctl status
watchdog=off
alarm0=off
alarm1=off
alarm2=off
alarm3=unavailable
alarm4=unavailable
alarm5=unavailable
```

- To report status of alarms and UNIX-running watchdog (in nonfault-tolerant mode):

```
% u4ftalctl status
watchdog=off
alarm0=off
alarm1=off
alarm2=off
alarm3=off
alarm4=off
alarm5=off
```

The following example shows how to assign the status of each alarm to a (Bourne or Korn) shell variable:

```
% eval u4ftalctl status
% echo $alarm1
off
```

The following code example shows how to set and clear an alarm.

CODE EXAMPLE D-1 Setting and Clearing an Alarm

```
/*
 * alarmctl.c
 *
 * program to set/clear alarms
 * synopsis - this program expects 2 args :- alarm# and setting
 *
 */

#include <unistd.h>
#include <sys/ioccom.h>
#include <fcntl.h>
#include <errno.h>
#include "u4ftalarm_io.h"

char *dev = "/dev/u4ftalarm:ctl";

int main(int argc, char **argv)
{
        int fd;
        int read_state = 0;
        u4ft_aldata_t *ad = NULL;
```

CODE EXAMPLE D-1 Setting and Clearing an Alarm *(Continued)*

```
        if (argc < 3) {
                printf("Usage: alarmctl [alarm#] [on/off]\n");
                perror(argv[0]);
                exit(1);
        }

        if ((fd = open(dev, O_RDONLY)) < 0) {
                printf("open failed\n");
                perror(argv[0]);
                exit(1);
        }

        ad = (u4ft_aldata_t *)calloc(1, sizeof(u4ft_aldata_t));
        if (ad == NULL) {
                printf("%s: calloc failed\n", argv[0]);
                perror(argv[0]);
                exit(1);
        }

        ad->u4ftalarm_no = atoi(argv[1]);
        if (ad->u4ftalarm_no < 0 || ad->u4ftalarm_no > 5) {
                printf("%s :- invalid alarm number\n", argv[0]);
                exit(1);
        }

        if (strcmp(argv[2], "on") == 0)
                ad->u4ftalarm_state = 1;
        else if (strcmp(argv[2], "off") == 0)
                ad->u4ftalarm_state = 0;
        else if (strcmp(argv[2], "state") == 0)
                read_state = 1;
        else {
                printf("%s :- invalid alarm state\n", argv[0]);
                exit(1);
        }

        if (read_state) {
                if (ioctl(fd, U4FTIOCALSTATE, ad) < 0) {
                        printf("U4FTIOCALSTATE failed for alarm %d\n", ad-
>u4ftalarm_no);
                        perror(argv[0]);
                } else {
                     printf("The state of alarm %d is %d\n", ad->u4ftalarm_no,
ad->u4ftalarm_state);
                }
        } else if (ioctl(fd, U4FTIOCALCTL , ad) < 0) {
                printf("U4FTIOCALCTL failed for alarm %d\n", ad->u4ftalarm_no);
```

CODE EXAMPLE D-1 Setting and Clearing an Alarm *(Continued)*

```
                perror(argv[0]);
        }

        close(fd);
}
```

APPENDIX **E**

Split Mode for Patch D

This appendix provides an alternative example split procedure to that documented in “Split Example” on page 64, which is designed to be used when running a Netra ft 1800 system with upgrade patch 108145-10 (patch D) installed.

Split Example

The following example demonstrates how to split a Netra ft 1800, upgrade a component such as an software patch, and then remerge the system. Each stage of the split process is documented, with a description of the state of the system at the end of each stage.

The procedure documented here is an instructor-led demonstration of split mode on the Netra ft 18000 system. As such, it is not intended as a tutorial, but provides one example of the use of split mode. The procedure should be used in conjunction with the *Netra ft 1800 User’s Guide* (part number 805-4529-10).

This procedure assumes that you have some user-level knowledge of SEVM 2.5.

Assumptions

- If NIS is running, edit the /etc/nsswitch.conf file so that the line:

```
hosts: xfn nis [NOTFOUND=return] files
```

reads

```
hosts: files xfn nis [NOTFOUND=return]
```

`files` is now the first option on the line, which will force the system to read the local `/etc/hosts` file first.

- Before the split, the combined system has the host name `seagoon` and IP address 129.156.203.152. After the split, side A will be the split winner with host name `seagoon` and side B will be the loser with the host name `seagoon-2` and IP address 129.156.203.18.
- OBP (PROM) variables:
  - *use-nvramrc?*=true
  - *auto-boot?*=true
  - *boot-device*=a-dsk0 b-dsk0
  - *diag-device*=a-dsk0 b-dsk0
  - *diag-switch?*=false

  Check using the `eeprom` command and, if necessary, set using:

  ```
  eeprom use-nvramrc?=true
  eeprom auto-boot?=true
  eeprom boot-device=a-dsk0 b-dsk0
  eeprom diag-device=a-dsk0 b-dsk0
  eeprom diag-switch?=false
  ```

  The *boot-device* PROM variable is concatenated with an NVRAM variable, *boot-list*, to produce a sequence of disks from which the system tries to boot. The *boot-list* variable is constructed from a motherboard EEPROM variable, which comprises the HDDs in the SEVM root disk group. Setting *boot-device* to the value above ensures that the system will try to boot from A-DSK0 and, if this fails, from B-DSK0. Since these HDDs are mirrored, either should produce a current copy of the boot disk.
- All devices on the losing side should be quiesced, that is. no devices using losing side FRUs should be used by any applications.
- Disk configuration:
  - `rootdisk` is the SEVM representation of `a-dsk0`, which is the disk device `c2t0d0`;
  - `disk01` is the SEVM representation of `b-dsk0`, which is the disk device `c3t0d0`.

Initial Configuration

Machine: 2P 512Mb Netra ft 1800

OpenBoot PROM version: release 24 (part no. 258-7354-10)

FPGA version: release 19 (part no. 258-7134-07)

Operating System: Netra ft 1800 Solaris 6.7 CD, supplemental CD , patches CD, SEVM 2.5 (including patch 105463)

Patch Level: Patch D-10

FRUs present: HDD in `A-DSK0` (`/dev/rdsk/c2t0d0`) called `rootdisk` in SEVM, mirrored in `B-DSK0` (`/dev/rdsk/c3t0d0`) called `disk01` in SEVM. No other HDDs are present in the machine. No PCI devices are attached.

Node name: `seagoon` (IP address 129.156.203.152)

Initial configuration (`cmsconfig`): As shown in FIGURE E-1

```
Item  Name          Fault Loc     State                 Page 1 of 1
-------------------------------------------------------------------------
0     A-MBD 0           A-MBD     enabled
1     B-MBD 0           B-MBD     enabled
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     enabled
8     HDD 0             A-DSK0    enabled
9     HDD 7             B-DSK0    enabled
10    ft_alarm 0                  usable
11    ft_core 0                   enabled     Fault tolerant
12    ft_network 0                online_up   A (online) & B (online)
13    ft_serial 0                 online
14    icn_system 0                disabled    System is not split

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

FIGURE E-1 Initial CMS Configuration

Notice the presence of:

- two motherboards (MBDs)
- two CAFs
- two CPUs
- two disk chassis (DSKs)

- two hard disks (HDDs)
- an `ft_alarm` (the alarms subsystem)
- an `ft_core` (the fault-tolerant virtual machine)
- `ft_network 0` (a fault-tolerant network)
- `ft_serial` (the serial subsystem)
- `icn_system`

Check that the system (`seagoon`) is in combined mode using the command:

```
seagoon# splitinfo -a
attributes=combined
```

Ensure that the Netra ft 1800 combined system is not acting as a router. This is accomplished by the existence of the file `/etc/notrouter`, which can be created by issuing the command:

```
seagoon# touch /etc/notrouter
```

Configuring for Split Mode

This example shows how to split a Netra ft 1800 with nodename `seagoon` into `seagoon` and `seagoon-2`. While `seagoon` is left running on side A, `seagoon-2` will be updated on side B, and then the two sides merged to reform a combined, or fault-tolerant, Netra ft 1800 system.

You may find it useful to have two xterms open, one for each of the following:

- A-side console, to monitor the progress of the split on the A-side
- B-side console, to monitor the progress of the split on the B-side

The following steps are required to prepare to split a system.

1. Edit the `splitd.conf` file
2. Set up ICN
3. Set the split master
4. Add other IP addresses to `/etc/hosts`
5. Configure `ft_network 0`
6. Check SEVM status
7. Check CPUs are in sync

8. Check split daemon status

9. Turn off boot disk checksumming

Editing the `split.conf` File

FIGURE E-2 shows the contents of `/etc/splitd.conf`, naming `seagoon` and `seagoon-2` as the two host names to be used following the split.

```
# 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 seagoon-i0 seagoon-2-i0
host_alt seagoon seagoon-2

# 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 seagoon seagoon-2
```

FIGURE E-2 The `/etc/splitd.conf` File

Setting Up ICN

Use `cmsconfig` to set the *hostname* entry in `icn0` to the IP address of the split winner's ICN network. In this case, *hostname* is set to `seagoon-i0`

```
Select: icn 0

Item  Name                          Value                  Page 1 of 1
-------------------------------------------------------------------------
0     state                         not_present
1     icn_cmd                       disable
2     description                   ICN controller
3     hostname                      seagoon-i0
4     interface_cmd                 up
5     user_label
6     memory_size                   524288
7     descriptor_limit              255
8     info
9     software_fault                no
10    interface_state               unknown
11    busylock                      no
12    action                        disable

(H)elp, (Number)to modify, (S)elect, (T)op or (Q)uit ?
```

Setting the Split Master

The split master negotiates the transfer of ownership of FRUs from one side to another. Only one side of a Netra ft 1800 can be the master, and this can be changed in combined or split mode. In this case, the following command is used to change the master to side A, which also happens to be the split winner:

```
seagoon# splitconf -m A
master = A
```

Note – The master does not have to be the same as the split winner or merge winner. The split or merge winner is the side that continues running during the split or merge operation, respectively, whereas the master negotiates FRU ownership transfers between sides.

Adding Other IP Addresses to /etc/hosts

Here, entries for both seagoon-2, and host names for both side's ICN 0 network instances have been added to the /etc/hosts file:

```
#
# Internet host table
#
127.0.0.1       localhost       loghost
129.156.203.152 seagoon
129.156.203.18  seagoon-2
192.168.1.1     seagoon-i0
192.168.1.2     seagoon-2-i0
```

In addition, localhost has been set as the loghost to ensure that logging is performed on the local machine after a split. Otherwise, logging would be directed across the network to the other side of the Netra ft 1800.

Configuring ft_network 0

Check that ft_network 0 is operating in fault-tolerant mode, that is, both A and B controllers are online.

```
Select: ft_network 0

Item  Name                        Value                  Page 1 of 1
------------------------------------------------------------------------
0     state                       online_up
1     description                 network multiplexor
2     user_label
3     hostname                    seagoon
4     preferred_controller        none
5     controllerA_FRU             A-CAF
6     controllerA_Funct           Net_0
7     controllerB_FRU             B-CAF
8     controllerB_Funct           Net_0
9     info
10    busylock                    no
11    devpath                     pnet0
12    link                        100 Mbps half-duplex link up
13    transceiver                 internal transceiver selected
14    usable_controllers          A (online) & B (online)
15    controller_in_use           A
```

The split mode software ensures that when the losing (B) side is rebooted after the split, it will (in this example) take on the host name `seagoon-2`.

Any other configured ft_networks should set to NULL any controllers using interfaces on the losing side. This will prevent the split loser from bringing online networks with host names that clash with host names on the split winner.

Checking the SEVM State

Check the state of the SEVM by issuing the following commands:

```
seagoon# ps -ef -oargs | grep vx
vxconfigd -m boot

seagoon# vxdctl mode
mode: enabled

seagoon# vxdisk list
DEVICE       TYPE      DISK         GROUP        STATUS
c2t0d0s2     sliced    rootdisk     rootdg       online
c3t0d0s2     sliced    disk01       rootdg       online
```

The first command ensures that the SEVM configuration daemon is running.

The second command ensures that the daemon in enabled.

The third command shows the status of the HDDs. In this case there are two HDDs under SEVM control, `rootdisk` and `disk01`, which are both members of the rootdg `disk` group. This SEVM HDD configuration is necessary for split mode to work correctly.

In addition, issue the `vxprint` command to determine the redundant state of SEVM as shown in CODE EXAMPLE E-1:

CODE EXAMPLE E-1 StorEdge Volume Manager State

```
seagoon# vxprint
Disk group: rootdg

TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg rootdg       rootdg       -        -        -        -        -       -

dm disk01       c3t0d0s2     -        17538444 -        -        -       -
dm rootdisk     c2t0d0s2     -        17538444 -        -        -       -

v  opt          fsgen        ENABLED  4097331  -        ACTIVE   -       -
```

CODE EXAMPLE E-1 StorEdge Volume Manager State *(Continued)*

```
pl opt-01       opt          ENABLED  4097331  -        ACTIVE   -       -
sd rootdisk-04  opt-01       ENABLED  4097331  0        -        -       -
pl opt-02       opt          ENABLED  4097331  -        ACTIVE   -       -
 sd disk01-01   opt-02       ENABLED  4097331  0        -        -       -

v  rootvol      root         ENABLED  1106028  -        ACTIVE   -       -
pl rootvol-01   rootvol       ENABLED  1106028  -        ACTIVE   -       -
sd rootdisk-B0  rootvol-01    ENABLED  1        0        -        -       Block0
sd rootdisk-02  rootvol-01    ENABLED  1106027  1        -        -       -
pl rootvol-02   rootvol       ENABLED  1106028  -        ACTIVE   -       -
sd disk01-02    rootvol-02    ENABLED  1106028  0        -        -       -

v  swapvol      swap         ENABLED  1052163  -        ACTIVE   -       -
pl swapvol-01   swapvol       ENABLED  1052163  -        ACTIVE   -       -
sd rootdisk-01  swapvol-01    ENABLED  1052163  0        -        -       -
pl swapvol-02   swapvol       ENABLED  1052163  -        ACTIVE   -       -
sd disk01-03    swapvol-02    ENABLED  1052163  0        -        -       -

v  usr          fsgen        ENABLED  7185591  -        ACTIVE   -       -
pl usr-01       usr          ENABLED  7185591  -        ACTIVE   -       -
sd rootdisk-05  usr-01       ENABLED  7185591  0        -        -       -
pl usr-02       usr          ENABLED  7185591  -        ACTIVE   -       -
sd disk01-04    usr-02       ENABLED  7185591  0        -        -       -

v  var          fsgen        ENABLED  4097331  -        ACTIVE   -       -
pl var-01       var          ENABLED  4097331  -        ACTIVE   -       -
sd rootdisk-03  var-01       ENABLED  4097331  0        -        -       -
pl var-02       var          ENABLED  4097331  -        ACTIVE   -       -
sd disk01-05    var-02       ENABLED  4097331  0        -        -       -
```

Notice that there are five volumes in the `rootdg` disk group, each containing two plexes. All plex KSTATE entries should be in the `enabled` state, and all plex STATE entries should be in the `active` state. These states are required in order for split mode to succeed. Recover any plexes that are not in these states so that they are in the `active` state.

Confirming the CPUs Are In Sync

Enter the command `cmsconfig`:

```
ps -elf -oargs | grep u4ftsplitd cmsconfig

Item  Name          Fault Loc     State                  Page 1 of 1
-------------------------------------------------------------------------
0     A-MBD 0             A-MBD   enabled
1     B-MBD 0             B-MBD   enabled
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   enabled
8     HDD 0               A-DSK0  enabled
9     HDD 7               B-DSK0  enabled
10    ft_alarm 0                  usable
11    ft_core 0                   enabled    Fault tolerant
12    ft_network 0                online_up  A (online) & B (online)
13    ft_serial 0                 online
14    icn_system 0                disabled   System is not split

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

Notice that the `ft_core` object reads `Fault tolerant`, indicating that the CPUs are in sync.

Checking the Split Daemon Status

Ensure that the split daemon, `u4ftsplitd`, is running by issuing the command:

```
seagoon# ps -elf -oargs | grep u4ftsplitd
/usr/platform/SUNW,Ultra-4FT/SUNWcms/lib/u4ftsplitd
```

and noting that the output contains u4ftsplitd. If it is not running, restart it with the command:

```
seagoon# /etc/init.d/u4ftsplit start
```

Turning off Boot Disk Checksumming

With patch D installed, it is necessary to turn off disk checksumming on the loser's boot disk to avoid SEVM errors swamping the split loser's console after a split. To do this, use `cmsconfig` to turn off checksumming in the `B-DSK0` FRU.

Quick Checklist

At this stage, confirm the following before proceeding:

- The CPUs are running in sync.
- There are HDDs present in `A-DSK 0` and `B-DSK 0`, which are mirrored using SEVM. In addition, these two HDDs are both members of the SEVM disk group `rootdg`. The SEVM configuration contains exactly *one* disk group (rootdg) containing *one* mirrored pair.
- `/etc/splitd.conf` names both `seagoon` and `seagoon-2,` and at least one of the ICN networks.
- `/etc/hosts` contains all IP addresses (for `seagoon`, `seagoon-2` and the ICN networks configured in `/etc/splitd.conf`).
- The `ft_network 0` is configured in fault-tolerant mode.
- For each ICN network instance configured in `/etc/splitd.conf`, the `hostname` field of the appropriate object in `cmsconfig` should be set. For example, if ICN network instance 0 is configured, the host name for ICN 0 in `cmsconfig` should be set.
- The boot disks are mirrored and the mirroring process has completed. A fully-mirrored volume has an ACTIVE state for the volume and plex fields in the output from the `vxprint` command.

Splitting the System

You are now ready to split the system, but, before proceeding, note the following caution:

Caution – Do not use motherboard resets (`AAres` and `BBres`) or any other mechanism in which the motherboard is powered off, or perform a power cycle or an operation that results in a power cycle, on either side when in split mode. Doing so will result in undesirable failures occurring in certain FRUs on the other side. This is because some resources on a motherboard may belong to the other side and these resources will be removed from service if the motherboard is reset.

If the system is split naturally, that is, if all the resources residing on the side-A motherboard belong to side A and all resources residing on side-B motherboard belong to side B, or if the resources on the motherboard that needs to be reset are not deemed to be critical for providing service to the clients of the other side, it is possible to use a motherboard reset.

However, to avoid failure of the bridges, which will cause subsequent merge operations to fail, in the CMS first disable the motherboard that is to be reset on the side the side that is *not* going to be reset. this will prevent bridge failures and will enable the system to merge successfully when it is brought back into fault-tolerant operation.

Issue the `split` command from either side:

```
splitadm -w a split
```

Note – This will cause many warning messages from SEVM as it loses sight of its mirror.

FIGURE E-3 shows the output on the A-side console after the `split` command has been issued from the A-side.

```
seagoon# splitadm -w a split
NOTICE: vxdmp: Path failure on 32/124
WARNING: vxvm:vxio: error on Plex var-02 while writing volume var offset 119388
length 2
WARNING: vxvm:vxio: Plex var-02 detached from volume var
WARNING: vxvm:vxio: disk01-05 Subdisk failed in plex var-02 in vol var
vxvm:vxconfigd: NOTICE: Offlining config copy 1 on disk c3t0d0s2:
        Reason: Disk write failure
vxvm:vxconfigd: NOTICE: Detached disk disk01
vxvm:vxconfigd: NOTICE: Detached plex opt-02 in volume opt
vxvm:vxconfigd: NOTICE: Detached plex rootvol-02 in volume rootvol
vxvm:vxconfigd: NOTICE: Detached plex swapvol-02 in volume swapvol
vxvm:vxconfigd: NOTICE: Detached plex usr-02 in volume usr
vxvm:vxconfigd: NOTICE: Detached plex var-02 in volume var
vxvm:vxassist: ERROR: Cannot allocate space to replace subdisks
vxvm:vxassist: ERROR: Cannot allocate space to mirror 1106028 block volume
vxvm:vxassist: ERROR: Cannot allocate space to mirror 1052163 block volume
vxvm:vxassist: ERROR: Cannot allocate space to replace subdisks
vxvm:vxassist: ERROR: Cannot allocate space to replace subdisks
icn network setupWARNING: Out-of-sync on CPUset B

 (via CMS): icn.keepalive done.
domain = A
attributes = split,master
master = A
```

FIGURE E-3 A-Side Console After `split` Command

The B-side console display is shown in FIGURE E-4, which is a copy of the A-side console output before the split causes a reboot of the B-side.

```
NOTICE: vxdmp: Path failure on 32/124
WARNING: vxvm:vxio: error on Plex var-02 while writing volume var offset 119388
length 2
WARNING: vxvm:vxio: Plex var-02 detached from volume var
WARNING: vxvm:vxio: disk01-05 Subdisk failed in plex var-02 in vol var
vxvm:vxconfigd: NOTICE: Offlining config copy 1 on disk c3t0d0s2:
        Reason: Disk write failure
vxvm:vxconfigd: NOTICE: Detached disk disk01
vxvm:vxconfigd: NOTICE: Detached plex opt-02 in volume opt
vxvm:vxconfigd: NOTICE: Detached plex rootvol-02 in volume rootvol
vxvm:vxconfigd: NOTICE: Detached plex swapvol-02 in volume swapvol
vxvm:vxconfigd: NOTICE: Detached plex usr-02 in volume usr
vxvm:vxconfigd: NOTICE: Detached plex var-02 in volume var
```

FIGURE E-4 B-Side Console After `split` Command

Assuming all has gone well, check that the winning host is running on side A:

```
seagoon# u4ftvmctl -c
This is CPUset_A (0)
```

Issuing the `cmsconfig` command on `seagoon` should display information similar that in FIGURE E-5:

```
seagoon# cmsconfig

Item   Name         Fault  Loc     State                     Page 1 of 1
-------------------------------------------------------------------------
0      A-MBD 0             A-MBD   enabled
1      B-MBD 0             B-MBD   enabled
2      CAF 0               A-CAF   enabled
3      CAF 1               B-CAF   disabled
4      CPU 0               A-CPU   enabled
5      CPU 1            X  B-CPU   enable_failed   Could not disable: busy
6      DSK 0               A-DSK   enabled
7      DSK 1               B-DSK   disabled
8      HDD 0               A-DSK0  enabled
9      HDD 7               B-DSK0  disabled
10     ft_alarm 0                  usable
11     ft_core 0                   enabled         Running on A-CPU
12     ft_network 0                online_up       A (online)
13     ft_serial 0                 online          2 (unusable)
14     icn 0                       enabled
15     icn 1                       disabled
16     icn 2                       disabled
17     icn 3                       disabled
18     icn_system 0                enabled

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

FIGURE E-5 CMS Configuration After Split

Note that all the B-side FRUs, apart from the B-side motherboard, are in a disabled state.

Note – With patch D installed, the state of the B-CPU is reported incorrectly. Use `cmsconfig` to bring it to a disabled state.

Bringing Up the Losing Side

The losing (B) side will attempt a reboot from B-DSK0, finally resulting in a prompt to enter SEVM license keys:

CODE EXAMPLE 5-1 B-Side Reboot

```
Sun Ultra 4FT UPA/PCI(2 X UltraSPARC-II 296MHz), No keyboard
OpenBoot 3.7 [PROTO-P1b-sd_st: Fusion-B2], 256 MB memory installed, Serial #9539444.
Ethernet address 8:0:20:91:8f:74, Host ID: 80918f74.

Running preboot tests:                  SUCCESS
Boot device: b-dsk0  File and args:
Loading: ufs-file-system package 1.4 04 Aug 1995 13:02:54
Fcode UFS Reader 1.10 96/10/15 00:57:29
Loading: /platform/SUNW,Ultra-4FT/ufsboot
Loading: /platform/sun4u/ufsboot
SunOS Release 5.6 Version 108145-10 [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1997, Sun Microsystems, Inc.
WARNING: forceload of drv/scsi failed
NOTICE: Ultra-4FT DDI extensions installed
WARNING: forceload of drv/scsi failed
WARNING: forceload of drv/ssd failed
WARNING: forceload of drv/sf failed
WARNING: forceload of drv/pln failed
WARNING: forceload of drv/soc failed
WARNING: forceload of drv/socal failed
}rconsconfig: device at /pci@6,2000/u4ftbus@0/u4ioslot@4/ebus@4/se@14,400000:b is
inaccessible.
rconsconfig: device at /pci@6,2000/u4ftbus@0/u4ioslot@4/ebus@4/se@14,400000:a is
inaccessible.
VxVM starting in boot mode...
vxvm:vxconfigd: WARNING: Detaching plex rootvol-01 from volume rootvol
vxvm:vxconfigd: WARNING: Detaching plex usr-01 from volume usr
vxvm:vxconfigd: WARNING: Disk rootdisk in group rootdg: Disk device not found
vxvm: NOTE: Setting partition /dev/dsk/c3t0d0s1 as the dump device.
VxVM starting special volumes ( swapvol var )...
dumpvp_setup: Setting partition /dev/dsk/c3t0d0s1 as the dump device.
The / file system (/dev/vx/rdsk/rootvol) is being checked.
/dev/vx/rdsk/rootvol: 3023 files, 77365 used, 441552 free
/dev/vx/rdsk/rootvol: (336 frags, 55152 blocks,  0.0% fragmentation)
The /usr file system (/dev/vx/rdsk/usr) is being checked.
/dev/vx/rdsk/usr: 25591 files, 436045 used, 3099937 free
/dev/vx/rdsk/usr: (2785 frags, 387144 blocks,  0.0% fragmentation)
FIRST LOSER BOOT
YOU MUST ENTER THE VERITAS LICENCE KEYS HERE
Please enter your key: 7141 4164 7746 8669 8047 077

vrts:vxserial: INFO: Feature name: CURRSET [95]
```

CODE EXAMPLE 5-1 B-Side Reboot *(Continued)*

```
vrts:vxserial: INFO: Number of licenses: 1 (non-floating)
vrts:vxserial: INFO: Expiration date: Sun 14 Nov 1999 08:00:00 (33.9 days from now)
vrts:vxserial: INFO: Release Level: 20
vrts:vxserial: INFO: Machine Class: All
vrts:vxserial: INFO: Key successfully installed in /etc/vx/elm/95.
Please enter your key: 8919 6476 4493 0199 2931 280

vrts:vxserial: INFO: Feature name: RAID [96]
vrts:vxserial: INFO: Number of licenses: 1 (non-floating)
vrts:vxserial: INFO: Expiration date: Sun 14 Nov 1999 08:00:00 (33.9 days from now)
vrts:vxserial: INFO: Release Level: 20
vrts:vxserial: INFO: Machine Class: All
vrts:vxserial: INFO: Key successfully installed in /etc/vx/elm/96.
Licensed features:
 Mirroring
 Concatenation
 Disk-spanning
 Striping
 RAID-5
*** REBOOT THE SYSTEM ***
syncing file systems... done
rebooting...
Resetting ...
```

This will be followed by a second reboot of side B.

When side B has successfully rebooted, you can log into both sides. The winning side (A) retains the identity of `seagoon`, and the losing side (B) has the new identity `seagoon-2`.

Check that the system is actually in split mode by issuing the command `splitinfo` from both sides:

```
seagoon# splitinfo -d -a
domain = A
attributes = split,master
master = A
```

```
seagoon-2# splitinfo -d -a
domain = B
attributes = split
master = A
```

Issuing `cmsconfig` on side B (`seagoon-2`) should now produce the following:

```
seagoon# cmsconfig

Item  Name         Fault  Loc     State                     Page 1 of 1
-------------------------------------------------------------------------
0     A-MBD 0             A-MBD   enabled
1     B-MBD 0             B-MBD   enabled
2     CAF 0           X   A-CAF   enable_failed   Unable to power on FRU
3     CAF 1               B-CAF   enabled
4     CPU 0           X   A-CPU   enable_failed   Unable to power on FRU
5     CPU 1               B-CPU   enabled
6     DSK 0           X   A-DSK   enable_failed   Unable to power on FRU
7     DSK 1               B-DSK   enabled
8     HDD 0               A-DSK0  enable_failed   Cannot be enabled until DSK 0
9     HDD 7               B-DSK0  enabled
10    ft_alarm 0                  usable
11    ft_core 0                   enabled         Running on B-CPU
12    ft_network 0                online_up       B (online)
13    ft_serial 0                 online          1 (unusable) 0 (unusable)
14    icn 0                       enabled
15    icn 1                       disabled
16    icn 2                       disabled
17    icn 3                       disabled
18    icn_system 0                enabled

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

Note – When running patch D, the losing (A) side FRUs are recorded as failed and are in an `enable_failed` state. In fact, the FRUs are not faulty and you should use cmsconfig to bring them to a `disabled` state.

Note that the `icn_system` is enabled. To ensure that the sides are correctly communicating, issue the following commands on both sides:

```
seagoon# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
     inet 127.0.0.1 netmask ff000000
pnet0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
     inet 129.156.203.152 netmask ffffff00 broadcast 129.156.203.255
     ether 8:0:20:91:8f:54
icn0: flags=8843<UP,BROADCAST,RUNNING,MULTICAST,PRIVATE> mtu 40945
     inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
     ether 8:0:20:91:0:52
```

```
seagoon# u4ftctl -d /dev/icn status
(icn#0): Online + Exporting + Importing
(icn#1): Not initialised
(icn#2): Not initialised
(icn#3): Not initialised
```

```
seagoon-2# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
     inet 127.0.0.1 netmask ff000000
pnet0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
     inet 129.156.203.18 netmask ffffff00 broadcast 129.156.203.255
     ether 8:0:20:91:8f:76
icn0: flags=8843<UP,BROADCAST,RUNNING,MULTICAST,PRIVATE> mtu 40945
     inet 192.168.1.2 netmask ffffff00 broadcast 192.168.1.255
     ether 8:0:20:91:0:74
```

```
seagoon-2# u4ftctl -d /dev/icn status
(icn#0): Online + Exporting + Importing
(icn#1): Not initialised
(icn#2): Not initialised
(icn#3): Not initialised
```

Note that `icn0` should have `inet` set to 192.168.1.1 on seagoon and 192.168.1.2 on `seagoon-2`. The netmasks should both be `ffffff00`, that is, 255.255.255.0.

Note also that the `u4ftctl` command should show any ICN networks in an `Online+Exporting+Importing` state. If they are not, disable the `icn_system` FRUs on both sides and then re-enable them. Check the output of the above commands again. If the ICN network instances are still not in the `Online+Exporting+Importing` state, stop and restart the split daemon by issuing the command:

```
seagoon# /etc/init.d/e4ftsplit stop
```

This will stop and restart the split daemon.

Check that the split daemon is running on each side:

```
seagoon# ps -efo args | grep u4ftsplitd
/usr/platform/SUNW,Ultra-4FT/SUNWcms/lib/u4ftsplitd
```

```
seagoon-2# ps -efo args | grep u4ftsplitd
/usr/platform/SUNW,Ultra-4FT/SUNWcms/lib/u4ftsplitd
```

If the daemon is not running, start it using the `u4ftsplit` command:

```
seagoon# /etc/init.d/u4ftsplit start
```

```
seagoon-2# /etc/init.d/u4ftsplit start
```

Updating `seagoon-2`

You can now make changes to seagoon-2 as required. In this example, a few files are created on seagoon-2 that should survive the merge.

```
seagoon-2# touch /toto /usr/toto /var/toto /opt/toto /etc/toto
```

You can also install a new operating system, new packages and patches, or upgrade an application on the split loser.

Merging the Systems

The steps required for merging the two systems are as follows:

1. Check the split daemons on both sides are communicating with each other.
2. Issue the `merge` command.
3. Bring the losing side FRUs back online.
4. Wait for the CPUsets to recombine.
5. Remirror the new boot disk.

Confirming the Split Daemons are Communicating

Before a merge can occur, the split daemons on the two sides of the machine must be communicating with each other over the chosen network (set up in `/etc/splitd.conf`), which is typically ICN. Confirm this by attempting to change split mastership on both sides:

```
seagoon# splitconf -m <side>
master = A
```

<*side*> is dependent on the existing master: if the existing master is A, <*side*> should be B; if the existing master is B, <*side*> should be A.

- If the change of mastership proceeds without producing an error and the `splitinfo` command shows <*side*> as the master on both sides, the split daemons are communicating.
- If this is not the case, stop the split daemon on each side using the command **`/etc/init.d/u4ftsplit stop`**, which should result in the split daemons stopping and then restarting. If necessary, recheck split daemon communication using the above procedure.

Issuing the `merge` Command

You can issue the `merge` command from either side. In this case, it is issued from the merge winner:

```
seagoon-2# splitadm -w B merge
icn network shutdown (via CMS): icn.keepalive done.
domain = C
attributes = combined
master = A
```

This will force the losing side (A) to reboot. and the side A console displays the following:

```
icn network shutdown (via CMS): icn.keepalive 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
Killed download daemon.
Oct 11 11:26:33 snmpdx: received signal 15
The system is down.

INIT: failed write of utmpx entry:"s6"

INIT: failed write of utmpx entry:"rb"
syncing file systems... done
rebooting...
Resetting ...
```

At this point the side A console may go dead because the newly-combined system has the A-CAF disabled.

Waiting for the CPUsets to Become Fault Tolerant

Check by running `cmsconfig` and noting that *ft_core* has the value `Fault tolerant`.

Bringing the Losing Side FRUs Back Online

Issue the `cmsconfig` command on the merge winner (`seagoon-2`, side B):

```
seagoon# cmsconfig

Item  Name         Fault  Loc     State                      Page 1 of 1
-------------------------------------------------------------------------
0     A-MBD 0             A-MBD   enabled
1     B-MBD 0             B-MBD   enabled
2     CAF 0               A-CAF   disabled        WARNING: Power off command
3     CAF 1               B-CAF   enabled
4     CPU 0               A-CPU   busy
5     CPU 1               B-CPU   enabled
6     DSK 0               A-DSK   disabled        WARNING: Power off command
7     DSK 1               B-DSK   enabled
8     HDD 0               A-DSK0  disabled        WARNING: Power off command
9     HDD 7               B-DSK0  enabled
10    ft_alarm 0                  usable
11    ft_core 0                   enabled         Waiting for A-CPU to be ready
12    ft_network 0                online_up       B (online)
13    ft_serial 0                 online          0 (unusable)
14    icn_system 0                disabled

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

Re-enable merge-losing side FRUs (in this case, A-CAF, A-DSK and A-DSK0). After re-enabling the A-CAF, the A-side console should return.

Remirroring the New Boot Disk

After the merge, you can check the state of the SEVM using the `vxdisk list` command, as follows:

```
seagoon-2# vxdisk list
 DEVICE       TYPE      DISK         GROUP        STATUS
 c2t0d0s2     sliced    -            -            online
 c3t0d0s2     sliced    disk01       rootdg       online
 -            -         rootdisk     rootdg       failed was:c2t0d0s2
```

The boot disk used by `seagoon-2`, containing the updated operating system, is `c3t0d0`, known to SEVM as `disk01`. The original disk used by `seagoon` before the upgrade is `c2t0d0`, known to SEVM as `rootdisk`, which has been failed by SEVM.

The merge process results in SEVM losing sight of `rootdisk` because the CMS tells it that it is disabled (which occurs because the merged system inherits its CMS state from the merge winner in which `rootdisk` was disabled).

The procedure now is to inform SEVM that `rootdisk` is still present in the Netra ft 1800. To do this, use the `vxdiskadm` command, main menu option 5 (Replace a failed or removed disk). The sequence of screens is as follows.

```
seagoon-2# vxdiskadm

Volume Manager Support Operations
Menu: VolumeManager/Disk

 1      Add or initialize one or more disks
 2      Encapsulate one or more disks
 3      Remove a disk
 4      Remove a disk for replacement
 5      Replace a failed or removed disk
 6      Mirror volumes on a disk
 7      Move volumes from a disk
 8      Enable access to (import) a disk group
 9      Remove access to (deport) a disk group
 10     Enable (online) a disk device
 11     Disable (offline) a disk device
 12     Mark a disk as a spare for a disk group
 13     Turn off the spare flag on a disk
 list   List disk information

 ?      Display help about menu
 ??     Display help about the menuing system
 q      Exit from menus

Select an operation to perform: 5
```

```
Replace a failed or removed disk
Menu: VolumeManager/Disk/ReplaceDisk

 Use this menu operation to specify a replacement disk for a disk
 that you removed with the "Remove a disk for replacement" menu
 operation, or that failed during use.  You will be prompted for
 a disk name to replace and a disk device to use as a replacement.
 You can choose an uninitialized disk, in which case the disk will
 be initialized, or you can choose a disk that you have already
 initialized using the Add or initialize a disk menu operation.

Select a removed or failed disk [<disk>,list,q,?] list
```

```
Disk group: rootdg

DM NAME       DEVICE       TYPE     PRIVLEN  PUBLEN   STATE

dm rootdisk   -            -        -        -        NODEVICE

Select a removed or failed disk [<disk>,list,q,?] rootdisk

Select disk device to initialize [<address>,list,q,?] list
```

```
DEVICE       DISK         GROUP        STATUS
c2t0d0       -            -            online
c3t0d0       disk01       rootdg       online

Select disk device to initialize [<address>,list,q,?] c2t0d0
```

```
This disk device is currently listed as in use by another host.
If you are certain that the other host is not using the disk, you
can choose to clear the use status. To use the disk the use status
must be cleared.
Output format: [Device_Name,Disk_Access_Name,Hostid]

[c2t0d0,c2t0d0s2,seagoon]

Clear use status? [y,n,q,?] (default: n) y
```

```
The following disk you selected for use appears to already have
been initialized for the Volume Manager.  If you are certain the
disk has already been initialized for the Volume Manager, then you
do not need to reinitialize the disk device.
Output format: [Device_Name]

c2t0d0

Reinitialize this device? [y,n,q,?] (default: y) y
```

```
The requested operation is to initialize disk device c2t0d0 and
to then use that device to replace the removed or failed disk rootdisk in disk
group rootdg.

Continue with the operation? {y,n,q,?] (default: y) y
```

```
Replacement of disk rootdisk in group rootdg with disk device
c2t0d0 completed successfully.

Replace another disk? [y,n,q,?] (default: n) n
```

Check the state of SEVM using the `vxdisk` list command:

```
seagoon-2# vxdisk list
DEVICE       TYPE      DISK         GROUP        STATUS
c2t0d0s2     sliced    rootdisk     rootdg       online
c3t0d0s2     sliced    disk01       rootdg       online
```

Note that there are two devices, both of which are online and mapped to two SEVM disks. At this point, SEVM knows about `c2t0d0` and has assigned `rootdisk` to it, as before. The mirroring from `disk01` to `rootdisk` can now proceed.

Check for the existence of a process `vxrecover`, which resynchronizes the boot disk mirror:

```
# rip# ps -ef | grep recover
root  9764  4250  0 17:13:10 console  0:00 grep recover
root  9173     1  0 17:12:00 ?        0:00 vxrecover -sb -g rootdg rootdisk
```

If it doesn't exist, execute it by issuing the command `vxrecover`. The progress of this remirroring can be followed using either `vxprint`, or the X-windows graphical interface, `vxva`.

If SEVM is unable to find the disk device `c2t0d0` for `rootdisk`, add the disk and initialize it before you select option 4.-

This is achieved by selecting option 1 from the main menu of `vxdiskadm`

Index

E

F

G

H

I

K

M

N

O

P

Q

R

S

T

U

V