Sun™ Cluster 3.0 Advanced Diagnostics and Troubleshooting IES-SC33

Student Guide With Instructor Notes

Sun Microsystems, Inc.
UBRM05-104
500 Eldorado Blvd.
Broomfield, CO 80021
U.S.A.

July 2001, Revision B

Table of Contents

Module 1

Troubleshooting Cluster Installations

Objectives

This module details the tasks you must perform to install a Sun™ Cluster 3.0 cluster using SunPlex™ Manager or custom JumpStart™. In addition, the module provides tips for troubleshooting Sun Cluster 3.0 installations.

By the end of this module you should be able to:

- Install and configure a cluster with SunPlex Manager
- Address known issues with SunPlex Manager installations
- Install and configure a cluster with `scinstall`'s custom JumpStart option
- Address known issues with custom JumpStart
- Address known issues with `scinstall`

The goal of this module is to provide students with alternative installation methods in addition to the `scinstall` method covered in ES-333. Troubleshooting tips are also provided wherever possible. By the end of this module, the students should have a working cluster. Installation of the administrative workstations is covered in ES-333 and should not be covered in detail here. Instead, install the administrative consoles in advance of the class to save class time.

Be sure to read the `README.setup_instructions` file in the course lab files directory for complete classroom setup instructions.

This module can also serve as a review to `scinstall` since the process is covered in the JumpStart section.

Additional Resources

The following references can provide additional details on the topics discussed in this module:

- *Sun Cluster 3.0 U1 Installation Guide*, 806-7069-10
- *Sun Cluster 3.0 U1 Release Notes*, 806-7078-10
- *Sun Cluster 3.0 U1 Release Notes Supplement*
- *Sun Cluster 3.0 U1 Data Services Installation and Configuration Guide*, 806-7071-10
- *Solaris 8 Advanced Installation Guide*, 806-0957-10

These documents are available online at `http://docs.sun.com`.

Sun Cluster 3.0 Installation Methods

In the ES-333 Sun Cluster 3.0 Administration course, you learned how to install the Sun Cluster 3.0 software using the command-line utility, `scinstall`, and to install the Sun Cluster 3.0 software after installing Solaris. This process is largely manual and can be prone to errors.

There are two additional methods you can use to install the Sun Cluster 3.0 software:

- Install Solaris™ software, then install SunPlex Manager and use it to install the Sun Cluster software
- Install Solaris software and Sun Cluster software in one operation using the `scinstall` utility's custom JumpStart option

The following sections list the tasks you must perform to install a cluster using SunPlex Manager or custom JumpStart.

Note – Regardless of the installation method you select, you must plan your installation carefully and (optionally) install the Cluster Control Panel on a workstation that has access to your cluster nodes. This module does not cover these activities. If you need to review this material, refer to the ES-333 Sun Cluster 3.0 Administration course material or the *Sun Cluster 3.0 U1 Installation Guide.*

Installing a Cluster With SunPlex Manager

Overview of SunPlex Manager

SunPlex Manager is a tool that uses a Graphical User Interface (GUI) to display cluster information, monitor configuration changes, and check the status of cluster components.

SunPlex Manager also allows you to perform the following installation and configuration tasks:

- Install Sun Cluster 3.0 7/01 (Update 1) software on a new cluster
- Install Solstice DiskSuite software
- Configure up to three Solstice DiskSuite metasets and associated metadevices, and create and mount cluster file systems for each
- Install Sun Cluster HA for NFS data service
- Install Sun Cluster HA for Apache scalable data service

Currently, however, SunPlex Manager cannot perform all Sun Cluster installation or administrative tasks. You must still use the command-line interface for some operations.

Preparing for Cluster Installation with SunPlex Manager

Before using SunPlex Manager to install your cluster, install Solaris 8 and any associated patches. Refer to the *Configuration Guide* at: `http://suncluster.eng/products/SC3.0/config` for configuration guidelines and supported versions of the Solaris 8 Operating Environment.

Plan your installation following the guidelines in Chapter 1 of the *Sun Cluster 3.0 U1 Installation Guide*. Online, this guide is available at: `http://docs.sun.com`.

After you have completed your planning, follow the directions in Chapter 2 of the *Sun Cluster 3.0 U1 Installation Guide* to install the Solaris 8 Operating Environment on each of your cluster nodes.

Note – If you plan to use SunPlex Manager to install Solstice DiskSuite™ while installing Sun Cluster software, create at least a 10 Megabyte file system on slice 7 with a mount point of /sds.

Note – Set the root password to be identical on each node of the cluster.

Performing Cluster Installation with SunPlex Manager

Installing a cluster with SunPlex Manager consists of the following major tasks:

1. Install the SunPlex Manager software packages.
2. Install the cluster framework, Sun Cluster HA for NFS (optional), and Sun Cluster HA for Scalable Apache (optional) with SunPlex Manager.
3. Perform post-installation setup.
4. Install and configure volume manager software.
5. Configure cluster framework software.
6. Plan, install and configure resource groups and data services.

Installing the SunPlex Manager Software Packages

Perform this installation as follows:

1. Verify that Solaris Apache is installed on your cluster nodes:

```
# pkginfo SUNWapchr
# pkginfo SUNWapchu
# pkginfo SUNWapchd
```

2. If these packages are not installed, you must install them from the Solaris image:

```
# cd <path_to_Solaris_image>/Solaris_8/Product
# pkgadd -d . SUNWapchr SUNWapchu SUNWapchd
```

3. Install any Apache patches. See the *Early Notifier #24617* (`http://sunsolve.Central/cgi/retrieve.pl?doc=enotify/24617`), and Release Notes (`http://suncluster.eng`) for required patch details.

4. Change to the *<path_to_Update_1_Files>*`/SunCluster_3.0/Packages` directory:

   `# cd <`*`path_to_Update_1_Files`*`>/SunCluster_3.0/Packages`

5. Install the SunPlex Manager software packages and answer yes to all prompts:

   `# pkgadd -d . SUNWscva SUNWscvr SUNWscvw`

6. Repeat Steps 1 through 5 on each node of the cluster.

Installing the Cluster Framework and Optional Data Services with SunPlex Manager

One way to present the SunPlex Manager screens might be to “demo” an installation in the class rather than talking through it.

Perform the installation as follows:

1. Verify that SunPlex Manager is installed on each node of the cluster:

   `# pkginfo SUNWscva SUNWscvr SUNWscvw`

2. If you intend to install Sun Cluster HA for NFS, or Sun Cluster HA for Scalable Apache, verify that the cluster configuration meets the requirements in the following table.

Table 1-1 Sun Cluster HA for NFS and Scalable Apache Installation Requirements

Software Package	Installation Requirement
Sun Cluster HA for NFS Data Service	At least two shared disks of the same size that are connected to the same set of nodes. Solstice DiskSuite software will be installed by SunPlex Manager. A logical hostname for use by Sun Cluster HA for NFS. The logical hostname must have a valid IP address that is accessible by all cluster nodes and is on the same subnet as the base hostnames of the cluster nodes.

Table 1-1 Sun Cluster HA for NFS and Scalable Apache Installation Requirements

Software Package	Installation Requirement
Sun Cluster HA for Apache Scalable Data Service	At least two shared disks of the same size that are connected to the same set of nodes. Solstice DiskSuite software will be installed by SunPlex Manager. A shared address for use by Sun Cluster HA for Apache. The shared address must have a valid IP address that is accessible by all cluster nodes and is on the same subnet as the base hostnames of the cluster nodes.

Note – Make sure the shared address and/or logical hostname you plan to use is entered in the `/etc/hosts` file on the cluster node and that the `hosts` entry in `/etc/nsswitch.conf` is set to look for `files` first.

3. Prepare file system paths to a CD-ROM image of each software product you intend to install as follows:

   a. Provide each CD-ROM image in a location available to each node. The CD-ROM images must be accessible to all nodes of the cluster from the same file system path. These paths can lead to one or more of the following locations:

      - CD-ROM drives exported to the network from machines outside the cluster
      - Exported file systems on machines outside the cluster
      - CD-ROM images copied to local file systems on each node of the cluster. The local file system must use the same name on each node

   b. Record the path to each CD-ROM image.

4. If there are patches required for Sun Cluster or Solstice DiskSuite, you can install them manually or use SunPlex Manager to install the patch. If you wish to use SunPlex Manager to install the patches:

   a. Copy patches required for Sun Cluster or Solstice DiskSuite software into a single directory on a file system that is available to each node.

   b. Ensure that only one version of each patch is present in this patch directory.

c. Ensure that the patches are uncompressed.

d. Record the path to the patch directory.

5. Obtain and complete configuration planning worksheets from the *Sun Cluster 3.0 Release Notes* at: `http://docs.sun.com` or set up a copy of your own planning worksheets. The worksheets are labelled:

a. Cluster and Node Names Worksheet

Example: Cluster and Node Names

Cluster name: scclustet

Private network address: 172 . 16 . 0.0 (default: 172.16.0.0)

Private network mask: 255.255. 0 . 0 (default: 255.255.0.0)

Sponsor node name: phys-schost-1

Private hostname

Default name: clusternode 1 -priv (node ID) Change to: phys-schost-1-priv (optional)

Additional nodes

Node name: phys-schost-2 Private hostname Default name: clusternode 2 -priv (node ID) Change to: phys-schost-2-priv (optional)	Node name: _____ Private hostname Default name: clusternode_____-priv (node ID) Change to: _____ (optional)

Figure 1-1 Sample Cluster and Node Names Worksheet

b. Cluster Interconnect Worksheet

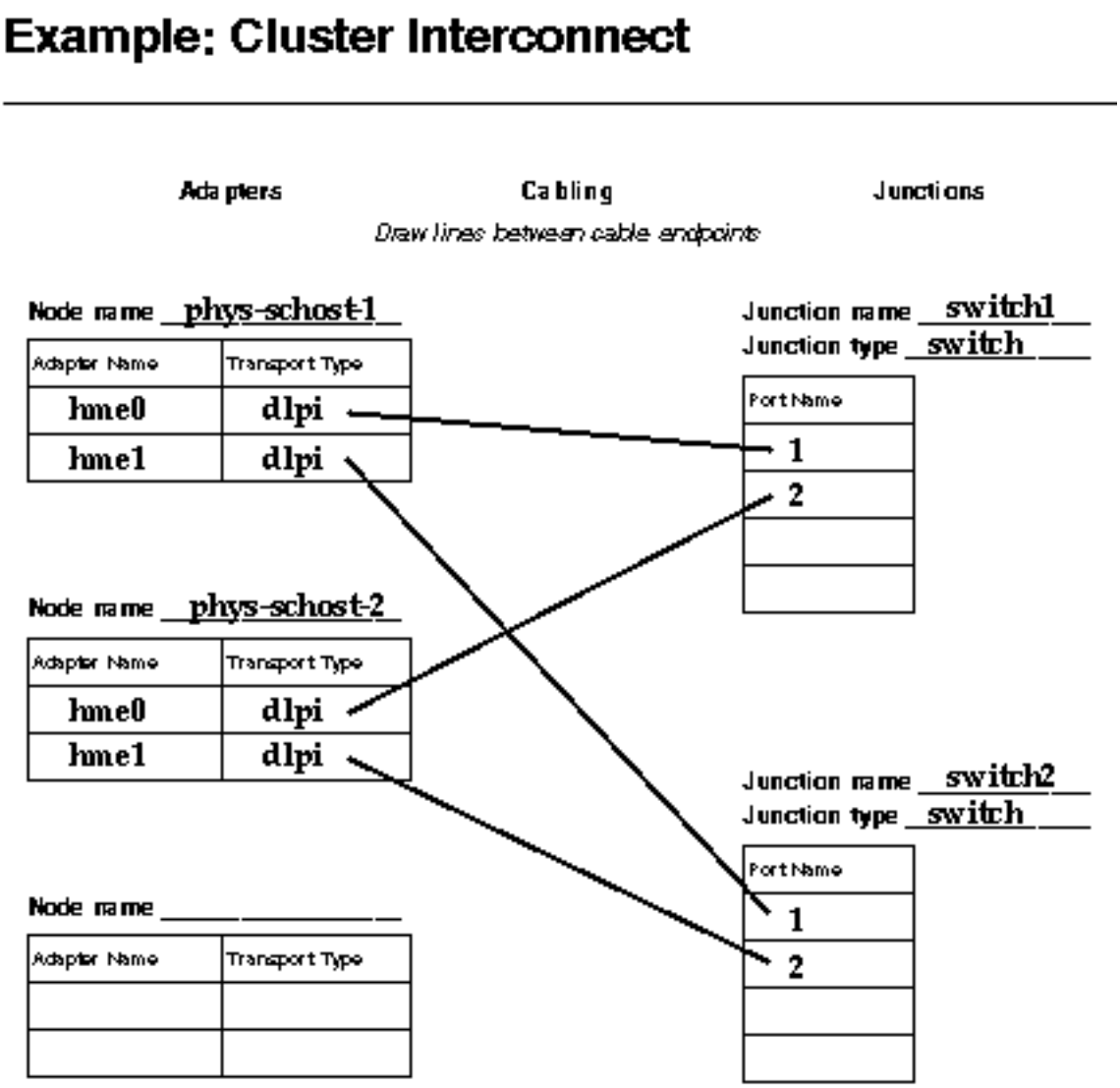

Figure 1-2 Sample Cluster Interconnect Worksheet

c. Network Resources Worksheet

Example: Network Resources--LogicalHostname

Resource name relo-galileo

Resource group name rg-oracle

Resourcetype:

☒ Logical hostname ☐ Shared address ☐ Data service/other

Hostnames used relo-galileo

Network name net-85

Adapter or NAFO group:

Node name	Adapter/NAFO group name
phys-schost-1	nafo0
phys-schost-2	nafo0

Resource type name

Dependencies

Extension properties:

Name	Value

Figure 1-3 Sample Network Resources Worksheet

6. Launch a browser from the administrative console, or any other machine outside the cluster.

7. Disable the browser's Web proxy (SunPlex Manager installation functionality is incompatible with Web proxies).

Note – If you are using Netscape, you can disable the proxy by selecting Edit / Preferences / Advanced / Proxies from the Netscape menu. Click Direct Connect to the Internet to disable the proxy.

8. Ensure disk caching and memory caching are enabled. The disk cache and memory cache size must be greater than 0.

Note – If you are using Netscape, you can access this information by selecting Edit / Preferences / Advanced / Cache from the Netscape menu.

9. From the browser, connect to port 3000 on one node of the cluster: `https://node:3000`. The Sun Cluster Installation screen displays in the browser window.

Note – If the browser displays a New Site Certification window, follow the onscreen instructions to accept the certificate.

10. Log in as user 'root' and enter the root password.

11. In the Sun Cluster Installation screen, verify that the cluster meets the listed requirements for using SunPlex Manager.

Figure 1-4 SunPlex Manager Installation Screen

12. If you meet all listed requirements, click Next to continue to the next screen.

13. Type a name for the cluster and select the number of nodes in your cluster.

Figure 1-5 SunPlex Manager Cluster Name Input Screen

14. Click Next to continue.

Note – You can use the Back button to return to a previous screen and change your information. However, SunPlex Manager does not save the information you supplied in the later screens. When you click Next, you must again type or select your configuration information in those screens.

15. Type the name of each cluster node. Click Next to continue.

Figure 1-6 SunPlex Manager Node Names Input Screen

16. From the pulldown lists for each node, select the names of the two adapters used for the private interconnects. Click Next to continue.

Figure 1-7 SunPlex Manager Interconnects Input Screen

17. Choose whether to install Solstice DiskSuite software. Click Next to continue.

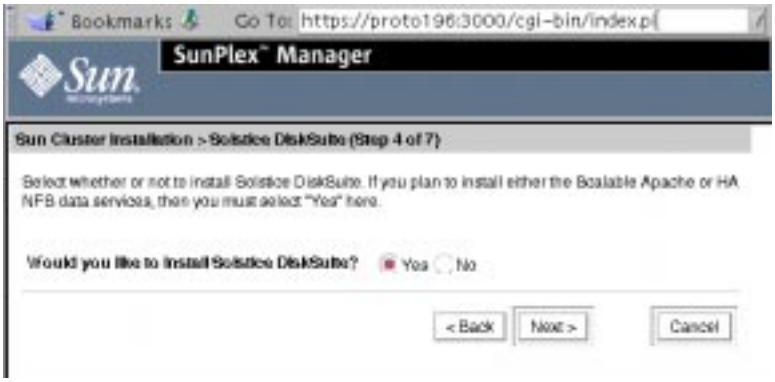

Figure 1-8 SunPlex Manager Solstice DiskSuite Input Screen

Warning – When Solstice DiskSuite is installed, any data on all shared disks will be lost.

18. Choose whether to install Sun Cluster HA for NFS, Sun Cluster HA for Apache, or both, as follows:

    a. For Sun Cluster HA for NFS, specify the logical hostname the data service will use.

b. For Sun Cluster HA for Apache, specify the shared address the data service will use.

c. Click Next to continue.

Figure 1-9 SunPlex Manager Data Services Input Screen

19. Type the path for each CD-ROM image needed to install the packages you specified. Optionally, type the path for the patch directory.

    a. Each specified path for a CD-ROM image must be to the directory containing the `.cdtoc` file for the CD-ROM.

    b. For any software package you do not install, leave the relevant path field blank.

c. If you have already installed the required patches or plan to install them manually, leave the Patch Directory Path field blank.

Figure 1-10 SunPlex Manager Image Paths Input Screen

20. Click Next to continue.

21. If the information you supplied is correct as displayed in the Confirm Information screen, proceed to Step 22.

Figure 1-11 SunPlex Manager Confirm Information Screen

If the information is not correct, perform the following steps to correct the configuration information:

a. Click Back until you return to the screen with the information to change.

Note – When you click Back to back up to a previous screen, any information you typed in the subsequent screens is lost.

b. Type the correct information and click Next.

c. Retype or reselect the information in each screen until you return to the Confirm Information screen.

d. Verify that the information in the Confirm Information screen is now correct.

22. Click Begin Installation to start the installation process.

a. If the browser displays a New Site Certification window, follow the onscreen instructions to accept the certificate.

b. If the browser prompts for login information, type 'root' and the root password.

Warning – Do not close the browser window or change the URL during the installation process.

During installation, the screen displays brief messages about the status of the cluster installation process. When installation is complete, the browser displays the cluster monitoring and administration GUI.

Note – SunPlex Manager installation output is logged in the `/var/cluster/spm` directory.

Note – Sun Cluster installation output is logged in the `/var/cluster/logs/install/scinstall.log.pid` file, where `pid` is the process ID number of the `scinstall` instance.

23. Use SunPlex Manager to verify quorum assignments and modify them, if necessary.

Note – Shared quorum devices are optional for clusters with three or more nodes. SunPlex Manager might or might not have assigned quorum votes to any quorum devices, depending on whether appropriate shared disks were available.

24. Your cluster is now installed. If you configured SunPlex Manager to install Solstice DiskSuite and optional data services for you, your cluster will have one or more of the following metasets and cluster file systems, depending upon the number of shared disks on your cluster.

Table 1-2 Metasets Installed by SunPlex Manager

Shared Disks	Metaset Name	Cluster File System Mount Point	Purpose
First pair	mirror-1	`/global/mirror-1`	Sun Cluster HA for NFS or Sun Cluster HA for Apache scalable data service, or both

Table 1-2 Metasets Installed by SunPlex Manager

Shared Disks	Metaset Name	Cluster File System Mount Point	Purpose
Second pair	stripe-1	`/global/stripe-1`	Unused
Third pair	concat-1	`/global/concat-1`	Unused

Note – Although it is convenient to use SunPlex Manager to install DiskSuite and optional data services, the mount points and global file system sizes may not be optimal for your environment. In some cases, you may prefer to perform these installations manually after the cluster framework software installation.

25. Proceed to post-installation setup.

Note – The remaining steps are identical to the ones you performed during the command-line `scinstall` installation. Therefore, they are not covered in detail here. If you need to review this material, refer to the ES-333 Sun Cluster 3.0 Administration course material or the *Sun Cluster 3.0 U1 Installation Guide.*

Performing Post-Installation Setup

1. Verify that the name service look-up order is correct in `/etc/nsswitch.conf`.

   a. Verify that `cluster` is the first source look-up for the `hosts` and `netmasks` database entries.

   b. For the `hosts` and `netmasks` database entries, place files after cluster.

   c. For all other database entries, place `files` first in look-up order.

```
...
passwd:     files nis
group:      files nis
...
hosts:      cluster files nis
...
netmasks:   cluster files nis
...
```

2. Set up root's environment on each cluster node to include `/usr/sbin` and `/usr/cluster/bin` in the PATH environment variable and `/usr/cluster/man` and `/usr/share/man` in the MANPATH.

3. Verify that all nodes have joined the cluster:

```
# scstat -n
```

4. Verify device connectivity to the cluster:

```
# scdidadm -L
```

5. From any node, verify that cluster install mode is disabled:

```
# scconf -p | grep 'Cluster install mode:'
Cluster install mode: disabled
```

Installing and Configuring Volume Manager Software

If you used SunPlex Manager to install Solstice DiskSuite software, most of the Solstice DiskSuite installation is completed. The remaining steps are include:

1. (Optional) Mirror filesystems on the root disk.
2. (Optional) Create additional disksets.
3. (Optional) Add disk drives to the disksets.
4. (Optional) Repartition drives in a diskset.
5. Create and initialize the `/etc/lvm/md.tab` files.

Note – SunPlex Manager stored the information normally contained in the `md.tab` file in `/var/cluster/spm/metadevice_db`. Add the contents of this file to your `md.tab` file after creating and initializing any additional disksets.

6. For dual-string configurations, configure mediator hosts, check the status of mediator data, and, if necessary, fix bad mediator data.

Configuring Cluster Framework Software

1. (Optional) Create and mount any additional cluster file systems.
2. (Optional) Configure additional public network adapters.
3. (Optional) Create additional NAFO groups.
4. (Optional) Add a backup adapter to the `nafo0` group on each node:

```
# pnmset -c nafo0 -o add adapter
```

5. (Optional) Change a node's private hostname.
6. Edit the `/etc/inet/ntp.conf` file to update node name entries.

Planning, Installing and Configuring Resource Groups and Data Services

Refer to the *Sun Cluster 3.0 U1 Data Services Installation and Configuration Guide* to install and configure additional resource groups and data services.

Troubleshooting SunPlex Manager Cluster Installations

This section provides a list of known issues and offers some troubleshooting tips.

Known Issues with SunPlex Manager Cluster Installations

To ensure a smooth SunPlex Manager installation, prepare the cluster and create the configuration planning worksheets as recommended in the installation instructions. Some of the more commonly overlooked tasks include:

- If you are not using a naming service, verify that each node's name is entered into the `/etc/hosts` file on each node and that the `/etc/nsswitch.conf` file is correctly configured to look for files first.
- If you plan to install Solstice DiskSuite, you must create the 10 MB file system on slice 7 of the root disk using `newfs` and mount the file system as `/sds`. It is not sufficient to create the empty slice and leave it unmounted.
- Verify that all images are available to SunPlex Manager at the same time. SunPlex Manager does not have the ability to prompt you to swap CDs during the installation.

Troubleshooting Tips

Most installation failures can be traced to incorrect hardware configurations or incorrect data entered during the installation.

If you are certain that the configuration is correct:

1. Check the installation log in `/var/cluster/logs/install/scinstall.log.pid`.
2. Check the SunPlex Manager logs in `/var/cluster/spm`.
   a. The commands used to install the cluster and data services are listed in `messages`.
   b. Solstice DiskSuite information is logged in `sds_installation.log`, `meta*_db`, and `SUNWscvw_log`.

3. Review the scripts listed in `/etc/rc3.d/S80spminstall`. These scripts perform all the post-installation tasks and may provide clues to the cause of the failure.
4. Verify that you have installed a supported version of the Solaris Operating Environment and that you installed the correct Solaris software group.
5. Verify that you have installed all necessary patches for Solaris Operating Environment, cluster hardware, volume manager software, and Sun Cluster 3.0. Consult the *Early Notifier Document #24617* at `http://sunsolve.sun.com` to verify patch requirements.
6. If you still cannot determine the cause of the problem, consult SunSolve or local resources for clues.
7. Finally, if the cause can still not be determined, follow the escalation procedures defined for your group.

Installing a Cluster With scinstall's Custom JumpStart

Overview of Custom JumpStart

A Custom JumpStart installation is a type of installation in which the Solaris software is automatically installed on a system based on a user-defined profile.

With Sun Cluster 3.0's scinstall Custom JumpStart option, you can also automate the installation of the cluster or add nodes to an existing cluster.

This section will detail the steps necessary to use the scinstall Custom JumpStart option to create a new cluster.

Note – This module covers Custom JumpStart as it relates to a Sun Cluster 3.0 installation. It does not provide directions for installing a Custom JumpStart server. If you need to review Custom JumpStart directions, please refer to the *Solaris 8 Advanced Installation Guide*. This document is available online at http://docs.sun.com.

Preparing for Cluster Installation with scinstall's Custom JumpStart

1. Ensure that the cluster hardware setup is complete and connections are verified before you install any software.
2. Plan your installation following the guidelines in Chapter 1 of the *Sun Cluster 3.0 U1 Installation Guide*. Online, this guide is available at: http://docs.sun.com.
3. Obtain and complete configuration planning worksheets from the *Sun Cluster 3.0 Release Notes* at: http://docs.sun.com or set up a copy of your own planning worksheets. The worksheets3 are labelled:

a. Local File System Layout Worksheet

Example: Local File System Layout--With Mirrored Root

Node name phys-schost-1

Mirrored root

Volume Name	Component	Component	File System	Size
d1	c0t0d0s0	c1t0d0s0	/	1168MB
d2	c0t0d0s1	c1t0d0s1	swap	750MB
d3	c0t0d0s3	c1t0d0s3	/globaldevices	100MB
d7	c0t0d0s7	c1t0d0s7	SDS replica	10MB

Figure 1-12 Sample Local File System Layout Worksheet

b. Cluster and Node Names Worksheet

Example: Cluster and Node Names

Cluster name: sccluster

Private network address: 172 . 16 . 0.0 (default: 172.16.0.0)

Private network mask: 255.255. 0 . 0 (default: 255.255.0.0)

Sponsor node name: phys-schost-1

Private hostname

Default name: clusternode 1 -priv (node ID)

Change to: phys-schost-1-priv (optional)

Additional nodes

Node name: phys-schost-2	Node name:
Private hostname	Private hostname
Default name: clusternode 2 -priv (node ID)	Default name: clusternode _____ -priv (node ID)
Change to: phys-schost-2-priv (optional)	Change to: _____ (optional)

Figure 1-13 Sample Cluster and Node Names Worksheet

c. Cluster Interconnect Worksheet

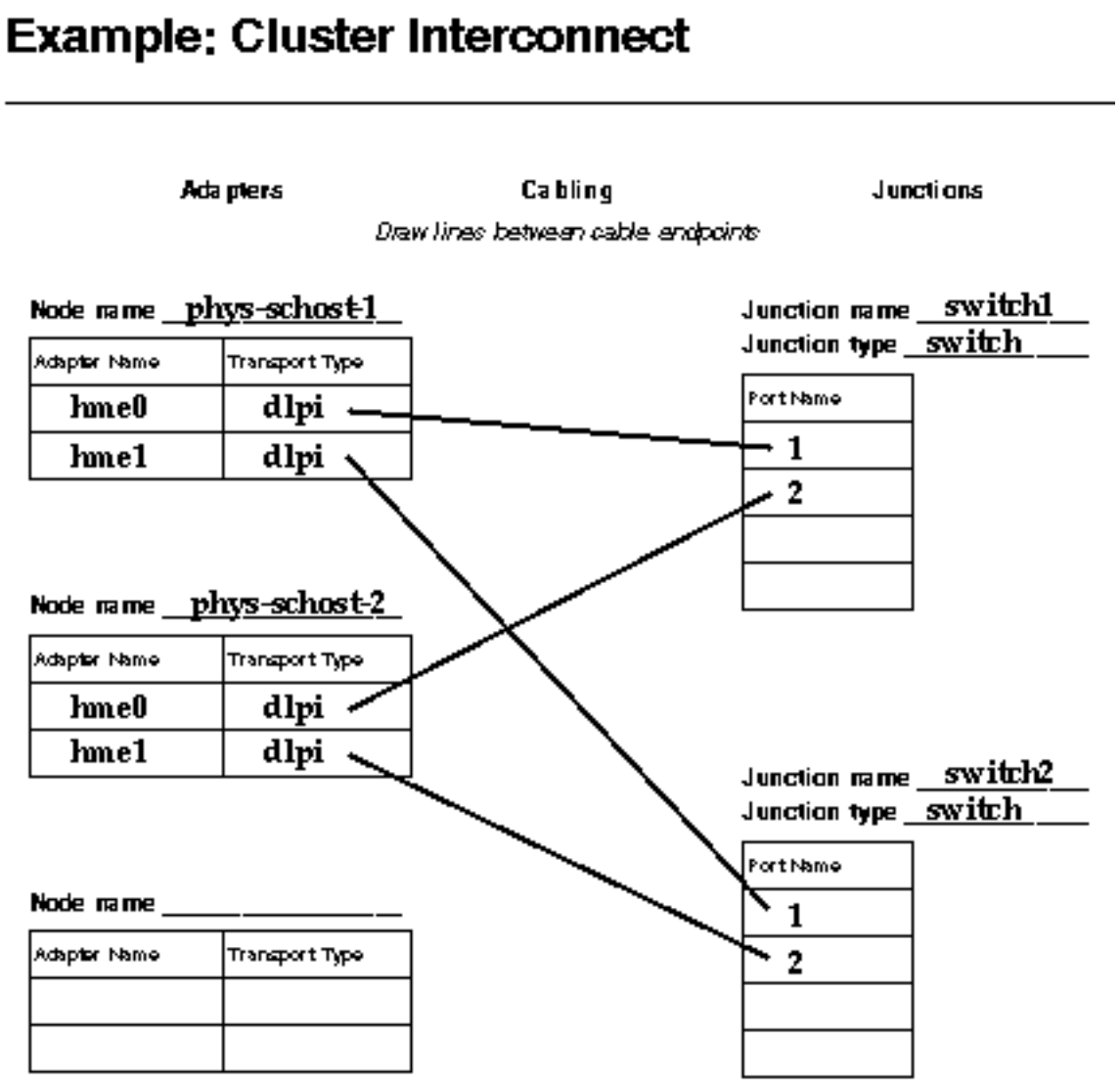

Figure 1-14 Sample Cluster Interconnect Worksheet

4. If you are using a naming service (such as NIS, NIS+, or DNS), add address-to-name mappings for all public hostnames and logical addresses, as well as the IP address and hostname of the JumpStart server.

Performing Cluster Installation with scinstall's Custom JumpStart Option

Installing a cluster with Custom Jumpstart consists of the following major tasks:

1. Set up the JumpStart install server for Solaris Operating Environment installation.
2. Set up the JumpStart install server for cluster installation.
3. Install Solaris Operating Environment and Sun Cluster 3.0 software.
4. Perform post-installation setup.
5. Install and configure volume manager software.
6. Configure cluster framework software.
7. Plan, install and configure resource groups and data services.

Setting Up the JumpStart Install Server for Solaris Operating Environment Installation

As superuser, set up the JumpStart install server for Solaris Operating Environment installation. See the `setup_install_server`(1M) and `add_install_client`(1M) man pages and the *Solaris 8 Advanced Installation Guide* for instructions on how to set up a JumpStart install server.

When you set up the install server, ensure that the following requirements are met.

- The install server is on the same subnet as the cluster nodes, but is not itself a cluster node.
- The install server installs a release of the Solaris Operating Environment required by the Sun Cluster software.
- A Custom JumpStart directory exists for JumpStart installation of Sun Cluster. This *`jumpstart-dir`* directory must contain a copy of the check utility and be NFS exported for reading by the JumpStart install server.
- Each new cluster node is configured as a Custom JumpStart install client.

Setting up the JumpStart Install Server for Cluster Installation.

1. Create a directory on the JumpStart install server to hold your copy of the Sun Cluster 3.0 7/01 CD-ROM, if one does not already exist.

   ```
   # mkdir -m 755 /export/suncluster
   ```

2. Change to the location of the Sun Cluster software Tools directory on the CD-ROM.

   ```
   # cd <path to Sun Cluster 3.0 software>
   /SunCluster_3.0/Tools
   ```

3. Copy the contents of the CD-ROM to a new directory on the JumpStart install server.

Note – The scinstall command creates the new installation directory as it copies the CD-ROM files.

```
# ./scinstall -a /export/suncluster/sc30
```

4. Ensure that the Sun Cluster 3.0 7/01 CD-ROM image on the JumpStart install server is NFS exported for reading by the JumpStart install server.

5. Use scinstall to configure Custom JumpStart finish scripts.

   JumpStart uses these finish scripts to install the Sun Cluster software.

   The following example uses the information completed on the sample configuration worksheets. Substitute the information from your configuration worksheets on an actual installation.

   a. On the JumpStart install server, start the scinstall utility.

   ```
   # cd /export/suncluster/sc30/SunCluster_3.0/Tools
   # ./scinstall
   ```

Warning – Interactive scinstall enables you to type ahead. Therefore, do not press Return more than once if the next menu screen does not appear immediately.

Note – Unless otherwise noted, you can press Control-D to return to either the start of a series of related questions or to the Main Menu.

b. From the Main Menu, type 3 (Configure a cluster to be JumpStarted from this install server).

Note – If option 3 does not have an asterisk in front, the option is disabled because JumpStart setup is not complete or has an error. Exit the scinstall utility, repeat Step 3 through Step 5to correct JumpStart setup, then restart the scinstall utility.

```
*** Main Menu ***

    Please select from one of the following (*) options:

      * 1) Establish a new cluster using this machine as the first node
      * 2) Add this machine as a node in an established cluster
      * 3) Configure a cluster to be JumpStarted from this install server
        4) Add support for new data services to this cluster node
        5) Print release information for this cluster node

      * ?) Help with menu options
      * q) Quit

    Option: 3

  *** Custom JumpStart ***


    This option is used to configure each node in a cluster to be
    JumpStarted from this Solaris install server. Before this option can
    be used, this server must already be set up as a Solaris install
    server and configured to JumpStart each node as a Solaris install
    client. Refer to the Solaris documentation for more information on
    how to set up a Solaris install server, Solaris install clients, and
    a custom JumpStart directory.

    You will be asked to provide all of the information usually needed to
    directly add each node to a cluster. This information will be stored
    for later use under whatever custom JumpStart directory you specify.
    The rules file will be updated to point to both default Solaris
    install profile and a special custom JumpStart finish script.

    Press Ctrl-d at any time to return to the Main Menu.


    Do you want to continue (yes/no) [yes]? yes
```

c. Specify the JumpStart directory name. Enter the path to the directory which contains the check file.

```
>>> Custom JumpStart Directory <<<

...

What is your JumpStart directory name?  /jumpstart
```

d. Specify the name of the cluster.

```
>>> Cluster Name <<<

...

What is the name of the cluster you want to establish? sccluster
```

e. Specify the names of all cluster nodes.

```
  >>> Cluster Nodes <<<

    This release of Sun Cluster supports a total of up to 8 nodes.

    Please list the names of all cluster nodes planned for the initial
    cluster configuration. You must enter at least two nodes. List one
    node name per line. When finished, type Control-D:

    Node name:  phys-schost-1
    Node name:  phys-schost-2
    Node name (Ctrl-D to finish):  ^D


    This is the complete list of nodes:

        phys-schost-1
        phys-schost-2

    Is it correct (yes/no) [yes]?  yes
```

f. Specify whether to use data encryption standard (DES) authentication.

```
  >>> Authenticating Requests to Add Nodes <<<

    Once the first node establishes itself as a single node cluster,
```

```
other nodes attempting to add themselves to the cluster configuration
must be found on the list of nodes you just provided. The list can be
modified once the cluster has been established using scconf(1M), or
other tools.

By default, nodes are not securely authenticated as they attempt to
add themselves to the cluster configuration. This is generally
considered adequate, since nodes which are not physically connected
to the private cluster interconnect will never be able to actually
join the cluster. However, DES authentication is available. If DES
authentication is selected, you must configure all necessary
encryption keys before any node can join (see keyserv(1M),
publickey(4)).

Do you need to use DES authentication (yes/no) [no]?  no
```

g. Specify the private network address and netmask.

Note – You cannot change the private network address after the cluster is successfully formed.

```
>>> Network Address for the Cluster Transport <<<

The private cluster transport uses a default network address of
172.16.0.0. But, if this network address is already in use elsewhere
within your enterprise, you may need to select another address from
the range of recommended private addresses (see RFC 1597 for
details).

If you do select another network address, please bear in mind that
the Sun Clustering software requires that the rightmost two octets
always be zero.

The default netmask is 255.255.0.0; you may select another netmask,
as long as it minimally masks all bits given in the network address
and does not contain any "holes".

Is it okay to accept the default network address (yes/no) [yes]? yes

Is it okay to accept the default netmask (yes/no) [yes]? yes
```

h. If this is a two-node cluster, specify whether the cluster uses transport junctions.

Note – You can specify that the cluster uses transport junctions, regardless of whether the nodes are directly connected to each other. If you specify that the cluster uses transport junctions, you can more easily add new nodes to the cluster in the future.

```
>>> Point-to-Point Cables <<<

    The two nodes of a two-node cluster may use a directly-connected
    interconnect. That is, no cluster transport junctions are configured.
    However, when there are greater than two nodes, this interactive form
    of scinstall assumes that there will be exactly two cluster transport
    junctions.

    Does this two-node cluster use transport junctions (yes/no)[yes]? yes
```

i. If this cluster uses transport junctions, specify the transport junction names.

```
What is the name of the first junction in the cluster [switch1]? switch1

What is the name of the second junction in the cluster [switch2]? switch2
```

Note – You must use transport junctions if a cluster contains three or more nodes. You can use the default names `switchN` or create your own names.

j. Specify the cluster interconnect transport adapters and, if used, the names of the transport junctions they connect to.

Note – You can configure up to two adapters by using the `scinstall` command. You can configure additional adapters after Sun Cluster software is installed by using the `scsetup` utility.

```
>>> Cluster Transport Adapters and Cables <<<

    You must configure at least two connection points to the private
    cluster transport for each node in the cluster. More than two
    connection points are allowed, but this interactive form of scinstall
    assumes exactly two.

    Note that interactive scinstall does not allow you to specify any
```

```
    special transport adapter properties settings. If your adapters have
    special properties which must be set, you may need to use
    non-interactive scinstall by specifying a complete set of command
    line options. For more information, please refer to the man pages for
    your adapters in the scconf_transp_adap family of man pages (e.g.,
    scconf_transp_adap_hme(1M)).

 For node "phys-schost-1",
    What is the name of the first cluster transport adapter? hme0

...

 For node "phys-schost-1",
    Name of the junction to which "hme0" is connected [switch1]? switch1

...

For more information regarding port naming requirements, refer to the
    scconf_transp_jct family of man pages (e.g.,
    scconf_transp_jct_dolphinswitch(1M)).

For node "phys-schost-1",
    Okay to use the default for the "hme0" connection (yes/no) [yes]? yes

 For node "phys-schost-1",
    What is the name of the second cluster transport adapter?  hme1

 For node "phys-schost-1",
    Name of the junction to which "hme1" is connected [switch2]? switch2

 For node "phys-schost-1",
    Use the default port for the "hme1" connection (yes/no) [yes]? yes

 For node "phys-schost-2",
    What is the name of the first cluster transport adapter? hme0

 For node "phys-schost-2",
    Name of the junction to which "hme0" is connected [switch1]? switch1

 For node "phys-schost-2",
    Okay to use the default for the "hme0" connection (yes/no) [yes]? yes

 For node "phys-schost-2",
    What is the name of the second cluster transport adapter? hme1

 For node "phys-schost-2",
```

```
    Name of the junction to which "hme1" is connected [switch2]? switch2

 For node "phys-schost-2",
    Use the default port for the "hme1" connection (yes/no) [yes]? yes
```

k. Specify the global devices file system name.

```
  >>> Global Devices File System <<<

    Each node in the cluster must have a local file system mounted on
    /global/.devices/node@<nodeID> before it can successfully participate
    as a cluster member. Since the "nodeID" is not assigned until
    scinstall is run, scinstall will set this up for you. However, in
    order to do this, you must supply the name of either an
    already-mounted file system or raw disk partition at this time. This
    file system or partition should be at least 100 MB in size.

    If an already-mounted file system is used, the file system must be
    empty. If a raw disk partition is used, a new file system will be
    created for you.

    The default is to use /globaldevices.

 For node "phys-schost-1",
    Is it okay to use this default (yes/no) [yes]? yes

 For node "phys-schost-2",
    Is it okay to use this default (yes/no) [yes]? yes
```

l. Accept or decline the generated scinstall commands.

Note – If you do not accept the generated commands, the scinstall utility returns you to the Main Menu. From there you can rerun menu option 3 and provide different answers. Your previous answers display as the defaults.

```
  >>> Confirmation <<<

    Your responses indicate the following options to scinstall:

------------------------------------------

 For node "phys-schost-1",
      scinstall -c /jumpstart -h phys-schost-1
           -C zoo
```

```
          -F
          -T node=phys-schost-1,node=phys-schost-2,authtype=sys
          -A trtype=dlpi,name=hme0 -A trtype=dlpi,name=hme1
          -B type=switch,name=switch1 -B type=switch,name=switch2
          -m endpoint=:hme0,endpoint=switch1
          -m endpoint=:hme1,endpoint=switch2

    Are these the options you want to use (yes/no) [yes]? yes

----------------------------------------

 For node "phys-schost-2",
      scinstall -c /jumpstart -h phys-schost-2
          -C zoo
          -N proto192
          -A trtype=dlpi,name=hme0 -A trtype=dlpi,name=hme1
          -m endpoint=:hme0,endpoint=switch1
          -m endpoint=:hme1,endpoint=switch2

    Are these the options you want to use (yes/no) [yes]? yes

----------------------------------------

    Do you want to continue with JumpStart set up (yes/no) [yes]? yes
```

m. If necessary, make adjustments to the default class file, or profile, created by scinstall.

The scinstall command creates the following autoscinstall.class default class file in the *jumpstart-dir*/autoscinstall.d/3.0 directory.

```
install_type    initial_install
system_type     standalone
partitioning    explicit
filesys         rootdisk.s0 free /
filesys         rootdisk.s1 750 swap
filesys         rootdisk.s3 100  /globaldevices
filesys         rootdisk.s7 10
cluster         SUNWCuser       add
package         SUNWman         add
```

The default class file installs the End User System Support software group (SUNWCuser) of Solaris software. For Sun Enterprise E10000 server servers, you must install the Entire Distribution + OEM software group (SUNWCXall). Also, some third-party software, such

as Oracle, might require additional Solaris software packages. See your third-party documentation for any Solaris software requirements.

You can change the profile in one of the following ways.

- Edit the autoscinstall.class file directly. These changes are applied to all nodes in all clusters that use this custom JumpStart directory.
- Update the rules file to point to other profiles, then run the check utility to validate the rules file.

As long as the Solaris Operating Environment install profile meets minimum Sun Cluster file system allocation requirements, there are no restrictions on other changes to the install profile. See System Disk Partitions for partitioning guidelines and requirements to support Sun Cluster 3.0 software.

6. Create a patch directory to contain Solaris and hardware patches. This directory must be exported so that it is visible to the JumpStart install server. Copy all uncompressed patches to this directory.
7. Create links to the patch directory in each *jumpstart-dir*/autoscinstall.d/nodes/*node* directory on the JumpStart install server.

```
# ln -s <path to patch dir> jumpstart-dir/autoscinstall.d/nodes/node/patches
```

8. Set up files to contain the necessary hostname information locally on each node.
   a. On the JumpStart install server, create a directory tree under each node to hold the hosts file:

```
# mkdir -p jumpstart-dir/autoscinstall.d/nodes/node/archive/etc/inet
```

   b. Create a hosts file that contains at least the following entries:

- IP address and hostname of the NFS server that holds a copy of the Sun Cluster CD-ROM image. This could be the JumpStart install server or another machine.
- IP address and hostname of each node in the cluster.

Place the hosts file in a directory that is visible to the JumpStart install server.

   c. Create links to the hosts file in each node directory on the JumpStart install server.

```
# ln -s <path to hosts file> \
jumpstart-dir/autoscinstall.d/nodes/node/archive/etc/inet/hosts
```

9. Add your own post-installation finish script.

   You can add your own finish script, which is run after the standard finish script installed by the `scinstall` command.

   a. Name your finish script `finish`.

   b. Copy your finish script to the *`jumpstart-dir`*/autoscinstall.d/nodes/*`node`* directory, one directory for each node in the cluster.

      Alternately, place the `finish` file in a directory that is visible to the JumpStart install server and place links in the node directories as in the previous step.

Installing Solaris Operating Environment and Sun Cluster 3.0 Software.

1. If you use an administrative console, display a console screen for each node in the cluster.

   Otherwise, you must connect to the consoles of each node individually.

2. From the ok PROM prompt on the console of each node, type the boot net - install command to begin the network JumpStart installation of each node:

   `ok boot net - install`

Note – Sun Cluster installation output is logged in the `/var/cluster/logs/install/scinstall.log.pid` file, where `pid` is the process ID number of the `scinstall` instance.

3. When the installation is successfully completed, each node is fully installed as a new cluster node.

4. Install any Sun Cluster software patches.

   See the *Sun Cluster 3.0 U1 Release Notes* for the location of patches and installation instructions.

5. If you installed any Sun Cluster software patches that require you to reboot the entire cluster, perform the following steps to reboot:

   a. From one node, shut down the cluster.

   `# scshutdown`

Warning – Do not reboot the first-installed node of the cluster until after the cluster is shut down.

b. Reboot each node in the cluster.

```
ok boot
```

Note – Until cluster install mode is disabled, only the first-installed node, which established the cluster, has a quorum vote. In an established cluster that is still in install mode, if the cluster is not shut down before the first-installed node is rebooted, the remaining cluster nodes cannot obtain quorum and the entire cluster shuts down. Cluster nodes remain in install mode until the first time you run the `scsetup` command, during the post-installation setup.

6. Complete the remaining steps to install and configure the cluster:

   a. Perform post-installation setup.

   b. Install and configure volume manager software.

   c. Configure cluster framework software.

   d. Plan, install and configure resource groups and data services.

Note – The remaining steps are identical to the ones you performed during the command-line `scinstall` installation. Therefore, they are not covered in detail here. If you need to review this material, refer to the ES-333 Sun Cluster 3.0 Administration course material or the *Sun Cluster 3.0 U1 Installation Guide.*

Troubleshooting Cluster JumpStart Installations

This section provides a list of known issues and offers some troubleshooting tips.

Known Issues with Cluster JumpStart Installations

To ensure a smooth installation with Custom JumpStart, prepare the cluster and create the configuration planning worksheets as recommended in the installation instructions. Two of the more commonly overlooked tasks include:

- If you are not using a naming service, verify that each node is entered into the `/etc/hosts` file on the JumpStart install server and that the `/etc/nsswitch.conf` file is correctly configured to look for files first.
- Verify that all images and the jumpstart directory are NFS-exported.

Bug ID 4359321

Problem Summary: The `scinstall` utility enables you to specify `/global` as the directory name for the global devices file system. However, because the mount point for the global devices file system is `/global/.devices/node@nodeid`, this specification should not be enabled.

Workaround: Re-install the node using `/globaldevices` as the directory name for the global devices file system. Although not preferred, fixing the entries in the `/etc/vfstab` files, rebooting the cluster, and then running the `scgdevs` command is a possible workaround. Check that each `/global/.devices/node@nodeid` entry in each `/etc/vfstab` file has the global mount option set.

Troubleshooting Tips

If your JumpStart installation is not working properly, first determine whether the problem is due to Custom JumpStart or the `scinstall` JumpStart option.

1. If the node cannot reach the Custom JumpStart server to begin the installation, the problem is most likely due to the Custom JumpStart setup. Verify that you have set up the JumpStart install server correctly, according to the instructions in the *Solaris 8 Advanced Installation Guide*.
2. If the node installs Solaris Operating Environment and the Sun Cluster 3.0 software correctly, but the cluster is not working, the problem is most likely due to the `scinstall` setup.
   a. Examine the installation log file in `/var/cluster/logs/install/scinstall.log.pid` file on each node for any clues to the failure.
   b. Verify the content of the files in *`jumpstart-dir`*`/autoscinstall.d/3.0`. If you find an error, it may make the most sense to re-run scinstall to recreate the JumpStart configuration.
   c. Verify that you have set up the `hosts` and `finish` files correctly on the JumpStart install server. Also, make sure that each node has a copy or link to these files in *`jumpstart-dir`*`/autoscinstall.d/nodes/`*`node`*. Correct any errors and re-install.
   d. Verify that you have installed a supported version of the Solaris Operating Environment and that you installed the correct Solaris software group.
   e. Verify that you have installed all necessary patches for Solaris Operating Environment, cluster hardware, volume manager software, and Sun Cluster 3.0. Consult the *Early Notifier Document #24617* at `http://sunsolve.sun.com` to verify patch requirements.
3. If you still cannot determine the cause of the problem, consult SunSolve or local resources for clues.
4. Finally, if the cause can still not be determined, follow the escalation procedures defined for your group.

Troubleshooting scinstall Cluster Installations

This section provides a list of known issues and offers some troubleshooting tips.

Known Issues With scinstall Cluster Installations

To ensure a smooth installation with scinstall, prepare the cluster and create the configuration planning worksheets as recommended in the installation instructions. One commonly overlooked task is:

- If you are not using a naming service, verify that each node's name is entered into the /etc/hosts file on each node and that the /etc/nsswitch.conf file is correctly configured to look for files first.

Bug ID 4359321

Problem Summary: The scinstall utility enables you to specify /global as the directory name for the global devices file system. However, because the mount point for the global devices file system is /global/.devices/node@nodeid, this specification should not be enabled.

Workaround: Re-install the node using /globaldevices as the directory name for the global devices file system. Although not preferred, fixing the entries in the /etc/vfstab files, rebooting the cluster, and then running the scgdevs command is a possible workaround. Check that each /global/.devices/node@nodeid entry in each /etc/vfstab file has the global mount option set.

Troubleshooting Tips

Most installation failures can be traced to incorrect hardware configurations or incorrect data entered during the installation.

If you are certain that the configuration is correct:

1. Check the installation log in `/var/cluster/logs/install/scinstall.log.pid.`
2. Verify that you have installed a supported version of the Solaris Operating Environment and that you installed the correct Solaris software group.
3. Verify that you have installed all necessary patches for Solaris Operating Environment, cluster hardware, volume manager software, and Sun Cluster 3.0. Consult the *Early Notifier Document #24617* at `http://sunsolve.sun.com` to verify patch requirements.
4. If you still cannot determine the cause of the problem, consult SunSolve or local resources for clues.
5. Finally, if the cause can still not be determined, follow the escalation procedures defined for your group.

Selecting an Installation Method

No installation method is preferred over another. Table 1-3 summarizes some of the advantages and disadvantages of each method.

Table 1-3 Installation Method Comparison

Installation Method	Advantages	Disadvantages
`scinstall`	Text-based, requires no special preparation	Manual, can be prone to errors. Solaris and cluster software must be installed manually on each node.
SunPlex Manager	Graphical interface, performs most cluster installation and configuration functions automatically	Solaris and SunPlex Manager must be installed manually on each node
`scinstall` with custom JumpStart	Provides greatest automation, ensures consistent installations	Requires additional preparation time, still must run `scinstall` manually, requires JumpStart expertise

You may choose to consider custom JumpStart if you are installing a cluster with three or more nodes. If you are not familiar with JumpStart, however, you may wish to select one of the more manual methods. In any case, select the method that makes the most sense for your environment and level of expertise.

Module 1 Lab: Installing a Cluster with SunPlex Manager

Make sure to prepare the classroom according to the README_setup_instructions in the lab files directory. Solaris 8 should be installed with all patches. The students will need access to the Sun Cluster framework and agents software, Solaris 8 software, DiskSuite and cluster patches (if any). In addition, they must be able to access the cluster through an administrative console as well as a Web browser, both of which can be on the same workstation.

After the students have finished the labs, ask them to discuss their experiences. Was this method easier or more difficult than the scinstall? Are they likely or not to use this method in the future?

The purpose of the following lab is to use SunPlex Manager to create a working two-node cluster. You will complete the following tasks:

1. Install the SunPlex Manager software packages.
2. Install the cluster framework, Sun Cluster HA for NFS, and Sun Cluster HA for Scalable Apache with SunPlex Manager.
3. Perform post-installation setup.
4. Install and configure volume manager software.
5. Configure cluster framework software.

Task: Installing the SunPlex Manager Software Packages

1. Obtain the name of the administrative workstation and each cluster node from your instructor.

   Administrative Workstation Name: _____

   Node 1 Name: _____

   Node 2 Name: _____

2. Obtain the location of the Sun Cluster software from your instructor.

   Location to Sun Cluster software:

   _____

3. Log in as root to one of your cluster nodes from the administrative console.
4. Verify that Solaris Apache is installed on your cluster nodes:

   ```
   # pkginfo SUNWapchr
   # pkginfo SUNWapchu
   ```

```
# pkginfo SUNWapchd
```

5. If these packages are not installed, you must install them from the Solaris image. Obtain the location of the Solaris software from your instructor and install the packages as follows:

```
# cd <path_to_Solaris_image>/Solaris_8/Product
# pkgadd -d . SUNWapchr SUNWapchu SUNWapchd
```

6. Install any necessary Apache patches. Ask your instructor for the location of the patch files.

7. Change to the <*path_to_Update_1_Files*>/SunCluster_3.0/Packages directory:

```
# cd <path_to_Update_1_Files>/SunCluster_3.0/Packages
```

8. Install the SunPlex Manager software packages and answer yes to all prompts:

```
# pkgadd -d . SUNWscva SUNWscvr SUNWscvw
```

9. Repeat Steps 3 through 8 on each node of the cluster.

Installing the Cluster Framework, Solstice DiskSuite and Optional Data Services with SunPlex Manager

1. Fill in the worksheet below. Obtain any missing information from your instructor.

Table 1-4 Installation Worksheet

Item	Detail
Cluster Name	
Node 1 IP Address	
Node 2 IP Address	
Node 1 Adapter 1	
Node 2 Adapter 1	
Node 1 Adapter 2	
Node 2 Adapter 2	
NFS Logical Hostname and IP Address	
Apache Shared Address Name and IP Address	
Path to Solaris Operating Environment Image	
Path to Sun Cluster Framework Software	
Path to Sun Cluster Data Services Software	
Path to Sun Cluster and Solstice DiskSuite Patches	

2. Add the cluster node names and IP address to the `/etc/hosts` file.

3. Verify that `hosts` entry in `/etc/nsswitch.conf` is set to look for `files` first.

4. Verify that each cluster node has at least a 10MB /sds partition mounted on slice 7. If needed, create this file system:

```
# newfs /dev/rdsk/c0t0d0s7
# mkdir /sds
# mount /dev/dsk/c0t0d0s7 /sds
```

Note – You may need to substitute a different device name for *c0t0d0*, but the `/sds` file system must be mounted on slice 7.

5. Launch a browser from the administrative console, or any other machine outside the cluster.

6. Disable the browser's Web proxy.

Note – If you are using Netscape, you can disable the proxy by selecting Edit / Preferences / Advanced / Proxies from the Netscape menu. Click Direct Connect to the Internet to disable the proxy.

7. Ensure disk caching and memory caching are enabled. The disk cache and memory cache size must be greater than 0.

Note – If you are using Netscape, you can access this information by selecting Edit / Preferences / Advanced / Cache from the Netscape menu.

8. From the browser, connect to port 3000 on one node of the cluster: `https://node:3000`. The Sun Cluster Installation screen displays in the browser window.

Note – If the browser displays a New Site Certification window, follow the onscreen instructions to accept the certificate.

9. Log in as user 'root' and enter the root password.

10. In the Sun Cluster Installation screen, verify that the cluster meets the listed requirements for using SunPlex Manager.

11. If you meet all listed requirements, click Next to continue to the next screen.

12. Type a name for the cluster and select the number of nodes in your cluster. Click Next to continue.

13. Type the name of each cluster node. Click Next to continue.

14. From the pulldown lists for each node, select the names of the two adapters used for the private interconnects. Click Next to continue.
15. Choose Yes to install Solstice DiskSuite software. Click Next to continue.
16. Choose Yes to install Sun Cluster HA for NFS and Sun Cluster HA for Apache.
    a. For Sun Cluster HA for NFS, specify the logical hostname the data service will use.
    b. For Sun Cluster HA for Apache, specify the shared address the data service will use.
    c. Click Next to continue.
17. Type the path for each CD-ROM image needed to install the packages you specified. Optionally, type the path for the patch directory.
    a. Each specified path for a CD-ROM image must be to the directory containing the `.cdtoc` file for the CD-ROM.
    b. For any software package you do not install, leave the relevant path field blank.
    c. If you have already installed the required patches or plan to install them manually, leave the Patch Directory Path field blank.
18. Click Next to continue.
19. If the information you supplied is correct as displayed in the Confirm Information screen, proceed to Step 20.

    If the information is not correct, perform the following steps to correct the configuration information:
    a. Click Back until you return to the screen with the information to change.
    b. Type the correct information and click Next.
    c. Retype or reselect the information in each screen until you return to the Confirm Information screen.
    d. Verify that the information in the Confirm Information screen is now correct.
20. Click Begin Installation to start the installation process.

a. If the browser displays a New Site Certification window, follow the onscreen instructions to accept the certificate.

b. If the browser prompts for login information, type 'root' and the root password.

During installation, the screen displays brief messages about the status of the cluster installation process. When installation is complete, the browser displays the cluster monitoring and administration GUI.

21. Use SunPlex Manager to verify quorum assignments and modify them, if necessary.

22. Your cluster is now installed.

Task: Performing post-installation setup

1. Verify that the name service look-up order is correct in `/etc/nsswitch.conf`.

   a. Verify that `cluster` is the first source look-up for the `hosts` and `netmasks` database entries.

   b. For the `hosts` and `netmasks` database entries, place files after cluster.

   c. For all other database entries, place `files` first in look-up order.

```
...
passwd:     files nis
group:      files nis
...
hosts:      cluster files nis
...
netmasks:   cluster files nis
...
```

2. Set up root's environment on each cluster node to include `/usr/sbin` and `/usr/cluster/bin` in the PATH environment variable and `/usr/cluster/man` and `/usr/share/man` in the MANPATH.

3. Verify that all nodes have joined the cluster:

```
# scstat -n
```

4. Verify device connectivity to the cluster:

```
# scdidadm -L
```

5. From any node, verify that cluster install mode is disabled:

```
# scconf -p | grep 'Cluster install mode:'
Cluster install mode: disabled
```

Task: Installing and Configuring Volume Manager Software

SunPlex Manager installed and configured Solstice DiskSuite on your cluster.

In a normal installation, you would now mirror the filesystems on the root disk and configure the `/etc/lvm/md.tab` file. In the interest of time, you will skip those steps. However, you still must add mediator hosts for your configuration.

1. Become superuser on the node that currently masters the diskset to which you intend to add mediator hosts.
2. Run the `metaset` command to add each node with connectivity to the diskset as a mediator host for that diskset.

```
# metaset -s mirror-1 -a -m node1
# metaset -s mirror-1 -a -m node2
```

Task: Configuring Cluster Framework Software

1. Verify the current public network configuration:

```
# pnmstat -l
```

2. Obtain the name of the public network backup adapter visually or from your instructor.
3. Add the backup adapter to the `nafo0` group on each node:

```
# pnmset -c nafo0 -o add adapter
```

4. Edit the `/etc/inet/ntp.conf` file to update node name entries.

Module 2

Troubleshooting Cluster Components

Objectives

This module provides an introduction to the Sun Cluster 3.0 Software components and associated debug facilities.

By the end of this module you should be able to:

- List the major components of the Sun Cluster 3.0 Software architecture
- Describe the relationship between the major components of the Sun Cluster 3.0 Software architecture
- Describe the relationship between the major Sun Cluster 3.0 Software components and the Solaris™ Operating Environment
- Use adb command to access the debug facilities for the major components of the Sun Cluster 3.0 Software architecture

Instructor Note: Emphasize to the students that this is an introduction only and is not intended to go into the depth of troubleshooting that the cluster sustaining group would perform. Here, troubleshooting is limited to identifying component debug facilities to narrow down a problem, not to identify a specific bug in the cluster software. Also, emphasize that this is the most "academic" module of the course.

One way to present this material is through a digital projector attached to a laptop or a workstation. Lecture from the overhead slides or the book, and stop after each component to show the debug buffer on the screen, using one of the class clusters. Ask the students to access the same debug buffer on their clusters. Allow time for everyone to access the buffers. Discuss anything interesting in the buffer.

Additional Resources

Additional resources – The following references can provide additional details on the topics discussed in this module:

- Sun Cluster 3.0 Product Team TOI Web Site: `http://galileo.eng/SC30TOI/index.html`
- Sun Cluster 3.0 Information: `http://suncluster.eng`
- Sun Cluster 3.0 Mailing Lists: suncluster@Sun.COM, sc30-users@Eng
- Sun Cluster 3.0 Engineering Site, `http://suncluster.eng/products/SC3.0`

Sun Cluster 3.0 Software Architecture

Figure 1-1 shows a high level view of the cluster software architecture and its relationship to client systems, other nodes and storage.

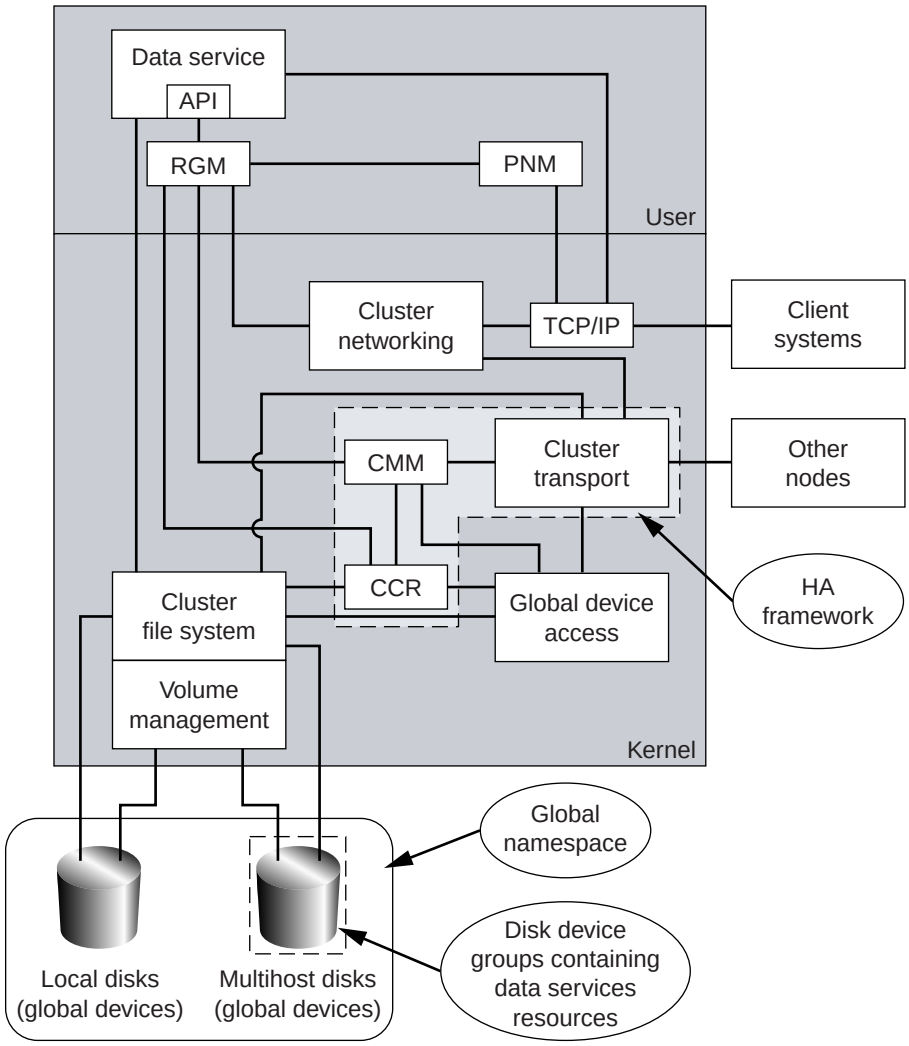

Figure 2-1 Sun Cluster 3.0 Software Architecture

API = Application Programming Interface, RGM = Resource Group Manager, PNM = Public Network Management, CMM = Cluster Membership Monitor, CCR = Cluster Configuration Repository

Sun Cluster 3.0 System Components

Figure 1-2 shows the major cluster components and their relationships.

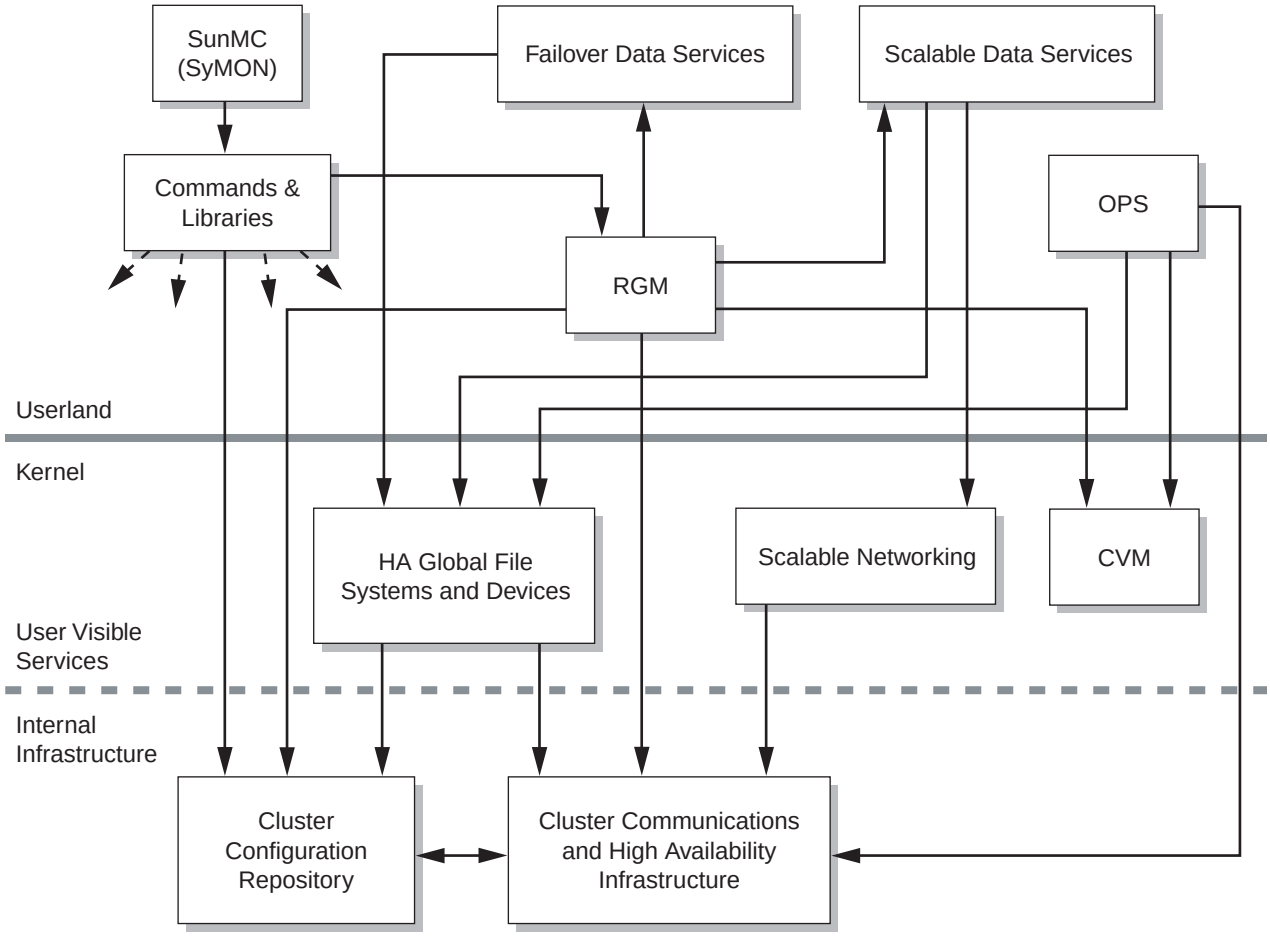

Figure 2-2 Sun Cluster 3.0 System Components

RGM = Resource Group Manager, OPS = Oracle Parallel Server, CVM = Cluster Volume Manager. The students should be familiar with those acronyms.

Most cluster components are now integrated into the Solaris Operating Environment kernel. This is a significant change from SC 2.2 and enables the use of core Solaris services (devices, filesystems and networks).

Instructor Note: Point out that OPS is a port of the 2.2 OPS software and is not as tightly integrated into the cluster as "normal" data services. Refer students to the Configuration Guide at `http://suncluster.eng.sun.com/products/SC3.0/config` for the latest supported OPS configurations.

The remainder of this module will focus on each cluster component.

Cluster Communications and High-Availability Infrastructure

Figure 2-3 illustrates the cluster communications and high-availability infrastructure. Distributed Cluster Services (DCS) include the HA Global File Systems and Devices, Resource Group Manager (RGM), and Scalable Networking. CCR is the internal component of the Cluster Configuration Repository.

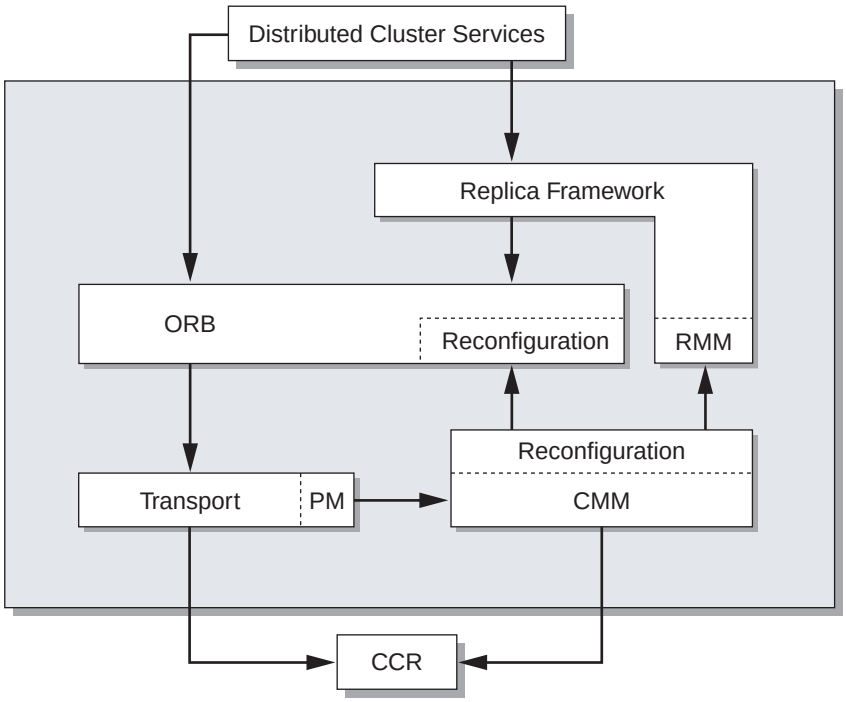

Figure 2-3 Cluster Communications

Note – Do not confuse Distributed Cluster Services (DCS) with the Device Configuration System (DCS) used for global devices. Both acronyms are used throughout engineering documentation.

Object Request Broker (ORB)

Refer students to the Appendix if they are interested in learning more about the ORB.

Most Sun Cluster components are implemented in the object-oriented language C++. Because of this, the principal form of communication between major Solaris Clustering components is accomplished through object invocations.

The ORB is the portion of Solaris clustering that provides object and communications support to other Sun Cluster components. The ORB's primary function is to provide an efficient and reliable means of performing object invocations while simultaneously managing object references. It also provides all communications support to Sun Cluster components, including communications between nodes and between user and kernel space.

Debugging the ORB

The ORB uses debug buffers to log diagnostic information. Use `adb` to enable the ORB's debug buffer on a running system:

```
# adb -wk /dev/ksyms /dev/mem
orb_trace_options/WFFFFFFFF
$q
```

The Sun Cluster Diagnostic Tool Kit (DTK), which is explained in the next module, also contains a tool for displaying ORB state: `orbadmin`.

Instructor Note: Before moving on to the next section, Transport, refer back to Figures 1-2 and 1-3 again and point out where the transport fits into the overall cluster system.

Transport

The transport is responsible for the following functions:

- Managing:
  - Communication over the private network
  - Communication infrastructure for the ORB
  - Communication support for applications
- Detecting and recovering from communication failure
- Communicating node reachability status to the Cluster Membership Monitor (CMM)

Transport Types

Sun Cluster 3.0 can support multiple transport types:

- Remote Shared Memory (RSM)
- Data Link Provider Interface (DLPI), implemented on top of TCP
- UNODE

RSM is not supported with the Sun Cluster 3.0 general release but will be supported in the future. UNODEs are user-level cluster nodes and only used by the development team for debugging cluster code.

RSM is needed for SCI and Wildcat support.

Implementation is inheritance-based. There is a generic transport type, and each specific transport type contains all the features and functions of the generic type as well as those that are specific to its own type.

Transport Components

The major components of the transport are represented in the following table.

Table 2-1 Major Transport Components and Their Functions

Component	Function
Clconf	Transport's mechanism for accessing the Cluster Configuration Repository (CCR). Clconf is used by other cluster components, but the transport uses it only for accessing the CCR.
Topology Manager (TM)	Maps adapter / cable / switch information from the CCR into paths between nodes. A path is the end-to-end connection between a pair of adapters hosted on two nodes. See Figure 2-4 on page 2-9.
Path Manager (PM)	Monitors the status of the paths to determine node reachability. It sends "heartbeats" as 64-bit tokens in real-time threads every one second to each known path to make the determination. If a node is unreachable, it notifies the CMM.
Endpoint Registry (ER)	Repository of all available data paths
Pathend / Endpoint State Machines	Manages changes in data path state. A path is implemented as a pathend and endpoint pair, where the pathend is used to monitor the status of the connection and the endpoint is used to handle the data transport.
Transport Specific Implementations	Manages the details of the interaction between the cluster components
HA Private Networking (also known as ipconf)	Manages communications between non-cluster applications. An HA-IP address is configured between each node and is hosted on the loopback interface. It uses static routes to other nodes.
Miscellaneous	`ifkconfig` – configures adapters at the kernel level `hb_threadpool` – dedicated threadpool for heartbeats

Transport paths are shown in the figure below:

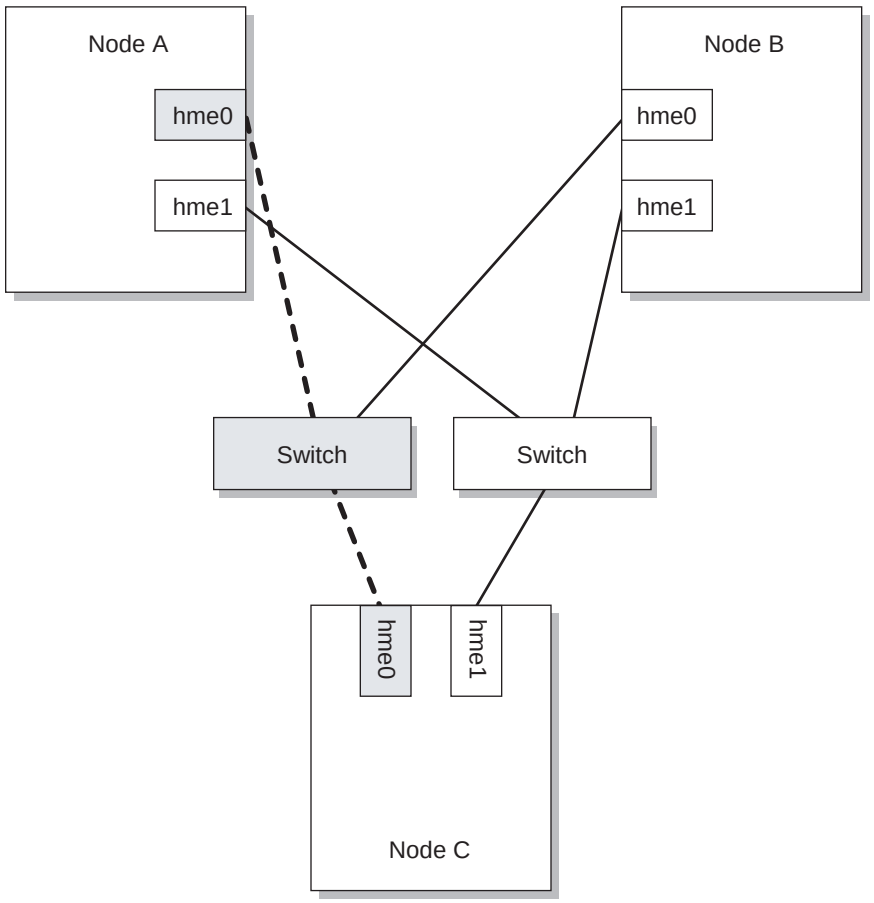

Figure 2-4 Transport Paths

Interactions Within the Transport

Figure 1-5 illustrates the interactions of the components within the transport and with the other cluster components.

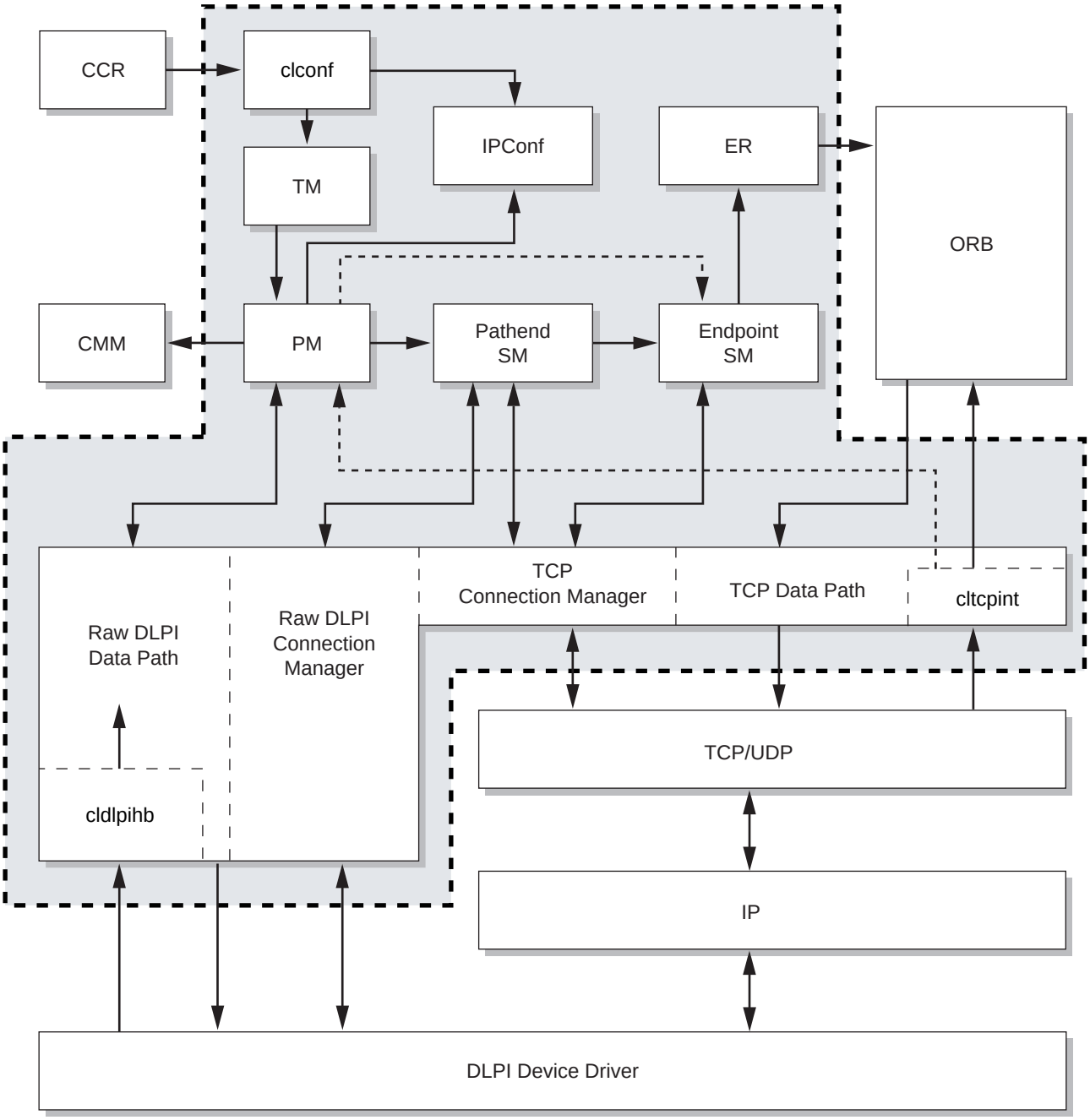

Figure 2-5 Interactions Within the Transport

The ipconf Configuration

The partial output below from `ifconfig -a` illustrates the HA-IP address:

```
# ifconfig -a
lo0: flags=10008c9<UP,LOOPBACK,RUNNING,NOARP,MULTICAST,IPv4> mtu 8232
index 1
        inet 127.0.0.1 netmask ff000000
lo0:1: flags=10088c9<UP,LOOPBACK,RUNNING,NOARP,MULTICAST,PRIVATE,IPv4>
mtu 8232 index 1
        inet 172.16.193.2 netmask ffffffff
```

Debugging the Transport

The transport uses debug buffers to log diagnostic information. The following table lists the debug buffers available through adb.

Table 2-2 Debug Buffers

Debug Buffer Name	Function
pm_dbg	Path manager
pathendp->hb_dbg[p,o]	Heartbeats (two buffers, one for current and one for old)
tcp_debug	TCP connections
raw_dlpi_dbg	Raw DLPI connections
cmm_dbg_buf	CMM debug buffer

Transport Example: Accessing the Path Manager Debug Buffer on a Running System

To access one of the debug buffers on a running system, type the following:

```
# adb -k /dev/ksyms /dev/mem
*pm_dbg/s
```

Your output will be similar to:

```
30002d90008:    th 300019c8fe0 tm  14440361: Path Manager (30000135510)
initialized
th 2a1001d3d40 tm  14461229: Path Manager add client (30001d88008) type 0
```

```
th 300019c8fe0 tm  14462490: Adding A(30002cb3698)
th 300019c8fe0 tm  14462506: Adding P(30002c53530)
th 300019c8fe0 tm  14467708: Adding A(30002cb3be8)
th 300019c8fe0 tm  14467723: Adding P(30002c533f0)
th 300019c8fe0 tm  14468182: registered cmm (30001e48008)
th 30001b93540 tm  14871168: register_pathend proto193:hme0 -
proto192:hme0 P(30
002c533f0) PE(30001e74e20) HBDB(300019f5070)
th 30001b93540 tm  14871171: first pathend P(30002c533f0) PE(30001e74e20)
th 30001b93540 tm  14871179: P(30002c533f0) PE(30001e74e20) added
th 30001b93540 tm  14871181: pathend accepted P(30002c533f0)
PE(30001e74e20)
th 30001b93540 tm  14871184: path_up (30002c533f0)
th 30001b93540 tm  14871185: Increasing max_timeouts[2] from 0 to 12000
th 30001b93540 tm  14871187: Updating max_clockthread_delay from
230584300921369
3951 to 5250000000.
th 30001b93a80 tm  14872019: (2,978741239) node_is_reachable
th 30001b93a80 tm  14872021: (2,978741239) calling CMM node_is_reachable.
th 300019c8fe0 tm  15035295: Path Manager add client (30001d92678) type 1
th 300019c8fe0 tm  15035311: Path Manager add client (3000317db90) type 1
th 300019c8d40 tm  15671226: register_pathend proto193:qfe0 -
proto192:qfe0 P(30
002c53530) PE(30001e74fe0) HBDB(300019f5160)
th 300019c8d40 tm  15671230: P(30002c53530) PE(30001e74fe0) added
th 300019c8d40 tm  15671232: pathend accepted P(30002c53530)
PE(30001e74fe0)
th 300019c8d40 tm  15671234: path_up (30002c53530)
```

The preceding example illustrates the successful addition of two paths to the Path Manager. In the case of an interconnect failure, the buffer might include additional output such as:

```
th 2a100045d40 tm 582553014: Pathend proto192:hme0 - proto193:hme0
P(30002c593f0
) PE(30001e78fe0) HBDB(3000012e408) dropped (reason 2)
th 2a100045d40 tm 582553018: new minimum timeout is 10000 ms
th 2a100045d40 tm 582553019: Updating max_clockthread_delay from
5250000000 to 5
250000000.
th 2a100045d40 tm 582553022: max_timeouts[1] = 12000, tout = 12000, q =
1000
th 2a100045d40 tm 582553024: new maximum tout to node 1 is 10000
th 2a100045d40 tm 582553025: max_timeouts[1] has decreased from 12000 to
12000.
th 2a100045d40 tm 582553027: path_down (30002c593f0)
```

The debug buffer records the failure of the path between `proto192:hme0` and `proto193:hme0` and marks the path as down `(th 2a100045d40 tm 582553027: path_down (30002c593f0).`

To exit adb:

$q

Instructor Note: Before moving on to the next section, CMM, refer back to Figures 1-2 and 1-3 again and point out where the CMM fits into the overall cluster system.

Cluster Membership Monitor (CMM)

The CMM is responsible for the following functions:

- Establishes cluster-wide agreement on the set of nodes that form the cluster
- Coordinates synchronized reconfiguration when cluster membership changes
- Handles cluster partitions (Split Brain, Amnesia) using the majority voting quorum rule and quorum devices
- Ensures full connectivity among cluster members using information from the Transport

Instructor Note: The students should know a fair amount about the CMM from the first class. Make sure they remember what a split brain, amnesia, majority quorum rule ((n/2)+1), and quorum devices are.

CMM Architecture

The following figure illustrates the CMM Architecture.

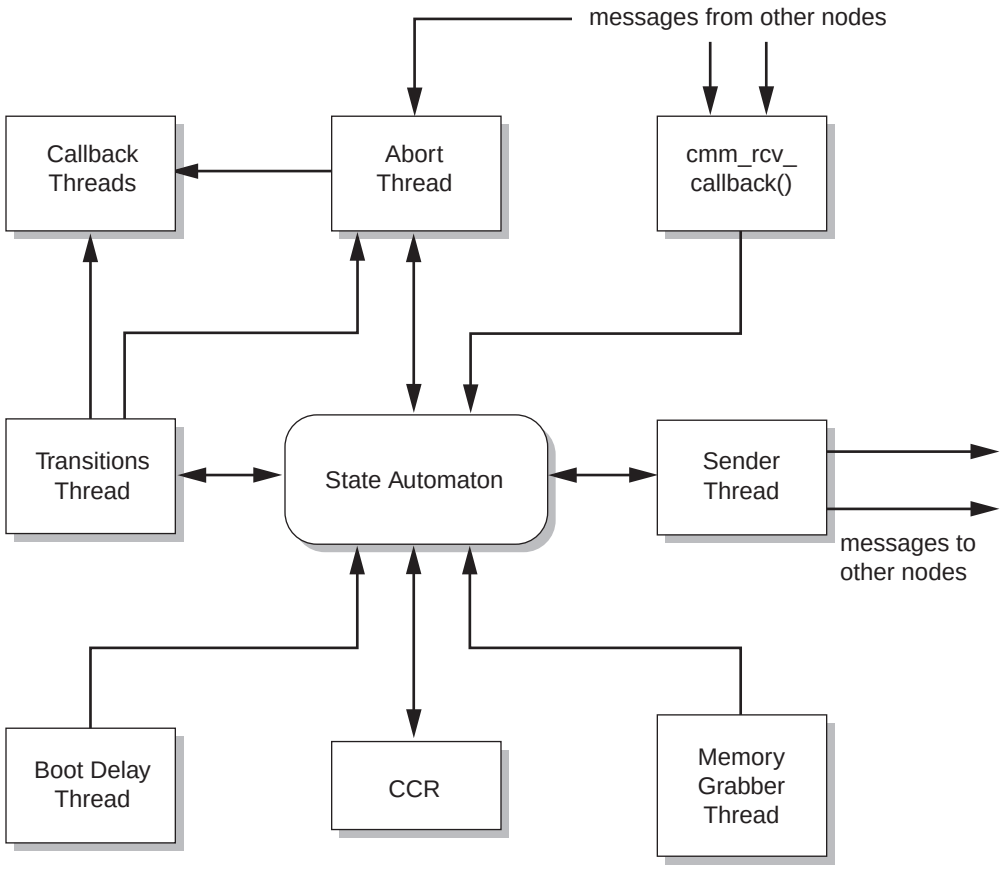

Figure 2-6 CMM Architecture

Most of the components are have descriptive names that match their functions. The following component functions, however, may not be obvious from their names.

State Automaton

The state automaton is the key CMM component. Each node maintains its view of the cluster state. External events, such as the arrival of a message from another node, a timeout, or a completion of a reconfiguration program change the automaton state. When changing its state, the automaton may initiate reconfiguration programs on the local node.

The following figure illustrates the node states and transitions. Note that an abort or stop state can be entered from any other state.

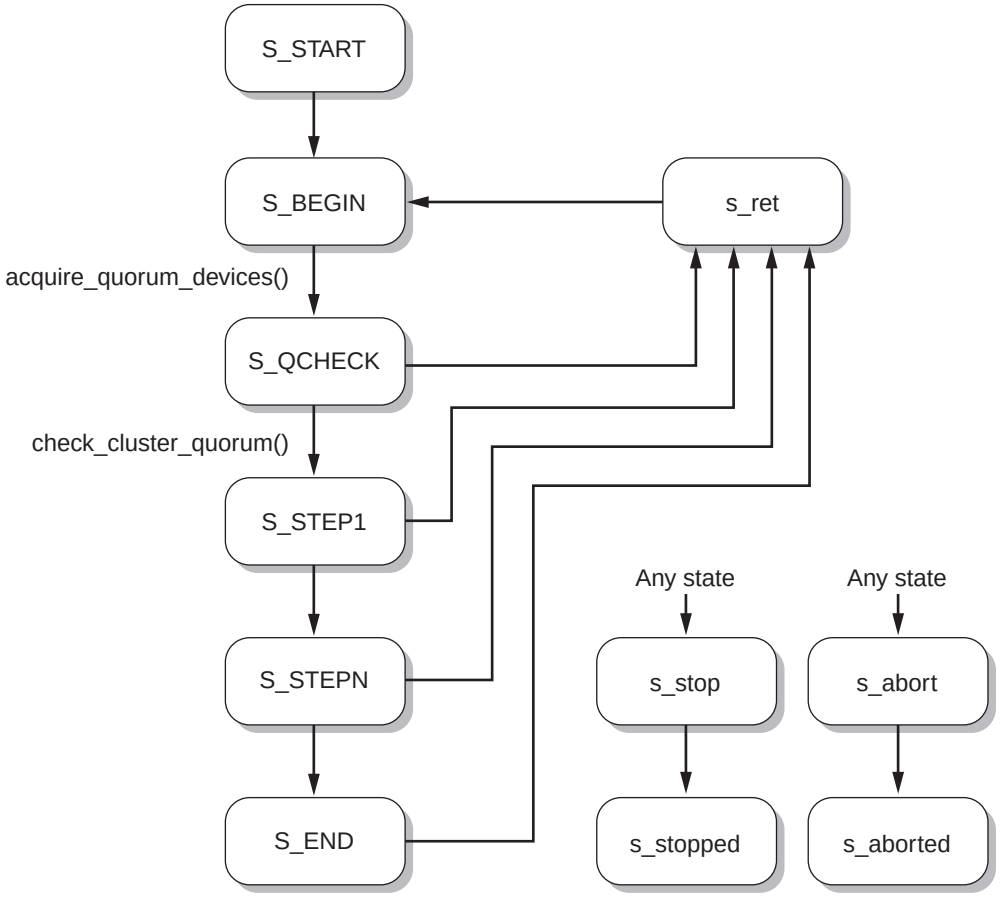

Figure 2-7 CMM Automaton States

Transitions Thread

The transitions thread executes the reconfiguration programs and synchronizes with other nodes through the state automaton.

Boot Delay Thread

The boot delay thread is used during the initial cluster boot to allow time for all the nodes to contact each other. The default boot delay timeout is 60 seconds.

Memory Grabber Thread

This thread is used only in debug mode to simulate out of memory conditions.

Debugging the CMM

The CMM uses the `cmm_dbg_buf` debug buffer to log diagnostic information. The information can include details about quorum devices, cluster membership and reconfiguration steps. This buffer may be accessed using the same procedure described in the section named Transport Example: Accessing the Path Manager Debug Buffer on A Running System:

```
# adb -k /dev/ksyms /dev/mem
*cmm_dbg_buf/s
```

Instructor Note: Before moving on to the next section, Replica Framework, refer back to Figures 1-2 and 1-3 again and point out where the Replica Framework fits into the overall cluster system.

Replica Framework

The Replica Framework provides a mechanism for ensuring that a service (object) is highly available across a cluster by maintaining a replica of a service and its state. If the primary service provider fails, the secondary provider can take over without any noticeable loss of service to the client.

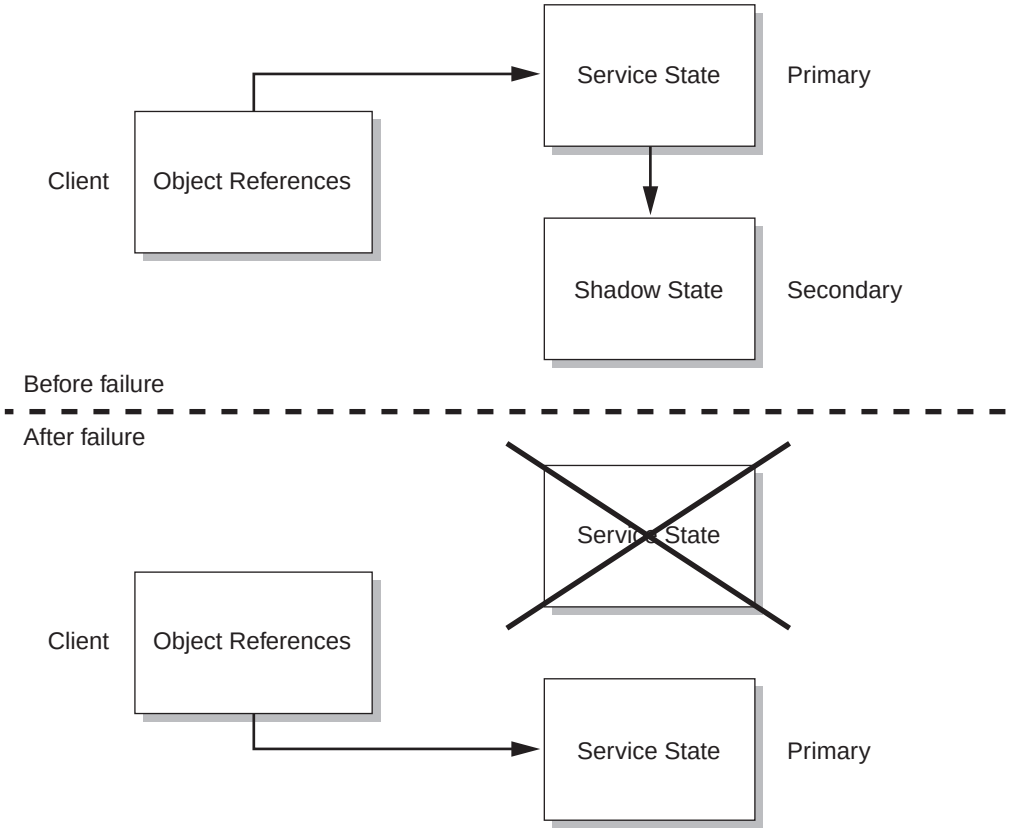

Figure 2-8 Replica Framework

Replica Manager (RM)

The replica manager is a logically centralized manager for all replicated services. It maintains a registry of services and providers for each service. It also selects the primary and secondary service providers based on configured parameters for services.

To make the RM highly available, the RM consists of a primary and secondary like other services. If the primary RM fails, the secondary RM can take over without any noticeable loss of service.

Replica Manager Manager (RMM)

The RMM uses the same replica framework to ensure that the replica manager is also highly available. If the node hosting the RM is down, the RMM can select a new node to host the Replica Manager.

Replica Manager Agent (RMA)

The RMA is active on each node and is registered with the RM during a node boot. It performs operations on behalf of the primary RM.

Debugging the Replica Framework

Most of the Replica Framework's debug buffers are not enabled in the Sun Cluster 3.0 released binary. However, the RM's debug buffer, `rm_dbg_buf`, contains all state changes and important events. This buffer may be accessed using the same procedure described in the section named "Transport Example: Accessing the Path Manager Debug Buffer on a Running System" on page 2-11:

```
# adb -k /dev/ksyms /dev/mem
*rm_dbg_buf/s
```

The output from this command is lengthy and can be redirected to a file:

```
# adb -k
>/tmp/buf.out
*rm_dbg_buf
$q
```

Instructor Note: You have completed all of the components on Figure 1-3. Point this out to the students and refer back to Figures 1-2 and 1-3 again. The next section, CCR, is on Figure 1-2.

Cluster Configuration Repository (CCR)

The CCR is the repository for Sun Cluster 3.0 global persistent configuration data. Updates to the CCR are guaranteed to be globally consistent, provided the updates are made through the CCR-exported interfaces. The CCR is similar to the CCD in Sun Cluster 2.x, except that it is accessible from kernel as well as userland users, supports multiple-update transactions, and supports multiple independent tables of data that may be updated simultaneously.

The CCR is a global repository with local copies on each node. It uses the replica framework to ensure that it is highly available.

CCR Components

The major CCR components are listed in the following table.

Table 2-3 Major CCR Components

Component	Function
Transaction Manager (TM)	Replicated HA service that provides write access to the repository.
Data Server (DS)	Manages persistent data and metadata and provides direct read access. There is one DS on each node.
Directory (DIR)	Client interface to the CCR that forwards read / write requests to the DS / TM.

CCR Data Representation

The CCR is represented as a collection of tables associated with persistent files.

CCR Tables

CCR tables are represented in the file system by ASCII text files, in the directory `/etc/cluster/ccr`. Each file starts with a generation number, `ccr_gennum`, and a checksum, `ccr_checksum`. The generation number

ccr_gennum indicates the current generation number of the CCR table file and is used to keep track of the most current version of the CCR table file among the different nodes in the cluster.

The checksum ccr_checksum indicates the checksum of the CCR table contents and provides a consistency check of the data in the table. The sample did_types file below contains an example of a CCR file whose ccr_gennum is 0 and ccr_checksum is: E1515A7592728B2A3E79BC685340125F.

Sample CCR Table: did_types

```
ccr_gennum      0
ccr_checksum    E1515A7592728B2A3E79BC685340125F
disk    1|1-3|rdsk|s2|0|100|1|1|0:s0,raw:a,raw:c|1:s1,raw:b,raw:c|2:s2,
raw:c,raw:c|3:s3,raw:d,raw:c|4:s4,raw:e,raw:c|5:s5,raw:f,raw:c|6:s6,raw:g
,raw:c|7:s7,raw:h,raw:c|0:s0,blk:a:b|1:s1,blk:b:b|2:s2,blk:c:b|3:s3,blk:d
:b|4:s4,blk:e:b|5:s5,blk:f:b|6:s6,blk:g:b|7:s7,blk:h:b

tape    2|1-3|rmt|l|1|10|8191|-1|0:,tp::c|1:b,tp:b:c|2:bn,tp:bn:c|3:
c,tp:c:c|4:cb,tp:cb:c|5:cbn,tp:cbn:c|6:cn,tp:cn:c|7:h,tp:h:c|8:hb,tp:hb:c
|9:hbn,tp:hbn:c|10:hn,tp:hn:c|11:l,tp:l:c|12:lb,tp:lb:c|13:lbn,tp:lbn:c|1
4:ln,tp:ln:c|15:m,tp:m:c|16:mb,tp:mb:c|17:mbn,tp:mbn:c|18:mn,tp:mn:c|19:n
,tp:n:c|20:u,tp:u:c|21:ub,tp:ub:c|22:ubn,tp:ubn:c|23:un,tp:un:c
```

CCR Infrastructure Table

The CCR cluster infrastructure table, /etc/cluster/ccr/infrastructure, contains most of the primary configuration data and properties for the low level cluster infrastructure, transport, and quorum devices. It provides a good snapshot of the cluster configuration.

Other CCR Files

The /etc/cluster/ccr directory contains an epoch file that indicates the cluster sequence number at the most recent time that the repository copy was part of the cluster. The cluster sequence number is the number of times the cluster has started and established quorum since the time of installation.

It is used by CCR recovery at the time a cluster is restarted to determine which nodes were not part of the previous cluster configuration, and therefore may not own the latest copy of ccr tables.

The /etc/cluster/ccr directory also contains a directory file that contains information about each CCR data table.

Manual Modifications to the CCR

Although manual modification to the CCR tables is strongly discouraged, there may be cases where it is necessary to make an emergency repair. The Sun Cluster Diagnostic Tool Kit contains a tool, ccradm, that can be used to make the repair. Directions for using this tool are contained in a later module.

Debugging the CCR

The CCR uses the ccr_dbg debug buffer to log diagnostic information. This buffer may be accessed using the same procedure described in the “Transport Example: Accessing the Path Manager Debug Buffer on a Running System” on page 2-11:

```
# adb -k /dev/ksyms /dev/mem
*ccr_dbg/s
```

Before moving on to the next section, refer back to Figure 1-2 and note that you have covered all of the sections on internal infrastructure. The section, CVM, is part of the User Visible Services in the kernel but is provided and supported by Veritas. Gloss over that and proceed to Scalable Networking.

Scalable Networking

Scalable Networking is responsible for the following functions:

- Maintaining a packet distribution table (PDT) at the Global Interface (GIF) node for each scalable service
- Distributing packets to nodes from the GIF based on the PDT. Packets belonging to a TCP session are always sent to the same instance if the node is functioning
- Maintaining a list of registered services
- Maintaining a list of instances bound / listening on which nodes
- Cleaning up state after node failures

Packet Distribution Table (PDT) Server

The PDT Server is the primary component of scalable networking. Its interactions with external and internal components can be seen in the following figure.

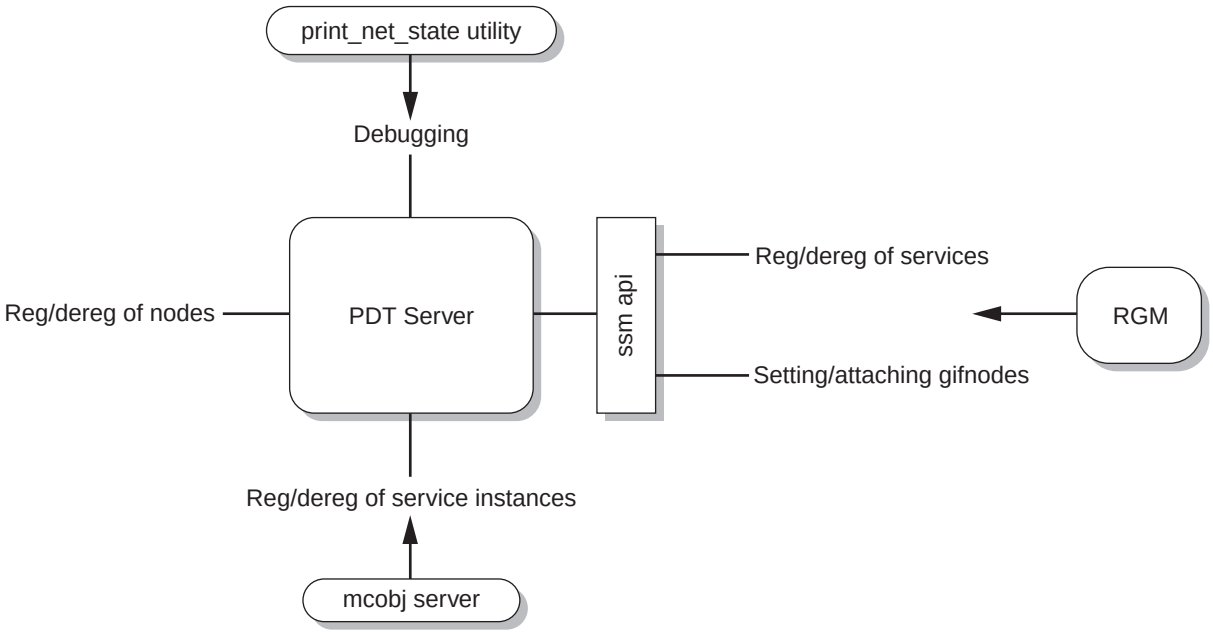

Figure 2-9 PDT Server

Packet Distribution Table (PDT)

The PDT is a hash table that maps the source address and port to a cluster node id. Packets from the source will be directed to the node id for eventual processing. There are 1009 buckets, and the buckets are filled based on the load-balancing weights.

By default, the packets are distributed evenly to all the nodes so the buckets will be filled evenly. For example, on a two-node cluster, 505 buckets will be assigned to node 1 and 504 buckets will be assigned to node 2. If a third node joins the cluster, the table will be adjusted so that one node is assigned to 337 buckets and the other two nodes are assigned to 336 buckets each.

Bucket #	0	1	2	3	4	5	6	7	8	9	10
Node ID	2	1	1	1	2	1	2	1	2	2	2

Table 1: PDT With 2 Instances

Bucket #	0	1	2	3	4	5	6	7	8	9	10
Node ID	2	1	1	***3***	2	***3***	2	1	***3***	***3***	2

Table 2: PDT With 3 Instances

Figure 2-10 Sample Packet Distribution Table

Forwarding List

The PDT Server maintains a forwarding list that is used when the PDT is changed, usually due to a reconfiguration. The list guarantees that packets belonging to a connection go to the same node even if the PDT table now points to a different node. The forwarding entry is purged when the connection is finished.

The forwarding list is not implemented as a hash table which makes it much slower. Therefore, it is best to avoid dynamic changes to the PDT.

Debugging Scalable Networking

The Sun Cluster Diagnostics Tool Kit contains the `print_net_state` utility, which is used to examine the contents of the PDT. This utility is discussed in greater detail in a later module.

There are also two debug buffers available for collecting diagnostic information:

- `net_dbg` – Records all networking information except forwarding information
- `net_dbg_fwd` – Records forwarding information

The `net_dbg` buffer is enabled by default. To disable it, add the following line to `/etc/system` and reboot the node:

```
set cl_net:clnet_net_dbg=0
```

The `net_dbg_fwd` buffer is not enabled by default. To enable this debug buffer, add the following line to /etc/system and reboot the node:

```
set cl_net:clnet_net_dbg_fwd=1
```

These buffers may be accessed using the same procedure described in the section named Transport Example: Accessing the Path Manager Debug Buffer on A Running System:

```
# adb -k /dev/ksyms /dev/mem
*net_dbg/s
*net_dbg_fwd/s
```

Again, refer back to Figure 1-2 before proceeding to the next section, HA Global File Systems and Devices.

High-Availability (HA) Global File Systems and Devices

Most of the following sections in global devices should be review for the students. Emphasize the sections on debugging.

HA Global Devices

Sun Cluster uses global devices to provide cluster-wide, highly available access to any device in a cluster, from any node, without regard to where the device is physically attached. If a node fails while providing access to a global device, Sun Cluster automatically discovers another path to the device and redirects the access to the path.

Sun Cluster global devices include disks, CD-ROMs, and tapes. However, disks are the only supported multiported highly available global devices.

Device ID (DID)

Sun Cluster manages global devices through the device ID (DID) pseudo driver.

The DID driver probes all nodes of the cluster and builds a list of unique disk devices, assigning each a unique major and minor number that is consistent on all nodes of the cluster. Access to the global devices is performed utilizing the unique device ID instead of the traditional Solaris device IDs.

Disk Device Groups

In Sun Cluster, all multihost disks must be under the control of the Sun Cluster Framework. An administrator first creates the volume manager disk groups with Solstice DiskSuite or VERITAS Volume Manager, and then registers the disk groups as Sun Cluster disk device groups.

Remember that disk device groups are independent of resource groups in Sun Cluster 3.0.

Global Namespace

The Sun Cluster mechanism that enables global devices is the global namespace.

The global namespace includes the /dev/global hierarchy as well as the volume manager namespace.

The following table shows the mappings between the local and global namespaces for a multihost disk, c0t0d0s0.

Table 2-4 Local and Global Namespaces Mappings

Component/Path	Local Node Namespace	Global Namespace
Solaris logical name	/dev/dsk/c0t0d0s0	/global/.devices/node@*ID*/dev/dsk/c0t0d0s0
DID name	/dev/did/dsk/d0s0	/global/.devices/node@*ID*/dev/did/dsk/d0s0
Solstice DiskSuite	/dev/md/*diskset*/dsk/d0	/global/.devices/node@*ID*/dev/md/*diskset*/dsk/d0
VERITAS Volume Manager	/dev/vx/dsk/*disk-group*/v0	/global/.devices/node@*ID*/dev/vx/dsk/*disk-group*/v0

The global namespace is automatically generated on installation and updated with every reconfiguration reboot. However, you can also generate it by running the scgdevs command.

Device Configuration System (DCS)

The DCS maintains a list of device services. It specifies the nodes that can host these devices and their relative preferences. It uses the userland program, clexd, to perform transitions.

DCS CCR Tables

Service classes are stored in the /etc/cluster/ccr/dcs_service_classes file:

```
ccr_gennum      0
ccr_checksum    11671B95EEEE2141D6F7446238541465
SUNWmd  /usr/cluster/lib/sc/run_reserve -C SUNWmd:transparent
DISK    /usr/cluster/lib/sc/run_reserve -C DISK:transparent
TAPE    /bin/true:eio
SUNWvxvm        /usr/cluster/lib/sc/run_reserve -C SUNWvxvm:transparent
```

Details about registered services are stored in /etc/cluster/ccr/dcs_service_<n> files, where <n> is an integer beginning at 1 and ending at the total number of available services:

```
ccr_gennum      3
ccr_checksum    8AFE51C67CAC91D1B4368A395A3EC87F
DCS_ServiceName dsk/d15
DCS_ServiceClass        DISK
DCS_FailbackEnabled     no
DCS_Suspended   no
DCS_NumberOfSecondaries 0
DCS_Nodes       (1,0) (2,0)
DCS_Devices     (did,480-487)
gdev    d15
autogenerated   1
```

If anyone asks, DCS_Suspended indicates whether the device is in maintenance or not.

Debugging Global Devices

The following sections describe common issues concerned with troubleshooting DID and device groups.

Troubleshooting DID:

Two of the most common issues with global devices are:

1. Failure to open a global device
   - Make sure the file is created and the device service is registered.
   - Rerun scgdevs to be sure.
2. Using a non-global device for a global mount
   - Make sure the /etc/vfstab file does not contain /dev/rdsk devices for global filesystems.

Troubleshooting Device Groups:

Here is a recommended procedure for troubleshooting device groups:

1. Check /var/adm/messages for errors.
2. Check device group status with scstat.
3. Check volume manager status (metaset, vxdisk list).
4. Check volume manager version and patches.

5. Check volume manager daemons (`rpc.metad`, `vxconfigd`).
6. Check volume manager configuration database (`metadb`, `rootdg`).
7. Check device and global device namespace (`/usr/cluster/bin/scgdevs`, `/usr/cluster/lib/dcs/scvxvmlg`).

In addition, verify that root is in group 14 in `/etc/group` if you are running SDS. If you are running VxVM, verify that all disks have slice 2 set to the entire disk and that Dynamic MultiPathing (DMP) is disabled.

HA Global File Systems

The following sections describe the Cluster File System (CFS), Proxy File System (PXFS), CFS mounts, and debugging PXFS.

Cluster File System

A cluster file system is a proxy between the kernel on one node, and the underlying file system and volume manager running on a node that has a physical connection to the disk(s).

Cluster file systems are dependent on global devices with physical connections to one or more nodes. However, the global devices can be accessed from any node in the cluster, whether or not that node has a physical connection to the storage device.

Proxy File System (PXFS)

The CFS is based on the Proxy File System. PXFS is built on top of the existing Solaris file system at the vnode interface. This enables PXFS to be implemented without extensive kernel modifications.

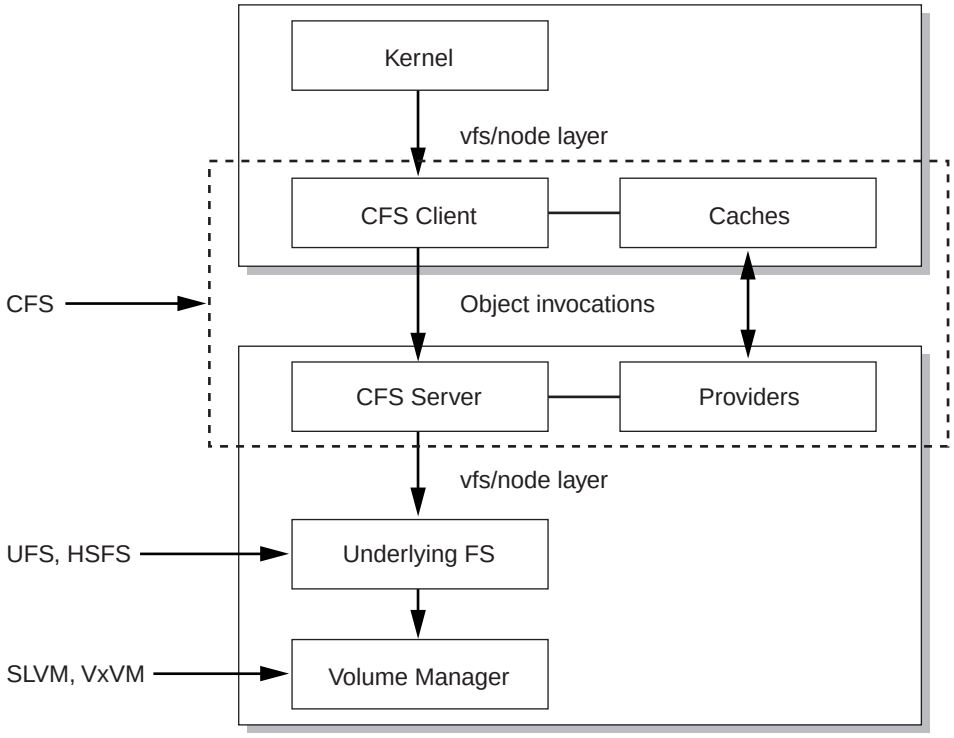

Figure 2-11 CFS Implementation

CFS Mounts

In a non-cluster UFS mount, the mount system call performs the following functions:

- Locks the mount point with `vn_vfswlock()`
- Loads UFS file system module
- Locks the `vfs_t` with `vfs_lock()`
- Calls `UFS VFS_MOUNT()` to initialize `vfs_t`
- `UFS VFS_MOUNT()` does "busy" checks on mount point
- Links `vfs_t` into the file system name space
- Unlocks `vfs_t` and mount point

In a cluster CFS mount, the same mount system call performs the following functions:

- Locks the mount point with `vn_vfswlock()`
- Loads PXFS file system module
- Locks the `vfs_t` with `vfs_lock()`
- Calls `PXFS VFS_MOUNT()` to initialize `vfs_t`
- `PXFS VFS_MOUNT()` does "busy" checks on mount point
- Links `vfs_t` into the file system name space
- Unlocks `vfs_t` and mount point

The `PXFS VFS_MOUNT()` performs the following additional functions:

- Reads the mount arguments into the kernel
- Gets a list of nodes the disk is attached to from the device configuration system (DCS)
- Calls the mount server to perform the global mount
- Mount server calls the mount client to lock the mount point and do "busy" checks on the other nodes
- Mount server calls mount client to create file system server replicas on the nodes with physical disk connections
- File system server gets a `become_primary()` call from the HA framework which calls `UFS VFS_MOUNT()` on the nodes with the physical disk connections
- Mount server calls mount client to create a proxy `vfs_t`

Debugging PXFS

PXFS uses the `pxfs_dbg` debug buffer to log diagnostic information. This buffer may be accessed using the same procedure described in the section named “Transport Example: Accessing the Path Manager Debug Buffer on a Running System” on page 2-11:

```
# adb -k /dev/ksyms /dev/mem
*pxfs_dbg/s
```

The output in this buffer is cryptic and does not contain much detail.

Before moving on to the next section, refer back to Figure 1-2 and note that you have covered all of the sections on User Visible Services in the kernel.

The last section, RGM, is the only cluster component in Userland.

Resource Group Manager (RGM)

The Resource Group Manager is a distributed application that provides the mechanism to make cluster resources highly available and scalable. It automatically starts and stops resources on selected cluster nodes in the event of hardware or software failures or reboots.

The Resource Group Manager is controlled by the `rgmd` daemon which is a multi-threaded daemon that runs in real time and is started by `/etc/rc3.d/S29initrgm` when the node boots in cluster mode.

If `rgmd` dies or is killed, the node panics to avoid data corruption and ensure a consistent view of cluster membership.

The lowest numbered node in a cluster is considered the President and all other nodes are Slaves.

RGM President and Slave Functions

The President is responsible for:

- Initiating and coordinating the process of bringing resource groups online and offline on different nodes
- Communicating orders to the Slaves

The Slaves are responsible for:

- Receiving orders from the President to bring resource groups online or offline on their nodes
- Requesting the President to fetch the current state information from their nodes

Note – The President may assume the role of both President and Slave.

The rgmd Threads

The rgmd is implemented as multiple threads as shown in the following table.

Table 2-5 The rgmd Threads

Thread Name	Function
receptionist	Receives rpc requests from API library routines
Userland Cluster Membership Monitor (UCMM) sender thread	Sends periodic broadcasts to rgmd daemons on other nodes
rgfreeze thread	Runs on the President node when the replica manager agent sends notification of a frozen or unfrozen service, such as a diskset becoming available
state machine thread	Runs on the Slave whenever the President requests a change in mastery of a resource group or informs Slaves of a configuration change.
wakeup thread	“Wakes up” the President when a Slave wants to inform it of state changes on the slave
failback thread	Runs on the President at the end of a cluster reconfiguration to relocate resource groups to more preferred masters if their failback properties are set to true
launch_method thread	runs whenever rgmd needs to run a resource callback method such as start, stop, validate, etc.

RGM Components

The major components of the RGM are shown in the following table.

Table 2-6 RGM Components

Component	Function
Userland Cluster Membership Monitor (UCMM)	Maintains CMM information gathered from the kernel CMM to enable the RGM to receive notification about connectivity changes such as node failures and rejoins
receptionist	The `rpc` server to receive `rpc` requests from API library routines
`rgm_comm`	Communicates with the `rgmds` running on other cluster nodes
fork execution daemon (FED)	Executes the resource callback methods
state machine	Core abstract component that drives the steps to take resource groups online and offline upon request from the President node

Tunable Properties

The behavior of the RGM can be influenced by modification to the following two resource group properties:

`Pingpong_interval`

The `Pingpong_interval` is a resource group property that is specified as a non-negative number of seconds. It defaults to 3600 seconds. The RGM uses this property when determining where to bring a resource group online after a cluster reconfiguration and when determining where to relocate a resource and its containing resource group.

If a resource group has failed to come online more than once within the past `Pingpong_interval` seconds on node *n*, then that node is considered ineligible to host the resource group.

The Failback property

The Failback property is a resource group property that is either `true` or `false`. The RGM uses this property when determining whether to recalculate the set of nodes where the group is online when the cluster membership changes.

If `true`, the set of primaries for the resource group will be recomputed whenever nodes join the cluster, and the RGM may switch the resource group from the current primary to a more preferred node.

If `false`, the resource group will remain on the existing primary, even if more preferred primaries join the cluster.

Debugging `rgmd`

The Diagnostic Tool Kit provides the `rgmd_debug` command to enable verbose debugging and examine the contents of the rgmd debug buffer. The command is explained in more detail in a later module.

Module 2 Lab

The students will be using adb to display debug buffers. Some of the students may be quite fluent with adb. Caution those students not to use adb to make any changes to the kernel. Other students should need no knowledge of adb to perform the lab.

This lab provides practice accessing the cluster components' debug buffers.

1. If you have not already done so, access some of the debug buffers referred to in the module on one of the cluster nodes. Follow the procedure outlined in the "Transport Example: Accessing the Path Manager Debug Buffer on a Running System" on page 2-11.

   Be sure to access the following buffers:

   `pm_dbg`

   `cmm_dbg_buf`

   `rm_dbg_buf`

   `ccr_dbg`

   `net_dbg`

2. Take down the other cluster node and access the same debug buffers. Which buffers recorded the downed node?

   _____

   _____

3. Boot the downed cluster node and access the same debug buffers. Which buffers recorded the rejoined node?

   _____

   _____

4. Disconnect one of the private interconnect cables. Access the same debug buffers.

   a. Which buffers recorded the interconnect failure?

   _____

   _____

   b. Is this what you expected? yes / no

   c. Why or why not?

   _____

_____

5. Reconnect the interconnect cable.

6. Disconnect one of the public network connections. Access the same debug buffers.

   a. Which buffers recorded the public network failure?

   _____

   _____

   b. Is this what you expected? yes / no

   c. Why or why not?

   _____

   _____

Module 3

Using Diagnostic Tools

Objectives

This module introduces the Sun Cluster Diagnostics Tool Kit and other cluster-specific diagnostic tools.

By the end of this module you should be able to:

- List the cluster-specific diagnostic tools
- Install the Sun Cluster Diagnostic Tool Kit
- List the components of the Sun Cluster Diagnostic Tool Kit
- Use Sun Cluster Diagnostic Tool Kit commands to troubleshoot a cluster
- Use Explorer to collect system information

Additional Resources

Additional resources – The following references can provide additional details on the topics discussed in this module:

- Sun Cluster 3.0 Diagnostic Tool Kit
  http://suncluster.eng/service/tools/scdtk.html
- Explorer Data Collection Tool Web Site
  http://eservices.central/knowledge/products/explorer

Sun Cluster DiagnosticTools

The cluster-specific tools for diagnosing cluster problems are:

- Sun Cluster 3.0 Diagnostics Tool Kit – A collection of documentation and tools for diagnosing cluster problems
- Command Line Utilities – A collection of command line utilities provided with Sun Cluster 3.0

Other tools that are also useful for diagnosing cluster problems include:

- Explorer – A tool to gather detailed system information

Instructor note: There are many commands in the DTK, and it is probably not worth going into detail on each one. While they are all listed here for completeness, emphasis should be placed on `ccradm`, `chkinfr`, `print_net_state` and `scsi`. Also cover the `scdtk_enable` and `scdtk_collect` man pages. Definitely spend some time going over the live crash dump section as it will be a common request from the CTE engineers to the ES support staff.

Sun Cluster 3.0 Diagnostics Tool Kit

The Diagnostic Tool Kit (DTK) is an add-on package available for Sun Cluster 3.0. This package contains additional diagnostic tools and documentation that can help service personnel troubleshoot cluster problems. The tools in the DTK will produce and collect debugging information. The tools do not analyze the data or attempt to diagnose the problem.

Note – The DTK is available to field service personnel as a downloadable package and should only be used by qualified cluster engineers or in conjunction with instructions from CTE/Engineering.

Installing the Diagnostic Tool Kit (DTK)

Use the following steps to obtain and install the DTK:

1. Download the DTK from:

`http://suncluster.eng/service/tools/scdtk.html`

The version of the DTK should match the version of the base Sun Cluster package. In some cases, the version of the DTK may be behind the Sun Cluster release. In that case, select the most recent version of the DTK.

To determine which version to download type the following command for each of the cluster nodes:

```
# pkgparam SUNWscr VERSION
```

Your output will display something similar to:

```
3.0.0,REV=2000.10.01.01.00
```

In this example the correct version to download would be the GA Release.

2. Copy the downloaded file to `/tmp`:

```
# cp SUNWscdtk_GA.tar.Z /tmp
```

3. Change to the `/tmp` directory:

```
# cd /tmp
```

4. Uncompress the file:

```
# zcat /SUNWscdtk_GA.tar.Z | tar xvf -
```

5. Use pkgadd to install the DTK:

```
# pkgadd -d . SUNWscdtk
```

6. If the package installs successfully, remove /tmp/SUNWscdtk:

```
# cd /tmp; rm -rf SUNWscdtk
```

7. Set your MANPATH to view the DTK man pages:

Bourne or Korn Shell users type:

```
# MANPATH=$MANPATH:/usr/cluster/dtk/man
# export MANPATH
```

CShell users type:

```
# setenv MANPATH "/usr/cluster/dtk/man:$MANPATH"
```

Using the DTK

The DTK contains the tools listed in Table 2-1. In some cases, the tool is included with the Sun Cluster 3.0 release and the DTK only contains the documentation for the tool. DTK-provided tools are always installed in /usr/cluster/dtk/bin by default.

Table 3-1 Sun Cluster Diagnostic Tool Kit Tools

Tool	Location	Function
ccradm	/usr/cluster/lib/sc	Administers CCR files
chkinfr	/usr/cluster/lib/sc	Validates infrastructure file contents
cmm_ctl	/usr/cluster/dtk/bin	Administers cluster membership control
dcs_config	/usr/cluster/dtk/bin	Queries Device Configuration System (DCS)
ddb	/usr/cluster/dtk/bin	Dumps debug buffers

Table 3-1 Sun Cluster Diagnostic Tool Kit Tools (Continued)

Tool	Location	Function
`orbadmin`	`/usr/cluster/dtk/bin`	Displays ORB state
`print_net_state`	`/usr/cluster/dtk/bin`	Prints networking state
`replctl`	`/usr/cluster/dtk/bin`	Provides an interface to the replica manager
`scsi`	`/usr/cluster/lib/sc`	Queries and clears SCSI reservations
`rgmd_debug`	`/usr/cluster/lib/sc`	Facilitates debugging of rgmd
`scdtk_enable/ scdtk_collect`	`/usr/cluster/dtk/man/ man5`	Man pages that describe diagnostic techniques for data services and collecting core data

The sections that follow will describe each command in detail and illustrate some typical uses.

Before moving on to the next section, ask the students to do Lab 1, which installs the DTK. As you move through the commands, demo the easier commands on screen and let the students play with each one for a few minutes. Good choices for the demos include print_net_state, chkinfr, cmm_ctl, orbadmin, replctl, and rgmd_debug printbuf. Let the students take a few minutes to try each command after you demo it.

The ccradm command

Review the ccradm commands with the students, but ask them not to play with them until the labs.

The ccradm command is an *emergency* command line interface designed only to be used when a Cluster Configuration Repository (CCR) table is corrupted or a cluster node cannot boot due to loss of operational quorum.

In normal operations, if an update is made to a CCR file, the ccr_gennum is incremented by 1, and the ccr_checksum is recomputed. The change is made to all active nodes in the cluster, ensuring that each node has consistent data.

If, however, the file must be manually edited as part of an emergency procedure, the checksum must be recomputed to make the data consistent. The ccradm tool can be used to recompute the checksum and set the generation number.

Using ccradm

The most common form of the ccradm command is:

usr/cluster/lib/sc/ccradm –i *ccr_table_file* [**–o**]

Substitute the actual name of the name of the CCR file for *ccr_table_file*:

usr/cluster/lib/sc/ccradm –i /etc/cluster/ccr/did_types [**–o**]

The -o option designates one CCR table file to be the master copy. This version of the CCR table file will override other versions of the file that are on the remaining nodes.

If the -o option is used on multiple nodes, such as from the cluster console, only one of the node's copies will be used as the master copy.

Generic Repair Procedure

The following is a generic repair procedure for fixing corrupted CCR tables:

1. Reboot all nodes in non-cluster mode (boot -x).
2. Edit the file /etc/cluster/ccr/<ccr_table_file> on all nodes to contain the correct data. The file must be identical on all nodes.

3. On all nodes, recompute the checksum and designate the edited CCR table file to be the override version.

```
    pnode1# /usr/cluster/lib/sc/ccradm -i /etc/cluster/ccr/
<ccr_table_file> -o
    pnode2# /usr/cluster/lib/sc/ccradm -i /etc/cluster/ccr/
<ccr_table_file> -o
    pnode3# /usr/cluster/lib/sc/ccradm -i /etc/cluster/ccr/
<ccr_table_file> -o
    pnode4# /usr/cluster/lib/sc/ccradm -i /etc/cluster/ccr/
<ccr_table_file> -o
```

The `-i` option causes `ccradm` to recompute the checksum and update the `<ccr_table_file>` with the new checksum.

The `-o` option tells the cluster to use the local edited file as the master copy and override any other versions of the file that are on the remaining nodes. The table's generation number will be reset to 0 after recovery.

4. Reboot all nodes in cluster mode.

Recovering from Loss of Operational Quorum in a Two Node Cluster

Amnesia can occur when all the nodes leave the cluster in staggered groups. In a two-node cluster consisting of nodes A and B, this can occur if node A leaves the cluster and then node B continues for a period of time before leaving.

If node B updated the CCR while node A was down, node A will not have the current copy of the CCR, and is said to have "amnesia." Even if node B made no changes to the CCR while node A was down, node A cannot tell whether or not its copy of the CCR is current.

Quorum devices ensure that node A will not be able to start up with a stale CCR by assigning the SCSI persistent group reservation to node B. This means that node A will not be permitted to start up until node B rejoins the cluster.

Attempts to boot node A will result in node A hanging and "`waiting operational quorum`".

This would normally be considered desirable. However, if node B cannot rejoin the cluster due to hardware failures or other reasons, node A must be able to rejoin the cluster or the entire cluster will remain down.

The ccradm tool can be used to allow node A to reboot by forcing the cluster to use node A's CCR tables as the master copy as follows:

1. Reboot node A in non-cluster mode (boot -x).
2. Edit the /etc/cluster/ccr/infrastructure file to set the quorum_vote to 1 on node A and 0 on node B and the quorum device(s):

```
cluster.nodes.1.properties.quorum_vote  1
cluster.nodes.1.properties.quorum_resv_key      0x3A104E7400000001
cluster.nodes.2.properties.quorum_vote  0
cluster.nodes.2.properties.quorum_resv_key      0x3A104E7400000002
cluster.quorum_devices.2.properties.votecount 0
```

If you are not sure which node is node 1 and which node is node 2, check the line entry for cluster.nodes.1.name:

```
cluster.nodes.1.name nodeA
```

3. Save the file and exit.
4. Use ccradm to make node A's copy of the infrastructure file the master copy and to recompute the table's checksum:

```
/usr/cluster/lib/sc/ccradm -i /etc/cluster/ccr/infrastructure -o
```

5. Reboot node A.
6. Reboot node B as appropriate.

Warning – This procedure will remove any changes made to node B's infrastructure file while node A was down.

Note – The ccradm command has other additional options that are listed on its man page. This is a dangerous command and should only be used if you are absolutely certain that you know what you are doing. Read the man page thoroughly before attempting any repair procedures. Appendix A contains the man page for ccradm.

The chkinfr Command

The chkinfr command validates the contents of the /etc/cluster/ccr/infrastructure file.

Specifically, chkinfr verifies the information about nodes, adapters, switches, cables, and quorum devices in the infrastructure file. It uses the .clpl files in the /etc/cluster/clpl directory to validate the supported adapters (for example, hme, qfe, ge), supported transport types (such as dlpi, rsm) for each type of adapter, and supported junction types (for example, switch).

The chkinfr command is invoked at boot time by the rc script /etc/rcS.d/S41bootcluster.sh before the cluster software is loaded. Any errors detected are displayed on the console and also stored in the /etc/cluster/chkinfr.err file. The system is halted if any errors are detected so that the administrator can boot into non-cluster mode to repair the file.

The chkinfr command can also be invoked from the command line to check the infrastructure file. The chkinfr command supports the options listed in Table 3-2.

Table 3-2 The chkinfr Options

Options	Function
-F	Forces the command return value to 0. Can be used to continue booting if chkinfr in the rc script reports errors
-i file	Validates specified file instead of the /etc/cluster/ccr/infrastructure default
-p directory	Searches for the .clpl files in specified directory instead of the default /etc/cluster/clpl

Using chkinfr From the Command Line

To use chkinfr from the command line, log in as root on a cluster node and type:

```
# /usr/cluster/lib/sc/chkinfr
```

The command will execute and exit with no output if the check is successful.

Errors, if any, are displayed onscreen and stored in the same `/etc/cluster/chkinfr.err` file used during the boot process:

```
# /usr/cluster/lib/sc/chkinfr
Error : Duplicate node name - pnode2
```

Appendix A contains the man pages for `chkinfr` (`chkinfr(1M)`) and the infrastructure file (`ccr_infrastructure(4)`). The man page `chkinfr (1M)` includes a list of possible causes where `chkinfr` reports errors, and `ccr_infrastructure(4)` includes a description of the `infrastructure` file's format as well as a sample file.

Note – The `chkinfr` command does not check for misspellings or typographical errors in the file. Its purpose is solely to verify that the information about the nodes, adapters, switches, cables, and quorum devices entries is correct.

The cmm_ctl Command

The cmm_ctl command is used to control and obtain information about the cluster membership.

This command is particularly useful for obtaining information about the state of the cluster, including details about Cluster Membership Monitor (CMM) reconfiguration, the various CMM configuration timeouts, and quorum-related information.

Using cmm_ctl to Obtain Cluster Membership State

To use cmm_ctl to obtain information about the state of the cluster, log in as root on a cluster node and type:

```
# /usr/cluster/dtk/bin/cmm_ctl -g
```

Your output will appear similar to the following:

```
*** Current Cluster State ***

        Local Node Id                   : 1
        Highest Node Id                 : 2
        Current reconfiguration seqnum  : 1
        Max reconfiguration step        : 11
        CMM is in the end state.

        *** Incarnation Numbers of Current Cluster Members ***

                Node 1  : 974153850
                Node 2  : 974153855

*** Cluster Configuration Information ***

        Max Nodes Supported     : 64
        Node Fenceoff Timeout   : 300 seconds
        Boot Delay              : 60 seconds
        Node Halt Timeout       : 5 seconds
        Failfast grace time     : 10 seconds
        Failfast panic delay    : 30 seconds
        Orb Stop Timeout        : 30 seconds
        Orb Return Timeout      : 30 seconds
        Orb Abort Timeout       : 30 seconds
        Orb Step Timeout        : 120 seconds
```

```
*** Cluster Quorum Information ***

        Total configured votes  : 3
        Current cluster votes   : 3
        Votes needed for quorum : 2

        *** Node quorum info ***

                Node 1  : state = UP, votes_configured = 1,
                          reservation_key = 0x3a104e7400000001
                Node 2  : state = UP, votes_configured = 1,
                          reservation_key = 0x3a104e7400000002

        *** Quorum device info ***

Quorum device 1 : global device name = '/dev/did/rdsk/d10s2',
                                  nodes with configured paths = 0x3,
                                  state = ONLINE, votes_configured = 1,
                                  reservation owner = node 1
```

The `cmm_ctl` tool also provides options to disable or enable transport path monitoring, disable or enable failfast, force a reconfiguration of the CMM, and force a failfast, stop or abort of a cluster node. See Appendix A for the `cmm_ctl` man page (`cmm_ctl(1M)`) and a list of all the available options.

The dcs_config Command

The dcs_config command provides a more direct way to query the DCS than the standard scconf(1M) interfaces. While update interfaces are available in this command, they are not supported, recommended or documented. All changes to the state of the DCS should be done using scconf.

Using dcs_config

There are two forms of the dcs_config command:

```
/usr/cluster/dtk/bin/dcs_config -c info [ -s service-name ]| [ -C
service-class ] | [ -d device-path ]

/usr/cluster/dtk/bin/dcs_config -c status [ -s service-name]
```

The -c info form provides general configuration information about the service, and the -c status form provides information about the service's current state. Using -c info or -c status commands without additional qualifying options shows all service classes and services in use.

Example: Using dcs_config to obtain information about all services and service classes.

To use dcs_config to determine which services and services classes are in use, log in as root and type the following command:

```
# /usr/cluster/dtk/bin/dcs_config -c info
```

Your output will appear similar to:

```
Services classes are :
    SUNWmd
    DISK
    TAPE
    SUNWvxvm
Services are :
    dsk/d1
    dsk/d2
    dsk/d3
    dsk/d4
    dsk/d5
    dsk/d6
    dsk/d7
```

```
    dsk/d8
    dsk/d9
    dsk/d10
    dsk/d11
    dsk/d12
    dsk/d13
    dsk/d14
    dsk/d15
    dsk/d16
    dsk/d17
    dsk/d18
    mirror-1
```

You can then gather more information about a particular service by typing:

```
# /usr/cluster/dtk/bin/dcs_config -c info -s dsk/d3
```

Your output will appear similar to:

```
Service name: dsk/d3
Service class: DISK
Switchback Enabled: False
Number of secondaries: All
Replicas: (Node id --> 1, Preference --> 0)(Node id --> 2, Preference -->
0)
Devices: (179, 96-103)
Properties:
        gdev --> d3
        autogenerated --> 1
```

You can also gather more information about a particular service class by typing:

```
# /usr/cluster/dtk/bin/dcs_config -c info -C TAPE
```

Your output will appear similar to:

```
Service class   --> TAPE
Takeover pgm    --> /bin/true
Failover type   --> eio
Services :
```

Example: Using dcs_config to Obtain Status of a Service

To obtain the status of a service, log in as root and determine the name of the service whose status you wish to obtain using the steps in the previous example.

To determine the status of the service `mirror-1`, type the following:

```
# /usr/cluster/dtk/bin/dcs_config -c status -s mirror-1
```

Your output will appear similar to:

```
Service Name: mirror-1
Active replicas: (1. State - Primary)(2. State - Secondary)
Service state: ONLINE
```

The ddb Command

The ddb command displays the threadlist and the contents of the kernel debugging buffers from either a running system or a system crash dump.

Using ddb

There is only one form of the ddb command:

```
/usr/cluster/dtk/bin/ddb kernel-symbols kernel-memory
```

The kernel-symbols component is usually /dev/ksyms for a running system or the UNIX file from a system crash dump.

The kernel-memory component is usually /dev/mem for a running system or the vmcore file from a system crash dump.

Example: Using ddb on a Running System

To use ddb on a running system, log in as root and type the following:

```
# /usr/cluster/dtk/bin/ddb /dev/ksyms /dev/mem
```

To redirect the output to a file:

```
# /usr/cluster/dtk/bin/ddb /dev/ksyms /dev/mem > /var/tmp/ddb.out 2>
/var/tmp/ddb.out
```

Example: Using ddb on a System Crash Dump

To use ddb on a system crash dump, type the following in the directory containing the system crash dump:

```
# /usr/cluster/dtk/bin/ddb unix.0 vmcore.0
```

To redirect the output to a file:

```
# /usr/cluster/dtk/bin/ddb unix.0 vmcore.0 > /var/tmp/ddb.out 2>
/var/tmp/ddb.out
```

The output from this command is lengthy and may include contents of the following debug buffers:

```
attr_dbg
    ccr_dbg
    ckpt_dbg
```

```
clconf_dbg
cmm_dbg_buf
dcs_dbg
er_dbg
flk_dbg
fp_dbg
mount_dbg
ns_dbg
```

Note – The ddb command uses adb macros and adb aborts macros when a symbol is not found. If your output ends with "`symbol not found`", the `/usr/cluster/dtk/lib/adb/dump_all` macro contains a symbol not found in the kernel symbols. This could be because a module was not loaded, or it could be because the `dump_all` macro is not matched to the version of SC3.0 being run.

The orbadmin Command

The orbadmin command is a general purpose tool for observing behavior in the ORB.

Using orbadmin

The general form of the command is:

```
/usr/cluster/dtk/bin/orbadmin [ -U ] [ -R opts ] [ -P opts ]
      [ -B opts ] [ -m opts ] [ -f opts ] [ -u opts ] [ -c opts ]
      [ -r opts ] [ -n node ] [ interval [ count ] ]
```

Table 3-3 summarizes the function of each option.

Table 3-3 The orbadmin Options

Flag	Function
-U	Display the length of the unref_threadpool's work queue, or in other words the number of unreference notifications that have not been processed.
-R opts	Display state refcount
-P opts	Display state resource_pool
-B opts	Display state resource_balancer
-m opts	Display message statistics
-f opts	Display flow control statistics
-u opts	Display unref_threadpool statistics
-c opts	Change flow control settings
-r opts	Read flow control settings
-n node	Restricts display of information to node instead of the default of displaying information for all nodes
Note: opts may be "all" or specific options that are listed in the man page in Appendix A.	

Example: Using orbadmin to Read Flow Control Settings

To read flow control settings, log in as root and type the following:

```
# /usr/cluster/dtk/bin/orbadmin -r all
```

Your output will appear similar to:

```
pool: 0
number_nodes: 2
threads_low: 158
threads_moderate: 30020
threads_high: 30020
threads_increment: 79
threads_minimum: 4029
```

Example: Using orbadmin to Change Flow Control Settings

To change flow control settings, use the `-c opts` flag; `opts` is one of:

`pool=xxx`	`resource_pool` number, defaults to `DEFAULT_POOL`. In the above example, this number is 0.
`low=xxx`	New value for threads low. This represents the number of threads below which the client will not voluntarily fall. It ensures that a client has some minimal number of threads for performing work. This reduces latency when a previously idle client requests server services.
`mod=xxx`	New value for threads moderate. This represents the dividing line between light and heavy numbers of threads. The system is more aggressive about creating / requesting server threads when the load is light.
`high=xxx`	New value for threads high. This represents the maximum number of threads. This value must be set such that the server node can really support this many active threads. If the value is set too high, flow control becomes effectively disabled, and the system load will reach the point where node failure results.
`min=xxx`	New value for threads minimum. This represents the number of threads a server should allocate on system startup and retain.

Any unspecified thread value remains unchanged.

Using the output from the previous example, to increase threads low from 158 to 200, log in as root and type:

```
# /usr/cluster/dtk/bin/orbadmin -c low=200
```

Note – As with any other command that modifies system settings, use this command with extreme caution and only when directed to by the Sun Cluster development team.

The print_net_state Command

The print_net_state command displays state information about clustering networking. The command is commonly used to display state information about scalable networking.

Using print_net_state

There are four forms of the command:

```
/usr/cluster/dtk/bin/print_net_state -p [ -v ] [ -P node ] [-G node ] -S
service

/usr/cluster/dtk/bin/print_net_state -s [ -P node ] [ -G node ] [ -S
service ]

/usr/cluster/dtk/bin/print_net_state -f node [ -v ] -G node -S service

/usr/cluster/dtk/bin/print_net_state -h -G node
```

Table 3-4 summarizes the function of each option. Appendix A contains the man page and the complete list of options.

Table 3-4 print_net_state Options

Option	Function
-p	Print PDT table
-s	Print service(s). Specifying -s without additional options prints all the services.
-f node	Print forwarding lists for node. A node of 0 indicates all nodes.
-h	Print hash tables.
-G node	Specifies the gif node as an integer (for example -G 0)
-S service	Specifies the scalable resource

Example: Using `print_net_state` to Display All Services

To display all clustered net services, log in as root and type the following:

```
# /usr/cluster/dtk/bin/print_net_state -s
```

Your output will appear similar to:

```
Services from PDTServer on node 0:
group name = apache-rs, cfginst 1 2, grpinst 1 2
        1009 PDT buckets, policy LB_WEIGHTED n1(1) n2(2)
    1. protocol=TCP, ipaddr=172.20.4.195, port=80, gifnode=1, srvinst 1 2
```

You can use the output from this command as input to the other forms of the command. The service is `apache-res`, and the `gifnode` is on node 1.

Example: Using `print_net_state` to Display the PDT

Use the output from the previous example as input to the following command:

```
# /usr/cluster/dtk/bin/print_net_state -p -v -S apache-rs
```

Your output will appear similar to:

```
PDTinfo from PDTServer 0 for service apache-rs
        PDT summary:
                node 1: 504 buckets
                node 2: 505 buckets
        PDT contents:
2 1 2 2 1 1 2 2 2 2 1 2 1 1 1 1 1 1 2 2 1 1 2 1 1 2 1 1 2 1 2 2 1 2 1 1 2
2 1 1
2 2 2 1 2 1 2 1 1 2 1 2 1 2 2 1 2 2 2 1 2 1 1 1 2 1 2 2 1 1 2 2 2 2 1 2 1
1 2 1
1 2 2 1 1 2 2 1 1 1 2 2 2 1 2 1 2 2 2 2 1 2 1 2 1 2 1 1 1 2 1 1 1 1 2 1 1
1 2 1
1 1 2 1 1 2 2 2 1 1 1 2 2 1 1 1 1 2 2 2 1 1 1 2 2 1 1 1 1 2 1 2 2 2 2 2 2
2 2 2
1 2 1 1 2 2 1 2 1 1 2 2 2 1 1 1 1 1 2 2 2 1 2 2 1 2 2 2 1 2 1 1 1 1 1 2 2
1 2 1 ...
```

The `replctl` Command

The `replctl` command provides an interface to the replica manager, which manages kernel-level HA services that are implemented using the replica framework.

The `replctl` tool is primarily a test utility for the replica manager and for services that are written using it. It can also be used to view the state of HA services.

Using `replctl`

The `replctl` command has three forms:

```
/usr/cluster/dtk/bin/replctl [ -d serv-desc [ repl-desc ] ]

/usr/cluster/dtk/bin/replctl [ -f ] -c serv-desc repl-desc state

/usr/cluster/dtk/bin/replctl -x serv-desc
```

Using the first form of the command without any options displays the state of all HA services.

Note – The second and third forms of the command are dangerous and should not be used outside of the product development team.

Appendix A contains the man page for `replctl` and details about the options.

Example: Using `replctl` to Obtain the State of All HA Services

To use `replctl` to view the state of all services, log in as root and type the following:

```
# /usr/cluster/dtk/bin/replctl
```

Your output will appear similar to:

```
SID  Service Description       Service State
10   repl_name_server          UP
     Replica Description       Replica State
     2                         PRIMARY
     1                         SECONDARY
```

```
SID  Service Description       Service State
11   ccr_server                UP
     Replica Description       Replica State
     2                         PRIMARY
     1                         SECONDARY

SID  Service Description       Service State
12   DCS                       UP
     Replica Description       Replica State
     2                         PRIMARY
     1                         SECONDARY

SID  Service Description       Service State
13   mount                     UP
     Replica Description       Replica State
     2                         PRIMARY
     1                         SECONDARY
```

In this example all of the Service States are UP, which means that the service has a primary. Table 2-5 lists the possible service states.

Table 3-5 Service State Definitions

State	Definition
UNINIT	The service exists but has never had a primary. This is usually a short lived state during service initialization.
UP	The service has a primary.
DOWN	The service was once up, but the primary died and there were no secondaries capable of taking over.
UNSTABLE	Transitory state which indicates that a change (for example, a switchover) is occurring to the state of the service.

The scsi Command

The scsi command provides a method of querying and clearing SCSI-2 and SCSI-3 Persistent Group reservations on cluster disks.

Using scsi

There is only one form of the scsi command:

```
/usr/cluster/lib/sc/scsi -c subcommand -d disk
```

Note – The DTK refers to the reserve command. While this command is available, the scsi command is preferred because it does not require clustering and can be used in non-cluster mode. The reserve (1M) command includes fencing interfaces that should not be used accidentally.

The disk must be specified using the /dev/did/rdsk/dxs2 where x is the appropriate d number (for example, /dev/did/rdsk/d1s2).

Table 3-6 The scsi -c Subcommands

Subcommand	Function
scrub	To scrub SCSI-3 reservations from the disk. This should not be done during normal cluster operation. Doing so may compromise data integrity.
release	To remove SCSI-2 reservations from the disk. This should not be done during normal cluster operation. Doing so may compromise data integrity.
status	To determine if this host has access to the disk. Prints 0 if the host has access, 1 if it does not.
inkeys	Print PGR registration keys on the disk.
inresv	Print PGR reservations on the disk.

Example: Using scsi to Scrub SCSI-3 Reservations From a Disk

Because SCSI-3 reservations are persistent and survive reboots, it may become necessary to remove a reservation. This is typically common during the initial setup and configuration if disks are moved around or changed.

To use the `scsi` command to scrub SCSI-3 reservations:

1. Log in as root on one of the cluster nodes that has physical access to the disk.
2. Determine the `did` name of the disk:

```
# /usr/cluster/bin/scdidadm -L
```

3. Verify that there is a SCSI-3 reservation on the disk. The following command assumes a disk with the full `did` name `/dev/did/rdsk/d2s2`.

```
# /usr/cluster/lib/sc/scsi -c inkeys -d /dev/did/rdsk/d2s2
Reservation keys(2):
0x3908a06d00000002
0x3908a06d00000001
```

4. Scrub the reservation:

```
# /usr/cluster/lib/sc/scsi -c scrub -d /dev/did/rdsk/d2s2
```

5. Verify that the command completed successfully:

```
# /usr/cluster/lib/sc/scsi -c inkeys -d /dev/did/rdsk/d2s2
Reservation keys(0):
```

Note – Do not use this command to correct loss of operational quorum problems, such as forcing the release of a node's reservation on a quorum disk. The purpose of this command is to scrub keys from disks that may contain reservations from a previous installation.

The rgmd_debug Command

The rgmd_debug command is a utility that facilitates debugging of the rgmd. This script permits the verbose debugging mode of the rgmd to be turned on or off at the local node. To enable or disable verbose debugging mode on multiple nodes, the script has to be invoked on each node.

When verbose debugging mode is turned on, the rgmd will log a large number of messages that will appear on the console and in /var/adm/messages.

The rgmd_debug command also allows the trace buffer in the rgmd to be emptied. The trace buffer has essentially the same information as the verbose debugging messages generated when debugging mode is turned on. When the trace buffer is full, the oldest messages are removed.

Each trace buffer entry has the thread ID of the rgmd thread that generated the message, and the last 8 digits of a microsecond timestamp.

Using rgmd_debug

The general form of the rgmd_debug command is:

```
/usr/cluster/lib/sc/rgmd_debug on | off | printbuf
```

Use on and off to enable or disable verbose debugging modes. Use printbuf to print out the rgmd_dbg buffer.

Example: Using rgmd_debug to Enable or Disable Verbose Debugging Mode

To use rgmd_debug to enable or disable verbose debugging mode:

1. Log in as root on each node that you wish to enable.
2. Type the following command on each node to enable debugging:

```
# /usr/cluster/lib/sc/rgmd_debug on
    rgmd verbose debugging is ON
```

3. Start a tail -f on the /var/adm/messages file on each node to view messages:

```
# tail -f /var/adm/messages
```

4. When you are finished viewing messages, press Ctrl-C on each node to stop the tail process.
5. Turn off debugging by typing the following on each node:

```
# /usr/cluster/lib/sc/rgmd_debug off
    rgmd verbose debugging is OFF
```

The scdtk_enable and scdtk_collect Man Pages

The scdtk_enable and scdtk_collect are man pages that contain information about enabling and collecting diagnostic information. Appendix A contains both man pages (scdtk_enable(5) and scdtk_collect(5)). The sections on debugging data services and obtaining "live" crash dumps are particularly useful for cluster diagnostics.

Debugging Data Services

Various levels of debugging output are available for data services through syslog. These levels are not enabled by default. To enable them, perform the following steps:

1. Log in as root on the node(s) on which you wish to enable debugging.
2. Open /etc/syslog.conf in an editor and add the following lines:

```
local7.debug        /dev/console
local7.debug        /var/adm/messages
```

The columns must be separated by tab(s).

3. Restart syslogd:

```
# pkill -HUP syslogd
```

4. Test the addition to /etc/syslog.conf:

```
# logger -p local7.debug test
```

You should see output with test on the console and at the end of /var/adm/messages:

```
Jan  4 12:28:26 pnode1 root: [ID 702911 local7.debug] test
```

5. Set the log level for the resource type.

The file that controls the debug level is named "loglevel" and is stored in a directory specific to a resource type. The Sun Cluster software currently uses the Vendor ID and resource type name (rtname) in the path for the debug file:

```
/var/cluster/rgm/rt/<VendorID.rtname>/loglevel
```

To turn on debugging for a resource type, type the following:

```
# mkdir -p /var/cluster/rgm/rt/<VendorID.rtname>
# echo <debuglevel> > /var/cluster/rgm/rt/<VendorID.rtname>/loglevel
```

The <VendorID.rtname> may be any of the supported resource types (SUNW.nfs, SUNW.dns, for example).

The <debuglevel> may be one of:

0	For no debug messages - error and information messages only
1	For the fewest debug messages, as well as all error and information messages
2-8	For an increasing number of debug messages
9	For the most debug messages

For example, the following commands will enable the highest debugging level on the SUNW.nfs resource type:

```
# mkdir -p /var/cluster/rgm/rt/SUNW.nfs

# echo 9 > /var/cluster/rgm/rt/SUNW.nfs/loglevel
```

Note – The debug level is read at the start of each method invocation. Changing the debug level will take effect on the next call of a given method.

Obtaining "Live" Crash Dumps

If one node in the cluster crashes and you would like to get crash dumps from other cluster nodes without taking them down, you can take a "live" crash dump with savecore -L. This, however, requires the creation of a dedicated dump device.

Example: Creating a Dedicated Dump Device

A dedicated dump device is any disk with sufficient capacity to contain the dump files. The cluster product team recommends that the device be as large as physical memory. If that is not possible, be sure that the dump device is at least half the size of physical memory.

Use the dumpadm command to configure the dedicated dump device:

1. Verify that the system does not currently contain a dedicated dump device:

```
# dumpadm
```

The output from this command will be the current status of the dump device:

```
      Dump content: kernel pages
       Dump device: /dev/dsk/c0t0d0s1 (swap)
Savecore directory: /var/crash/clusternode1
 Savecore enabled: yes
```

In this sample for a system named clusternode1, crash dumps are enabled as indicated by the Savecore enabled field. However, the dump device is also the swap partition. This means that this system does not have a dedicated dump device.

2. Determine the disk and directory you wish to use for the dump device. If the directory does not exist, create it before proceeding.
3. Create the dump device. The following example creates a dump device on the /dev/dsk/c7t3d0s1 device and will store the dump files in /space/crash:

```
# dumpadm -c all -d /dev/dsk/c7t3d0s1 -s /space/crash

    Dump content: all pages
    Dump device: /dev/dsk/c7t3d0s1 (dedicated)
    Savecore directory: /space/crash
    Savecore enabled: yes
```

Example: Obtaining a Live Crash Dump

To obtain the live crash dump, type the following:

```
# savecore -L
dumping to /dev/dsk/c7t3d0s2, offset 65536
100% done: 33030 pages dumped, compression ratio 3.99, dump succeeded
System dump time: Thu Jan  4 12:38:54 2001
Constructing namelist /space/crash/unix.0
Constructing corefile /space/crash/vmcore.0
100% done: 33030 of 33030 pages saved
```

You may now use your favorite debugger or the ddb DTK command to examine the `unix.0` and `vmcore.0` files.

To use adb to examine the core files:

```
# adb -k unix.0 vmcore.0
```

The same cluster component debug buffers that you accessed on the running system are also available in the core files and may be useful in determining the cause of a cluster failure after the fact.

The scdtk_info Man Page

The scdtk_info man page contains a list of various data files that are used by Sun Cluster 3.0:

- ccr_infrastructure: CCR cluster infrastructure table
- dcs_service: Device Service CCR table
- dcs_service_classes: Device Service Classes CCR table
- dcs_service_keys: Device Service Keys CCR table
- did_instances: DID Device Information CCR table
- did_types: DID Metadata CCR table
- rgm: Resource Group Manager CCR tables

Further information about each data file is contained in the Appendix.

macros.tar

The DTK contains a collections of macros that may be used with adb or mdb to gather debugging information. For each data structure represented there is a macro of the same name as the structure or class.

Before using the macros the tar files should be extracted:

```
# cd /usr/cluster/dtk/lib/adb
# tar xf macros.tar
# cd sparcv9
# tar xf macros.tar
```

For each macro there is a corresponding file with the suffix .txt that contains a more concise description of what the macro displays.

Example: Using a Macro

To dump out the pm_if data structure:

```
# adb -I /usr/cluster/dtk/lib/adb -k /dev/ksyms /dev/mem

$ <pm_if

30002d90008:    pm_if:refcnt
30002d90008:    _refcnt         vtbl
                74682033        0x3030303139633664
```

Sun Cluster 3.0 Command Line Utilities

Table 3-7 lists some of the Sun Cluster command line utilities that can also be used as diagnostic tools. These utilities were covered in detail in the beginning course and will not be described further in this module. Consult the appropriate man page if you require additional information on any of these commands and their options.

Table 3-7 Sun Cluster Diagnostic Command Line Utilities

Command	Function
`pnmstat [-l]`	Prints the status and configuration of NAFO groups
`sccheck`	Performs limited check and validation of Sun Cluster configuration
`scconf -p [-v[v]]`	Prints a listing of the current Sun Cluster configuration
`scdidadm -c`	Performs consistency check against the kernel representation of the devices and the physical devices
`scdidadm -l | -L [-h]`	List devices in the DID configuration file
`scinstall -p [-v]`	Prints version information
`scrgadm -p [-v[v]]`	Displays existing resource information
`scstat [-p[v[v]]`	Prints the status and configuration of the cluster

An Undocumented Option to scdidadm

scdidadm -k or -K asks the DID driver about the instances it currently knows about and is undocumented.

```
# scdidadm -k
#
# SEAGATE 3CD0TTSQ00007117
#
instance        1
        type DEVID_SCSI_SERIAL
        diskid 534541474154452033434430545453513030303037313137
        proto193:/dev/dsk/c0t0d0
#
#
#
instance        2
        type DEVID_NONE
        diskid 0
        proto193:/dev/dsk/c0t6d0
....

# scdidadm -K
instance        1
        /dev/dsk/c0t0d0 state: 0
instance        2
        /dev/dsk/c0t6d0 state: 0
instance        3
        /dev/dsk/c0t8d0 state: 0
instance        4
        /dev/dsk/c0t9d0 state: 0
instance        5
        /dev/dsk/c0t10d0        state: 0
instance        6
        /dev/dsk/c2t2d0 state: 0
instance        7
        /dev/dsk/c2t3d0 state: 0
...
```

If you are having a problem accessing a device and these commands show output, the problem may be DID-related. If no output is displayed for the device, the problem is most likely related to the device itself.

Explorer

Instructor Note: This section is presented for completeness. If the class is already familiar with explorer, it can be skipped.

Explorer is a support tool used to collect pertinent data from a system running Solaris. Sun engineers commonly use `explorer` to describe a system's configuration or to troubleshoot a problem.

The explorer script works by calling a collection of scripts located in the `/opt/SUNWexplo/tools` directory to gather the system data. It then bundles up the data into a compressed tarfile. If requested, `explorer` can also `uuencode` the file and email it to Sun or to a designated recipient.

Table 3-8 lists the scripts located in the tools directory and provides a brief description of each tool's function.

Table 3-8 Explorer Scripts

Script	Function
`cluster`	The `cluster` script collects extensive information on Sun Cluster
`cst`	The `cst` script collect CST (Configuration and Service Tracker) information
`disks`	The `disks` script collects extensive disk information
`emc`	The `emc` script collects EMC power information
`etc`	The `etc` script collects configuration files in `/etc`
`fru`	The `fru` script collects FRU ID information
`lic`	The `lic` script collects extensive license information
`lp`	The `lp` script collects extensive printer information
`messages`	The `messages` script collects messages files in `/var/adm`
`nbu`	The `nbu` script collects NetBackup information

Table 3-8 Explorer Scripts (Continued)

Script	Function
ndd	The ndd script collections TCP / IP configuration information
netinfo	The netinfo script collects extensive network information
patch	The patch script collects extensive patch information
photon	The photon script collects extensive information for Sun StorEdge A5X00
pkg	The pkg script collects extensive package information
sbu	The sbu script collects extensive Solstice Backup information
scextended	The scextended script collects extensive SSP information
sds	The sds script collects extensive Solstice DiskSuite information
sf15000-ndd	The sf15000-ndd script collects extensive Starcat configuration information
sf15000-sc	The sf15000-ndd script collects extensive Starcat configuration information
sonoma	The sonoma script collects extensive information for Sun StorEdge A3X00
ssa	The ssa script collects extensive information for SPARCstorage Arrays
ssp	The ssp script collects extensive information for Starfire System Service Processors
sysconfig	The sysconfig script collects extensive system configuration information
t3	The t3 script collects information for Sun StorEdge T3
t3extended	The t3extended script collects additional information for Sun StorEdge T3

Table 3-8 Explorer Scripts (Continued)

Script	Function
u4ft	The u4ft script collects extensive information for Netra FT1800
var	The var script collects log and configuration files in /var
vtsst	The vtsst scripts collects StorTools diagnostic information.
vxfs	The vxfs script collects extensive Veritas File System information
vxvm	The vxvm script collects extensive Veritas Volume Manager information

Installing Explorer

Explorer is available for download from http://eservices.central/knowledge/products/explorer. Installation directions are included at the site.

Using explorer Manually

To run explorer manually after installation, log in as root and type the following:

```
# /opt/SUNWexplo/bin/explorer
```

The output is a compressed tar file:

```
/opt/SUNWexplo/output/explorer.<hostid>.<hostname>-<date>-
tar.gz
```

To extract the output:

```
# /opt/SUNWexplo/bin/gzip.sparc -dc <file> | tar xvf -
```

Explorer contains additional options to mail the output to yourself or Sun support as well as to relocate the output. The explorer installation includes man pages that explain these options in detail.

Note – Sun Cluster 3.0 is only supported on Explorer 3.5 or higher.

Module 3 Labs

Place the SUNWscdtk_GA.tar.Z file in a location accessible by the students and note the location on the board.

The purpose of the following labs is to:

- Install the Sun Cluster 3.0 Diagnostics Tool Kit Software on a running cluster.
- Use selected DTK commands to obtain diagnostic information.
- Use the `ccradm` command to correct the loss of operational quorum on a two-node cluster.
- Obtain a “live” crash dump.

Refer to the material in your student guide to help you through the labs.

Lab 1: Installing the Sun Cluster 3.0 Diagnostics Tool Kit Software

1. Obtain the location of the DTK compressed tar file from your instructor.
2. Install the DTK using the procedures in the student guide.
3. Verify the installation:

   # pkginfo -l SUNWscdtk

Lab 2: Using Selected DTK Commands to Obtain Diagnostic Information

1. Validate the contents of the infrastructure file using the appropriate DTK command.

   a. What command did you use? _____

The students should use: /usr/cluster/lib/sc/chkinfr.

   b. Did the command complete successfully (yes or no)?

2. Enable the highest debugging level on the SUNW.nfs resource type on the cluster node that is currently not the primary for the NFS resource. (Hint: Use scstat to determine the primary.)

a. What steps did you take?

The students should follow the section “Debugging Data Services”.

_____

_____

_____

_____

_____

_____

b. Switch the NFS resource group to the node that you set up for debugging.

```
# scswitch -z -g nfs_resource_group -h nodename
```

c. Switch the NFS resource group back to the original node.

```
# scswitch -z -g nfs_resource_group -h nodename
```

d. Is there a difference in the diagnostic output on the node with the highest debugging level (yes or no)?

3. Print out the PDT for the Apache Web service.

a. What command(s) did you use?

_____

_____

```
# /usr/cluster/dtk/bin/print_net_state -s
# /usr/cluster/dtk/bin/print_net_state -p -v -S apache_resource
```

b. How many buckets does node 1 have? _____

c. How many buckets does node 2 have? _____

Lab 3: Using the ccradm Command to Boot After a Loss of Operational Quorum

In this lab, you will simulate a hardware failure and use ccradm to correct it.

1. Make sure you have console connections to both cluster nodes.
2. Choose one of your cluster nodes to be the one with amnesia and shut it down:

   # **init 0**

3. After the first node is at the ok> prompt, simulate a hardware failure by shutting down the second node:

   # **scshutdown -y -g 0**

4. Verify that your cluster has lost operational quorum by trying to boot the node that you first shut down in Step 2. What message do you see before it hangs?

   _____

The message should be: NOTICE: CMM: Cluster doesn't have operational quorum yet; waiting for quorum.

5. Force the first node back to the ok> prompt. Type Ctrl-]. This should put you to the telnet> prompt. Send a break to force the server down:

   telnet> send break

6. Follow the procedures outlined in the example "Recovering from Loss of Operational Quorum in a Two Node Cluster" to bring your the first node online. Bring your second node "back into service" by booting it into cluster mode.
7. Verify that your cluster is functioning properly:

   # **scstat**

Lab 4: Obtaining a "Live" Crash Dump

1. Follow the procedures outlined in the example "Creating a Dedicated Dump Device" to create a dedicated dump device on one of the cluster nodes. Use the boot disk if there is adequate space.
2. Follow the procedures outlined in the example "Obtaining a Live Crash Dump" to obtain the crash dump.

3. Use the adb command to read the crash dump files.

   a. What command did you use?

_____

   b. Examine one or more of the following debug buffers:

```
pm_dbg
cmm_dbg_buf
rm_dbg_buf
ccr_dbg
net_dbg
```

   c. Was the output what you expected? Why or why not?

_____

_____

Module 4

Cluster Monitoring With Sun™ Management Center 3.0

Objectives

This module introduces the Sun™ Management Center 3.0 tool available for Sun Cluster 3.0.

Upon completion of this module you should be able to:

- Install and configure Sun Management Center 3.0 for use with a cluster
- Use Sun Management Center to monitor a cluster

Additional Resources

Additional resources – The following references can provide additional details on the topics discussed in this module:

- *Sun Management Center 3.0 Software User's Guide*, part number 806-5942-10, `http://www.sun.com/sunmanagementcenter/docs/index.html`
- *Sun Cluster 3.0 U1 Installation Guide*, 806-7069-10
- Sun Management Center 3.0 Licensing Requirements, `http://sunwww.central/solaris/sunmanagementcenter/licensing`.

Sun Management Center 3.0

Point out to the students that the section on Sun Management Center provides only a brief introduction to Sun Management Center specifically as it applies to cluster monitoring. If the students wish to learn more about Sun Management Center, refer them to the User's Guide listed as an Additional Resource near the beginning of the module.

Students who completed the SunU Support Readiness Training (SRT) course were already exposed to Sun Management Center. If everyone in the class came from the SunU SRT, give the students the option to skip this module.

If, however, students in the class used the ES-333 class as a prerequisite, they may not have installed or used the software. If that is the case, cover this module in its entirety.

Sun Management Center Overview

Sun Management Center software is an open, extensible system monitoring and management solution that uses Java software protocol and Simple Network Management Protocol (SNMP) to provide enterprise-wide management of Sun products and their subsystems, components, and peripheral devices.

Table 4-1 Sun Management Center Technology

Feature	Description
System Management	Monitors and manages the system at the hardware and operating system levels. Monitored hardware includes boards, tapes, power supplies and disks.
Operating System Management	Monitors and manages operating system parameters including load, resource usage, disk space, and network statistics.
Application and Business System Management	Provides enabling technology to monitor business applications such as trading systems, accounting systems, inventory systems, and real-time control systems.
Scalability	Provides an open, scalable, and flexible solution to configure and manage multiple management administrative domains (consisting of many systems) spanning across an enterprise.

Sun Management Center Architecture

Sun Management Center software comprises three component layers:

- The console is the user interface that interacts with the user to initiate management tasks.
- The server (manager) executes management applications and sends requests to agents in order to perform management tasks on the user's behalf.
- The agents execute on the managed nodes to access the management information, monitor local resources, and respond to manager requests.

New Features of Sun Management Center 3.0

The initial release of Sun Cluster 3.0 supported Sun Management Center 2.1.1 only. This release is still supported, but Sun Management Center 3.0 is now recommended and contains the following new features:

- Grouping of objects provides an easy way to define and invoke complex tasks on a set of managed objects
- Enhanced alarm management and predictive failure analysis increase system availability
- Comprehensive online hardware diagnostics testing identifies faults before the system is affected
- Web-based interface provides anywhere access to management information
- GUI module builder provides a powerful, easy-to-use interface for developing custom modules
- New filtering capabilities help pinpoint problems quickly, even in systems with thousands of objects or nodes
- Secure management controls for dynamic reconfiguration and domain management
- Support for new UltraSPARC™ III based systems

You can find a list of the major changes from Sun Management Center 2.1.1 to Sun Management Center 3.0 in the Sun Management Center 3.0 Release Notes at `http://www.sun.com/sunmanagementcenter/docs/index.html#release`.

Sun Management Center 3.0 Licensing Requirements

Previous licensing models accessed all functionality. With the release of Sun Management Center 3.0 software, licensing is service-based rather than unit-based. Now, there are three levels of licensing: Basic, Advanced and Premier. These levels are defined below:

Basic

Sun Management Center 3.0 Basic Functionality includes basic hardware management/monitoring capabilities, dynamic reconfiguration, and basic OS monitoring. It is available free of charge for an unlimited number of nodes.

Advanced

The Advanced System Monitoring package provides full kernel reader functionality, Solaris Operating Environment health monitoring, file system monitoring, directory size monitoring, process monitoring, and log viewing. It is licensed on a per-node basis.

Premier

The Premier Management Applications package includes a Web interface to Sun Management Center, group operations, module configuration propagation, command line interface, import/export data, and data views. Premier is licensed on a per-Solaris image basis. Cost per node drops with volume.

Additional licensing information is available at `http://sunwww.central/solaris/sunmanagementcenter/licensing`.

Installing Sun Management Center 3.0

Installation of Sun Management Center for use with Sun Cluster 3.0 consists of the following steps:

1. Identify and install the Sun Management Center console, server and help server.

Note – The Sun Management Center console, server and help server should *not* be any of the cluster nodes. They can, however, reside on the cluster's administrative console system. Note the port configured for server-agent communication (the default is 161).

System requirements are detailed at:
`http://www.sun.com/sunmanagementcenter/docs/config-deploy3.0.guide.html`

2. Install the Sun Management Center agent component on each of the cluster nodes.

3. Install the Sun Cluster 3.0 modules on the Sun Management Center server, help server and console. The Sun Cluster 3.0 modules for Sun Management Center are:

Table 4-2 Sun Cluster Management Center Modules

Package Name	Sun Cluster Management Center Component Installed On:
SUNWscsam	Agent (Cluster Nodes)
SUNWscsal	Agent (Cluster Nodes)
SUNWscssv	Server
SUNWscshl	Help Server
SUNWscscn	Console

Note – The Agent components are automatically installed during the initial Sun Cluster 3.0 installation.

4. Configure the Sun Management Center to monitor a cluster.

Installing the Sun Management Center Servers

The following section describes installation of the Sun Management Console, Server and Help Servers.

1. Identify the server(s) you will be using as the Console, Server and Help Server.
2. Log in as root on each server and install the appropriate packages from the Sun Management Center CD-ROM or a networked file system:

```
# < Location of Sun Management Center Software >/sbin/es-inst
```

The `es-inst` script guides you through the installation process. Select the component(s) (server, console or help) that you wish to install on each server. Answer yes (y) to all questions and accept the defaults.

You may see the following warning during installation:

```
-------------- WARNING ------------------
It appears that agent.snmpPort 161 is already is use.
Sun Management Center agent may not be able to run due to this conflict.
There are two ways to correct this conflict:
1. Reconfigure the port that Sun Management Center uses.
2. Stop the process that is using the port.
You are currently running snmpdx, which may be causing the conflict.
```

If you see this message and are not sure whether you need `snmpdx`, the safest approach is to select another port number. In the example below, port 1161 is selected. Make a note of the port number as you will need it for configuration of the cluster nodes.

```
Do you want to use a different port number for agent.snmpPort? [y|n|q] y
Please enter any port greater or equal to 1100 : 1161
```

3. Select the Sun Management Center add-on product(s) (for example, Advanced System Monitoring) you want to install. You will be prompted to insert the second CD (2 of 2).

Note – No license is required for basic functionality. However, licensing is required for Advanced System Monitoring and/or Premier Management. Make sure you have the licensing information available if you plan to install add-on product(s).

4. Installing the Sun Management Center Agent Component

Install the Sun Management Center on each of the cluster nodes.

```
# < Location of Sun Management Center Software >/sbin/es-inst
...
```

```
Install SyMON Server Component? [y|n|q] n
Install SyMON Agent Component? [y|n|q] y
Install SyMON Console Component? [y|n|q] n
Install SyMON Help Documentation? [y|n|q] n
...
```

Installing the Sun Cluster 3.0 Sun Management Center Modules

The following section describes the installation of the Sun Cluster 3.0 Sun Management Center Modules on the Sun Management Center Servers.

1. Install the SUNWscssv package on the Sun Management Center Server:

```
# cd < Location of Sun Management Center Software >/Packages
# pkgadd -d . SUNWscssv
```

2. Install the SUNWscshl package on the Sun Management Center Help Server:

```
# cd < Location of Sun Management Center Software >/Packages
# pkgadd -d . SUNWscshl
```

3. Install the SUNWscscn package on the Sun Management Center Console Server:

```
# cd < Location of Sun Management Center Software >/Packages
# pkgadd -d . SUNWscscn
```

- Detailed installation instructions may also be found at: http://docs.sun.com, Sun Cluster 3.0 7/01 Collection, Sun Cluster 3.0 U1 Installation Guide.

Configuring the Sun Management Center to Monitor a Cluster

1. Before configuring Sun Management Center 3.0, install any Sun Management 3.0 patches and/or Sun Cluster 3.0 patches for the Sun Management Center 3.0 module.

   You can find the list of Sun Management Center 3.0 patches at: http://www.sun.com/solaris/sunmanagementcenter/download/patches.html.

   You can find the list of Sun Cluster 3.0 patches in the *Early Notifier Document #24617* at: http://sunsolve.sun.com.

2. On the Sun Management Center server, add the following entries to `/etc/group`:

esadm::3701:*<symon_admin_user_name>*
esdomadm::3702::*<symon_admin_user_name>*
esops::3703:*<symon_admin_user_name>*

3. Start the server processes using the `es-start -A` command:

```
# /opt/SUNWsymon/sbin/es-start -A
```

4. On each of the cluster nodes, start the Sun Management Console agent processes using the `es-start -a` command:

```
# /opt/SUNWsymon/sbin/es-start -a
```

5. On the Sun Management Center console, start the Sun Management Center console using the `es-start -c` command

```
# /opt/SUNWsymon/sbin/es-start -c
```

This command brings up the login screen. Log in using the appropriate user name, password and Sun Management Center server name.

Figure 4-1 Sun Management Center Login Window

6. From the Sun Management Center console, configure each of the cluster nodes:

   a. Highlight the default domain and select Create an Object from the Edit menu.

   b. In the Create Topology dialog box, choose the Node tab.

   c. From the Monitor Via drop-down selection list at the top of the dialog box, select SyMON Agent - Host.

d. Use the physical node name as the Node Label and Hostname. Leave the IP field blank and make sure to fill in the proper port number (as configured during Sun Management Center installation).

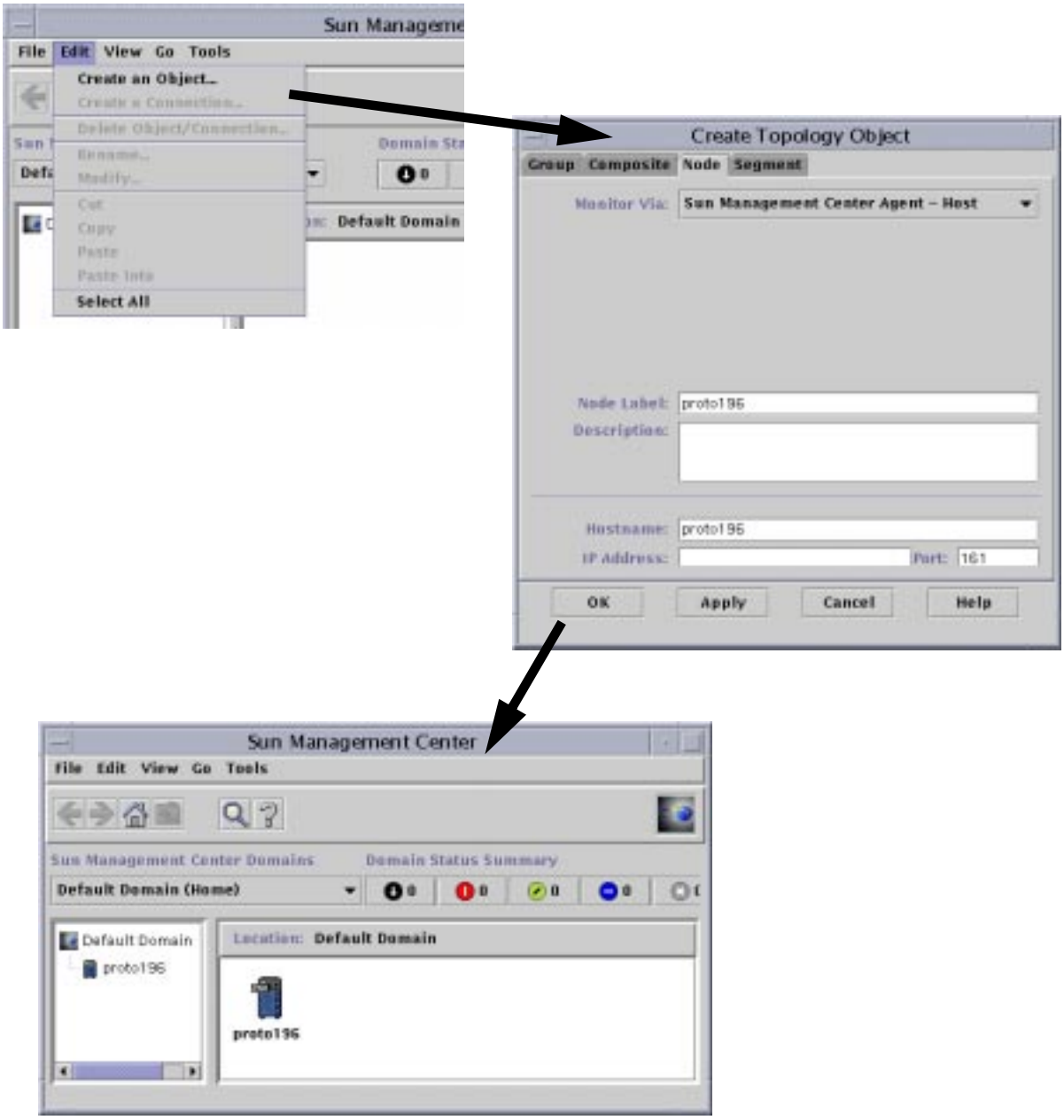

Figure 4-2 Creating Sun Management Center Topology Object

e. Double click the newly created icon for the node in the main console window to bring up the detail window for the node. Choose the Load Module item from the Module menu. From the presented list, choose the Sun Cluster module.

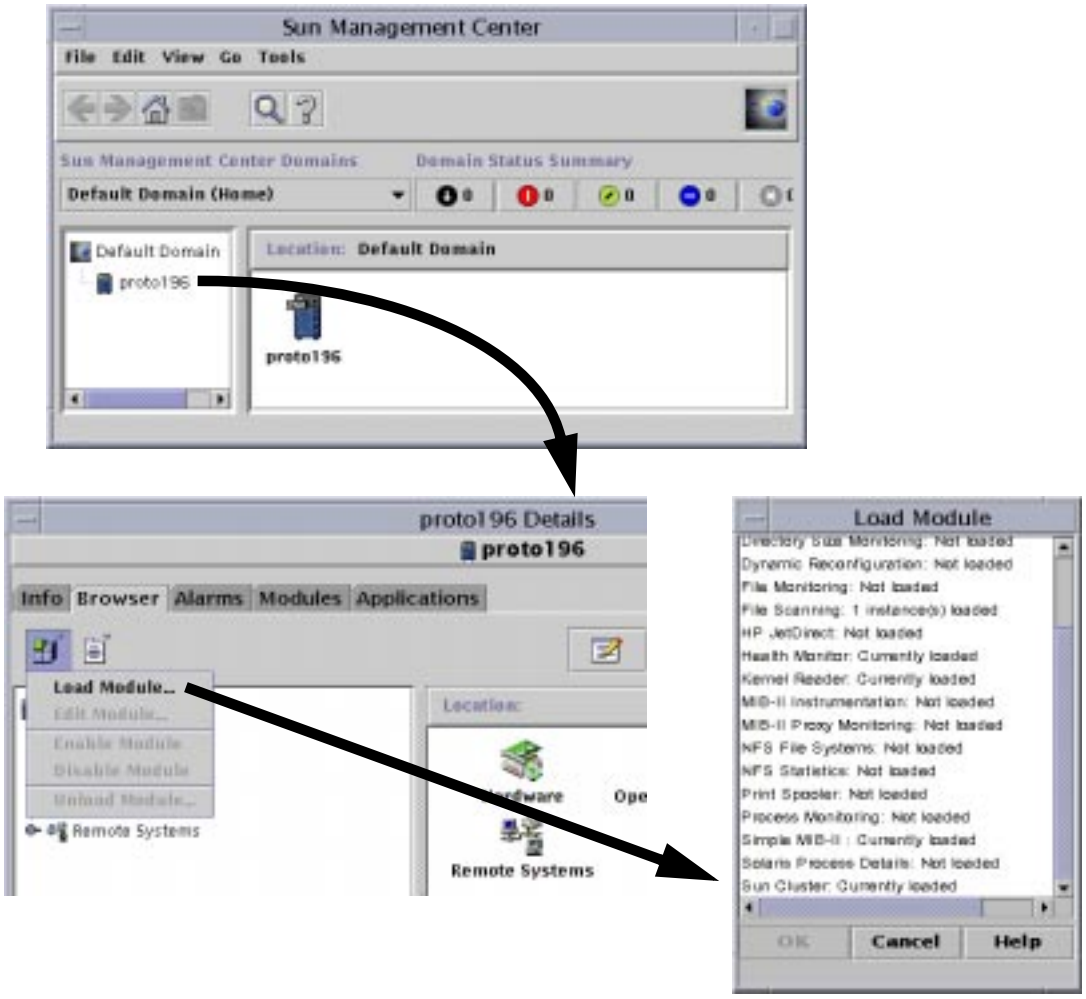

Figure 4-3 Loading a Sun Management Center Sun Cluster 3.0 Module

Monitoring the Cluster

Load SunMC on your screen and display on the overhead. If the students are interested, click through some of the icons to show the information that SunMC can display. In particular, click through one of the cluster nodes and select Operating System/Sun Cluster. Click the individual icons to display the monitored information.

Each time you log in to the Sun Management Center console, the default window appears with all servers in the default domain. In the following figure, `proto196` is the Sun Management Center server, and `proto192` and `proto193` are cluster nodes.

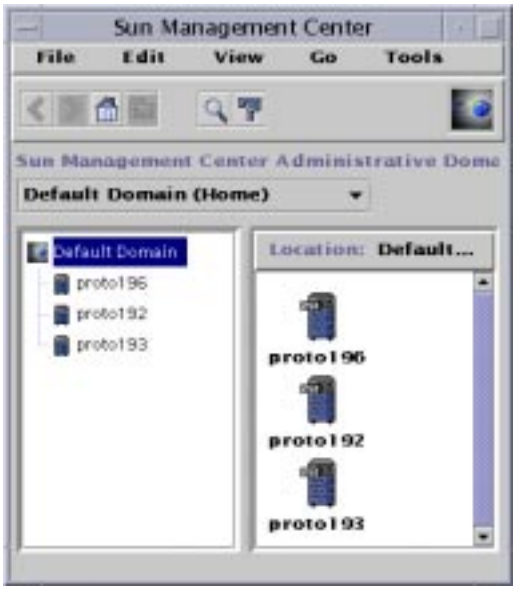

Figure 4-4 Sun Management Center Console

If a node goes down or otherwise exhibits problems, the node's icon should change to reflect the new status using one of the icons in the following figure.

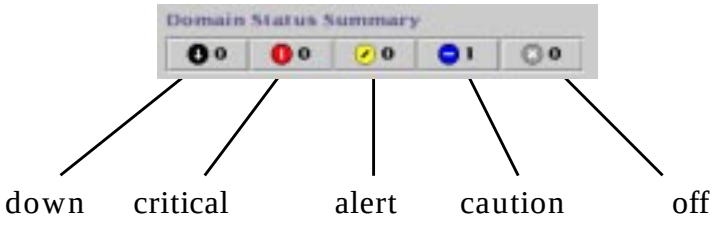

Figure 4-5 Sun Management Center Status Icons

Sun Cluster-specific attributes are under the Operating System attribute of the node. The following cluster information can be monitored:

- Cluster status
- Node status
- Transport, quorum and global devices
- Registered resource types and properties
- Resource group status and properties

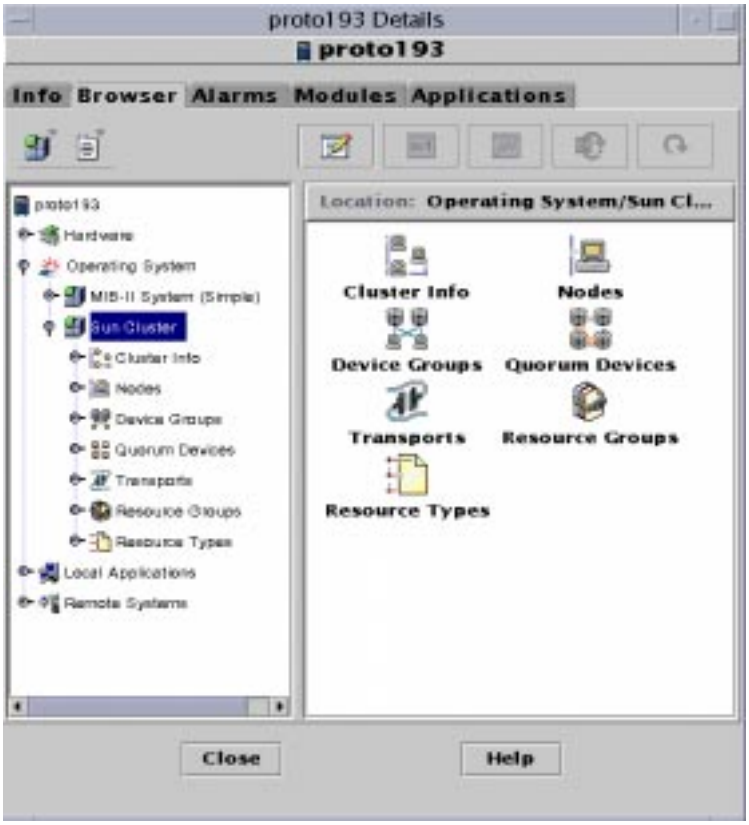

Figure 4-6 Cluster-specific Monitoring Details

By double clicking any of the cluster attributes, the appropriate status and property information will be displayed in a tabular format on the right side of the node's detail window.

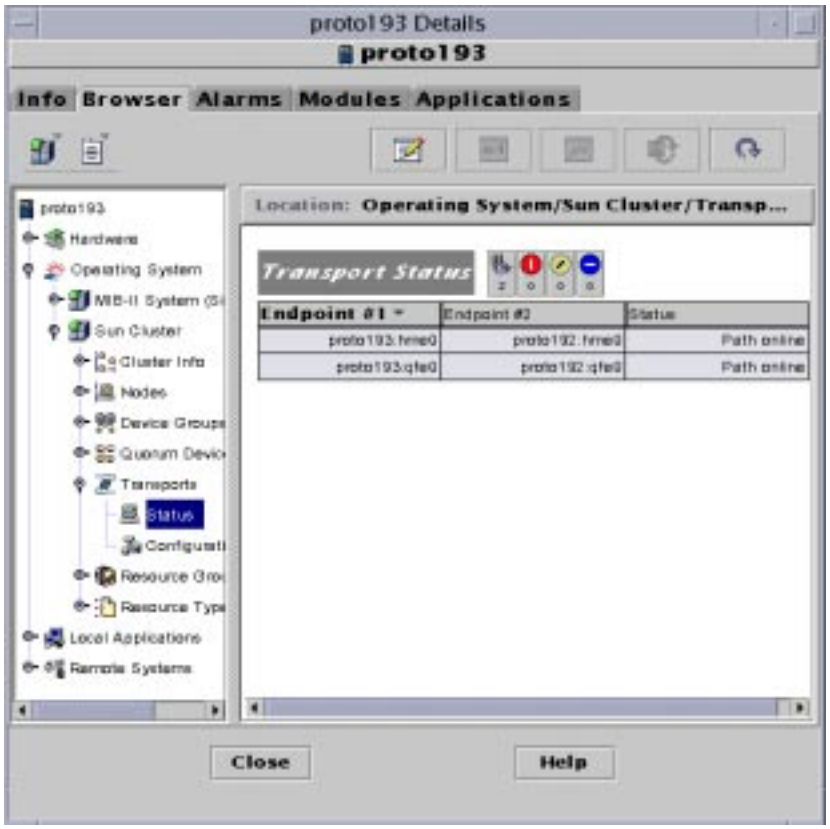

Figure 4-7 Sun Management Center Attribute Details

Sun Management Center Alarms Feature

In addition to simple cluster monitoring, an administrator can configure alarm actions to notify someone of a failure through email or another user-defined action.

Configuring a Simple Alarm

1. In the main console window, select the node that you wish to configure for the alarm. Click the right mouse button to display the pop-up menu, and select Alarm Action.

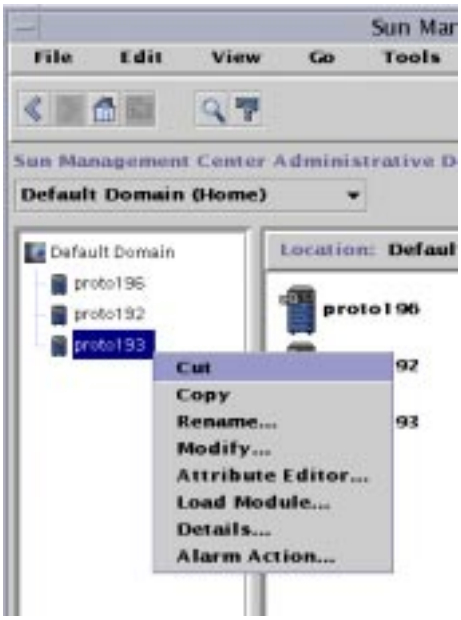

Figure 4-8 Opening an Alarm Action

2. Enter the email action and username(s) to be notified when the host and/or agent is down.

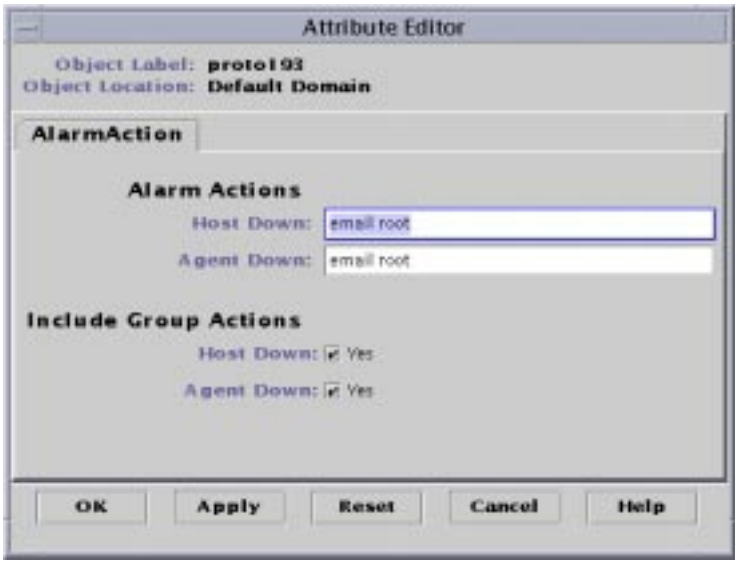

Figure 4-9 Entering an Alarm Action

To notify more than one person, use the format `email username1, username2`.

3. Click OK to close the AlarmAction Attribute Editor and apply the change.

To learn more about using Sun Management Center for monitoring, consult the user documentation at `http://www.sun.com/sunmanagementcenter/docs/index.html`.

Note – Note that the status of the second node is still indeterminate. The status will remain that way until the cluster is shut down and rebooted.

Using Sun Management Center 3.0 to Manage Resource Groups and Resources

Sun Management Center 3.0 may also be used as an administration tool to manage resource groups and resources. To learn more about using Sun Management Center to perform system administration tasks, consult the Sun Management Center online help.software.

Module 4 Lab: Installing and Configuring Sun Management Center 3.0

Make sure that the Sun Management Center 3.0 server is installed with the Sun Cluster 3.0 modules and that the software is accessible to the students.

In this Lab you will be installing and configuring the console and agent components of Sun Management Center 3.0 for Sun Cluster 3.0 monitoring. Refer to the material in the student guide for assistance.

1. Obtain the name of the Sun Management Center server and the location of the Sun Management Center files from your instructor.
2. Install the Sun Management Center Console Server on your cluster administrative workstation.
3. Install the Sun Management Center Agent component on each of your cluster nodes.
4. Install the Sun Cluster 3.0 Sun Management Center module on the Console server (administrative workstation).
5. Configure the Sun Management Center to monitor your cluster.
6. Test the functionality of the Sun Management Center by using the Sun Management Console to determine the answers to the questions below.

   a. What is the status of the transport path(s)? _____

   b. Which device groups are being monitored and which node is the primary for each?

   _____

   _____

   c. Which device is the quorum device and which node is the owner?

   _____

   d. What are the names of the Failover and Scalable Resource Groups in your cluster?

   _____

   _____

Module 5

Responding to Error Messages

Objectives

This module provides an introduction to cluster error logs and the *Sun Cluster 3.0 Error Messages Guide*.

By the end of this module you should be able to:

- List the locations of cluster error logs
- Identify the cause of a cluster failure using the cluster error logs
- Describe the format of the *Sun Cluster 3.0 Error Messages Guide*
- Respond to selected error messages

The goal of this module is to provide students with additional tools they can use for troubleshooting a cluster. It is a short module since it leverages the existing Error Messages Guide.

Additional Resources

The following references can provide additional details on the topics discussed in this module:

- *Sun Cluster 3.0 U1 Error Messages Guide*, 806-7076-10
- *Sun Cluster 3.0 U1 System Administration Guide*, 806-7073-10
- *Sun Cluster 3.0 U1 Installation Guide*, 806-7069-10

Cluster Error Logs

Most cluster error messages are logged to the `syslog` file(s) as defined in `/etc/syslog.conf`. By default, this means that most error messages are logged to `/var/adm/messages`.

Syslog

The guidelines used by the cluster development team to define severity are as follows:

1. Panic messages are used when killing a node. These are used when a system determines that it is better for this node to die immediately than for it to continue running. Examples are:
   - The node encountered an unrecoverable error and cannot continue.
   - The node detected a situation where it cannot guarantee data integrity.
   - The node detected a situation where it must die in order to allow the rest of the cluster to continue to operate (to break a deadlock, for example).
2. Warning messages are used to notify the administrator of problems or potential problems with the system. In some cases the administrator may need to take action to deal with the situation. In other cases, the system may recover from the problem without operator intervention.
3. Notice messages are used to provide informational messages to the administrator or troubleshooter / debugger. These may be used to indicate that something of interest happened, which is not necessarily indicative of a problem.
4. Event messages are used to notify the Sun Management Center GUI of a state change. The GUI team works with each development group to identify which components need to generate events. Event messages also appear on the console and in `/var/adm/messages` and they often satisfy the needs of Warning and Notice messages.

Specific error messages and the Error Messages Guide are described later in the module.

Data Service Logs

Each data service may write error messages to an application-specific log. Sun-supplied data services log to the locations listed inTable 5-1 below.

Table 5-1 Data Service Logs

Data Service	Error Log	Default Location	Function

HA for Apache	`mod_jserv.log`
	`jserv.log`
	`error_log`
		`/var/apache/logs`
		`/var/apache/logs`
		`/var/apache/logs`
			Logs C portion of Apache JServ
			Logs Java portion of Apache JServ
			Logs server errors

HA for DNS	logs to `syslog`	`/var/adm/messages`	Logs errors
HA for NFS	logs to `syslog`	`/var/adm/messages`	Logs errors

HA for Oracle	`message_log.*`
	`message_log.*`
	`$ALERT_LOG_FILE`
		/var/opt/SUNWscor/oracle_server
		/var/opt/SUNWscor/oracle_listener
		none, set as extension property
			Logs Server messages
			Logs Listener messages
			Logs alerts

HA for iPlanet	`error`	defined in `magnus.conf`	Logs errors
HA for Netscape Directory Server	logs to `syslog`	`/var/adm/messages`	Logs errors

HA for Sybase ASE	message_log
	syslog
	restart_history
		`/opt/SUNWscsyb/log`
		`/var/adm/messages`
		`/opt/SUNWscsyb/log`
			Logs Sybase messages
			Logs errors
			Logs fault monitor history

HA for SAP	startsap*.log
	stopsap*.log
	syslog
	trace files
		`<SAPSID>adm home`
		`<SAPSID>adm home`
		`/var/adm/messages`
		`work directory of each SAP instance`
			Logs SAP startup
			Logs SAP shutdown
			Logs cluster errors
			Logs internal SAP process/errors

Installation Logs

Cluster installation logs are stored in `/var/cluster/logs/install`. Install logs are named `scinstall.log.${pid}` and upgrade logs are named `scinstall.upgrade.log.${pid}`.

Identifying Failures Using the Cluster Error Logs

The cause of an application failure can be particularly difficult to determine, as the cause may be application-specific or related to failures within the cluster.

Troubleshooting Application Failures

One possible approach to troubleshoot an application crash:

1. Check the health of the cluster on the node where the application crashed:

```
# scstat -g -h <node>
```

The status returned for the application resource should be one of the values specified in Table 5-2.

Table 5-2 Application Resource Status

Status	Description
Online	The resource is online and providing service.
Degraded	The resource is online, but its performance or availability might be compromised in some way.
Faulted	The resource has encountered an error that prevents it from functioning.
Unknown	The current status is unknown or is in transition.
Offline	The resource is offline.

2. If the output from the `scstat` command indicated that all of the cluster components are working properly (such as interconnect, disk device groups), check the application-specific log if it exists.
3. If the output from the `scstat` command indicates a cluster-related failure, then check the `syslog` on the affected node for errors. It may also be necessary to check the `syslog(s)` on the other node(s) of the cluster. Look for the first failure message in each `syslog`.
4. If the cause of the failure is obvious from the `syslog`, take the appropriate steps to correct the error.
5. If the cause of the failure is not obvious, then check the Cluster Error Messages Guide for symptoms and resolutions.
6. If the cause is still not obvious, consult SunSolve or local resources for clues.

7. If the cause can still not be determined, follow the escalation procedures defined for your group.

Troubleshooting Installation Failures

Most installation failures can be traced to incorrect hardware configurations or incorrect data entered during the installation.

If you are certain that the configuration is correct, check the installation log for clues.

Troubleshooting Cluster Failures

Hardware and cluster software failures may be identified from the error logs as follows:

1. Check the health of the cluster on all nodes:

```
# scstat -p[v][vv]
```

Use `-pv` to obtain more verbose output and `-pvv` to obtain the most verbose output.

The status returned for the application resources will be one of the values specified in Table 5-2 on page 5-5.

The status returned for device groups will be one of the values specified below in Table 5-3.

Table 5-3 Device Group Status

Status	Description
Online	The device group is online. There is a primary node and devices within the group are ready for I/O.
Degraded	The device group is online, but not all of its potential primaries (secondaries) are up. For two-node connectivity, this status basically indicates that a stand-by primary does not exist, which means a failure of the primary node will result in a loss of access to the devices in the group.

Table 5-3 Device Group Status (Continued)

Status	Description
Wait	The device group is between statuses. This status might occur, for example, when a device group is going from offline to online.
Offline	The device group is offline. There is no primary node. The device group must be brought online before any of its devices may be used.

2. If the output from the `scstat` command indicates a cluster-related failure, then check the `syslog` on the affected node(s) for errors. It may also be necessary to check the `syslog`(s) on all other node(s) in the cluster. Look for the first failure message in each `syslog`.
3. If the cause of the failure is obvious from the `syslog`, take the appropriate steps to correct the error.
4. If the cause of the failure is not obvious, then check the Cluster Error Messages Guide for symptoms and resolutions.
5. If the cause is still not obvious, consult SunSolve or local resources for clues.
6. If the cause can still not be determined, follow the escalation procedures defined for your group.

Sun Cluster Error Messages

The Sun Cluster Error message guide contains a list of error messages that could appear on the system console or in `/var/adm/messages` while running Sun Cluster software. Each message includes the information shown in the following table.

Table 5-4 Error Message Components

Message Component	**Description**
Message ID	The message ID is an internally-generated ID that uniquely identifies the message.
Description	The description is an expanded explanation of the error that was encountered, including any background information that might aid you in determining what caused the error.
Solution	The solution is the suggested action or steps that you should take to recover from any problems caused by the error.

The message ID is a range of numbers between 100000 and 999999. The chapters are divided into ranges of message IDs. Within each chapter, the messages are ordered by message ID.

Throughout the messages, you will see print characters such as %s or %d. These characters will be replaced with a string or a decimal number in the displayed error message.

Note – There may be cases where an error message ID is not found in the Error Messages Guide.

Example: Locating an Error Message in the Error Messages Guide

To locate an error message in the Error Messages Guide:

1. Obtain the message ID from the displayed error message:

```
131492:pxvfs::mount(): global mounts are not enabled (need to run
"clconfig -g" first)
```

In this example, the message ID is 131492.

2. Locate the message in the Error Messages Guide chapter corresponding to the appropriate range. In this example, message ID 131492 is in Chapter 2, Message IDs 100000-199999:

```
131492:pxvfs::mount(): global mounts are not enabled (need to run
"clconfig -g" first)
```

Description – A global mount command is attempted before the node has initialized the global file system name space. Typically this caused by trying to perform a global mount while the system is booted in single user mode.

Solution – If the system is not at run level 2 or 3, change to run level 2 or 3 using the init(1M) command. Otherwise, check message logs for errors during boot.

Note – In some cases, the error message may be informational only and does not indicate an actual error.

Module 5 Labs

The purpose of the following labs is to:

- Identify faults based on initial problem descriptions
- Use the cluster error logs, application error logs, or the Sun Cluster 3.0 Error Messages Guide to determine the cause of the faults and resolve the problems
- Document the error codes and solutions

In each of the following labs, you will be presented with a problem. Use the cluster error logs, application error logs, or the Error Messages Guide to determine the cause of the problem. Correct the error and document your solution.

Ask your instructor to prepare your system before beginning each lab.

Instructor notes:

Use the following guide to introduce faults into the student clusters, one at a time. The solutions are contained below. Alternatively, you may use the lab script “IES-SC33_errors_lab_setup” to generate each fault. Many of these faults involve a reboot of the system. For the lab to be effective, the students should wait until the reboot has finished (simulating a normal environment) before logging into the node to track down the problem.

Lab 1: Node fails to start all data services after a reboot.

Preparation: Modify `/etc/cluster/pnmconfig` on one of the cluster nodes for each cluster to add a space between the device name and number: for example, `qfe 1` instead of `qfe1`. Reboot the node.

Solution: Run `pnmset` or restore `/etc/cluster/pnmconfig` file and reboot.

Associated error codes: 316344, 394584

Lab 2: Cluster node successfully joined cluster but cannot master Apache data service.

Preparation: Modify `/etc/hosts` file with incorrect hostname for Apache web server. Halt the server with `init 0`.

Solution: Restore `/etc/hosts` file to its original state.

Associated errors: `964072, 457121`

Lab 3: Commands `scstat` and `scrgadm` do not work on cluster node. Node panics when trying to move data service to it.

Preparation: Move the entire `/etc/cluster/ccr` directory to another location and create empty ccr directory in its place: `#mv /etc/cluster/ccr` /etc/cluster/ccr.save; mkdir /etc/cluster/ccr.

Solution: Boot node in non-cluster mode. Restore ccr to its original state by copying `ccr` from other node.

Associated errors: 674848

Lab 4: Cluster node panicked. Diagnose the cause.

Preparation: Kill `rgmd` process. Node will reboot automatically.

Solution: Search `/var/adm/messages` to find root cause.

Associated errors: 770355

Lab 5: Troubleshooting an Application Failure

Preparation: On one node of the cluster, copy the original /etc/apache/httpd.conf to httpd.conf.save. Change the path in DocumentRoot to /usr/global/mirror-1/apache-data/htdocs. Stop the Apache service with "/usr/apache/bin/apachectl stop". The service will stop and restart automatically.

Solution: The students will be forced to use the application error log to solve this problem. The application error log in `/var/apache/logs/error_log` will indicate that the path does not exist. The students should be able to determine this much at least. However, they may not remember that the configuration file is `/etc/apache/httpd.conf`. Be willing to provide hints if they do get this far, but encourage them to locate the error log before providing them.

How can an error like this happen in actual practice? Easily, if one is editing the `httpd.conf` files manually and not using the cluster console.

Lab 6: Node was taken down for maintenance, now it will not boot as a cluster member.

Preparation: Halt one of the cluster nodes. Swap the cluster interconnect cables from one interface to the other.

Solution: Repair interconnect wiring. Cluster will then continue to boot.

Associated errors: 618107, 604153

Lab 7: Cluster boots without errors but pnmswitch fails.

Preparation: Determine the active public network adapter by running `pnmstat -l`. Halt one of the cluster nodes. Remove the cable from any inactive public network adapter.

Solution: Fix network connections.

Associated errors: 143622, 164168

This one seems to be a problem with the `pnmd` daemon not reporting the failure of the backup adapter during a reboot. If the students are interested, have them repeat the lab but remove the cable from the primary adapter instead. The node will not reboot and will not switch over to the backup adapter.

Lab 8: Clients are unable to mount /global/mirror-1

Preparation: Change the /global/mirror-1/SUNW.nfs/dfstab.nfs-rs file to share something other than /global/mirror-1. Restart the resource group on the node that is primary for the nfs-rg group with: scswitch -R -h nodename -g nfs-rg. The nfs service will go offline.

Solution: Students must repair the dfstab.nfs-rs file and enable the resource group with: scswitch -Z -g nfs-rg.

Associated errors: 661614, 922363

Lab 9: Clients are unable to mount /global/mirror-1

Preparation: Add one or two blank spaces before the hosts entry in /etc/nsswitch.conf on the cluster node that is primary for the nfs-rg resource group. Restart the nfs-rg group with: scswitch -R -h nodename -g nfs-rg. The nfs service will become unavailable, but scstat -g will say it is online.

Lab 1: Node Fails to Start Data Services After Reboot

In this exercise, the node fails to start all data services after a reboot.

1. Log into the console on the cluster nodes and determine which node is having the problem.

   ```
   # scstat -g
   ```

2. Identify the cause of the problem and correct it.
3. Exit the console and ask your instructor to insert the next fault.

Error Messages/ Symptoms/ Conditions:

_____

_____

_____

Solution:

_____

_____

_____

_____

Lab 2: Node Does Not Master Data Service

In this exercise, the cluster node successfully joined the cluster but it does not master data device.

1. Log into the console on the cluster nodes and determine which node is having the problem.

   # scstat -g

2. Identify the cause of the problem and correct it.
3. Exit the console and ask your instructor to insert the next fault.

Error Messages/ Symptoms/ Conditions:

_____

_____

_____

Solution:

_____

_____

_____

_____

Lab 3: Node Panics Trying to Master Data Service

In this exercise, while using the `scat` and `scrgadm` commands, they do not work on the cluster node and the node will not master a data service.

1. Log into the cluster nodes through the console.
2. Type the following command on the problem node to verify the problem:

scstat

3. Type the following command on the working node to attempt to move a data service to the node:

```
# scswitch -z -g nfs-resource-group-name -h nodename
```

Substitute the name of the nfs resource group on your cluster for *nfs-resource-group-name* and the name of the working node for *nodename*.

The node should panic.

4. Identify the cause of the problem and correct it.
5. Exit the console and ask your instructor to insert the next fault.

Error Messages/ Symptoms/ Conditions:

_____

_____

_____

Solution:

_____

_____

_____

_____

Lab 4: Cluster node panicked

1. Log into the cluster nodes through the console.
2. Identify the node that panicked and determine the root cause.
3. Exit the console and ask your instructor to insert the next fault.

Error Messages/ Symptoms/ Conditions:

_____

_____

_____

Solution:

_____

_____

_____

_____

Lab 5: Troubleshooting an Application Failure

A customer calls and says that your Apache server is only working part of the time. It appears to generate “URL not found” errors at random intervals. You use your Web browser on the administrative workstation to connect to your Apache server on the cluster to reproduce the problem. You do get an occasional error, but it appears to go away after several reloads.

Your customer insists that even one failure is unacceptable. You must determine the cause of the problem and correct it as soon as possible.

1. Follow the procedures outlined in “Troubleshooting Application Failures” to determine the cause of the problem.
2. Take the necessary steps to correct the problem.
3. Was your solution successful? yes / no

   Extra Credit: How can you be sure?

   _____

The `/var/apache/logs/access_log` will show the successful access.

Error Messages/ Symptoms/ Conditions:

_____

_____

_____

Solution:

_____

_____

_____

_____

Lab 6: Node Will Not Reboot After Maintenance

A node was taken down for maintenance. It will not reboot as a cluster member.

1. Log into the cluster nodes through the console.
2. Identify the node that will not boot and determine the cause.
3. Correct the problem.
4. Exit the console and ask your instructor to insert the next fault.

Error Messages/ Symptoms/ Conditions:

_____

_____

_____

Solution:

_____

_____

_____

_____

Lab 7: pnmswitch Fails after Node Maintenance

In this exercise, the node was taken down for maintenance. It boots without errors but pnmswitch will not work.

1. Log into the cluster nodes through the console.
2. Identify the node that is down and boot it.

```
# boot
```

3. Verify that the cluster is functioning properly:

```
# scstat
```

4. Verify that the NAFO group is functioning properly.

```
# pnmstat -l
```

```
group   adapters        status  fo_time act_adp
nafo0   qfe1:qfe2       OK      NEVER   qfe1
```

5. Use pnmset to change the active adapter.

```
# pnmset -c nafogroup -o switch adapter
```

In this example, *nafogroup* is nafo0 and *adapter* is qfe2:

```
# pnmset -c nafo0 -o switch qfe2
```

6. Determine the cause of the problem and correct it.
7. Exit the console and ask your instructor to insert the next fault.

Error Messages/ Symptoms/ Conditions:

_____

_____

_____

Solution:

_____

_____

_____

_____

Lab 8: Clients are unable to mount /global/mirror-1

Clients have been unable to mount the /global/mirror-1 directory. Determine the source of the problem.

1. Log into the cluster nodes through the console.
2. Verify that the nfs resource has failed:

   ```
   # scstat -g
   ```

3. Determine the cause of the problem and correct it.
4. Exit the console and ask your instructor to insert the next fault.

Error Messages/ Symptoms/ Conditions:

_____

_____

_____

Solution:

_____

_____

_____

_____

Lab 9: Clients are unable to mount /global/mirror-1, nfs resource is online

Clients have been unable to mount the /global/mirror-1 directory. Determine the source of the problem.

1. Log into the cluster nodes through the console.
2. Check the status of the nfs resources:

   # scstat -g

3. Attempt to mount the /global/mirror-1 directory on your console server:

   # mount *nfs-server*:/global/mirror-1 /mnt

4. Determine the cause of the problem and correct it.

Error Messages/ Symptoms/ Conditions:

_____

_____

_____

Solution:

_____

_____

_____

_____

(Optional) Lab 10: Create your own failures

Use any time you have remaining to simulate your own failures. Observe the error messages and record the solutions. Try your “fault” out on another group and ask that group to troubleshoot your “fault”.

Module 6

Addressing Known Issues

Objectives

This module provides an overview of known Sun Cluster 3.0 issues, with an emphasis on the issues listed in the *Sun Cluster 3.0 U1 Release Notes*.

By the end of this module you should be able to:

- Identify Sun Cluster 3.0 resources for known issues and problem resolutions
- Apply workarounds to Sun Cluster bugs
- Identify and resolve SCSI-3 reservations
- Clear the `STOP_FAILED` error flag on resources

Additional Resources

The following references can provide additional details on the topics discussed in this module:

- *Sun Cluster 3.0 U1 Release Notes*, 806-7078-10, `http://docs.sun.com`
- *Sun Cluster 3.0 U1 System Administration Guide*, 806-7073-10, `http://docs.sun.com`
- *Sun Cluster 3.0 U1 Installation Guide*, 806-7069-10, `http://docs.sun.com`
- Early Notifier Document #24617, `http://sunsolve.central.sun.com/cgi/retrieve.pl?doc=enotify/24617`
- Bug Reports, `http://sunsolve.central`
- *Sun Cluster 3.0 Release Notes Supplement*, `http://suncluster.eng`

Sun Cluster 3.0 U1 Release Notes

This module provides an overview of the Sun Cluster 3.0 Release Notes. Each student should have a copy of the release notes. Skim through the notes with the students and emphasize the Restrictions and Known Issues sections. Also be sure to go over the Patches and Required Firmware Levels section. Advise internal audiences to access sunsolve through sunsolve.central and search for the Early Notifier Document #24617.

Encourage any students with 3.0 experience to relate any experiences they have that are mentioned as problems in the release notes.

The lab at the end of the module will illustrate a “Known Problem” that has a workaround suggested in the Release Notes.

Each release of the Sun Cluster 3.0 software comes bundled with a set of release notes that cover additional information not found in the product documentation. *The Sun Cluster 3.0 U1 Release Notes* discusses the following topics:

- New Features and Functionality
- Supported Products
- Features Nearing End of Life
- How to Install Sun Cluster AnswerBooks
- Restrictions
  - General
  - Cluster File System Restrictions
  - Network Adapter Failover (NAFO)
  - Service and Application Restrictions
  - Sun Cluster 3.0 HA for NFS Restrictions
  - Volume Manager Restrictions
  - Hardware Restrictions
- Patches and Required Firmware Levels
- Sun Management Center Software Upgrade
- System Administration and Procedure Updates
- Known Problems
- Known SunPlex Manager Problems
- Appendices with Sample Installation and Data Configuration Worksheets

Many of the problems reported by customers stem from the fact that customers (and even support personnel) often forget to consult Release Notes either when configuring a cluster or diagnosing a problem. Be sure to familiarize yourself with them, especially the Restrictions and Known Problems section.

An online copy of the Release Notes can be found at: `http://suncluster.eng/products/SC3.0` under Sun Cluster 3.0 User Documentation.

Note – New features, hardware qualifications, and bug reports are added to the Release Notes as a supplement, *Sun Cluster 3.0 U1 Release Notes Supplement*. This document is also available at `http://suncluster.eng/products/SC3.0` under Sun Cluster 3.0 User Documentation. It will be regularly updated and eventually integrated into the standard Sun Cluster documentation for the next release.

SCSI Reservations

Sun Cluster 3.0 uses two different kinds of SCSI reservations to ensure data integrity in the event of a node failure.

- SCSI-2 Exclusive Reservations

  These work only with dual-ported drives and operate essentially the same way as they did in Sun Cluster 2.2.

- SCSI-3 Persistent Group Reservations (PGR)

  These incorporate more robust failure fencing features and are only used explicitly by Sun Cluster 3.0 when a device is connected to more than two nodes. Registration keys are stored on the disk and remain even in the case of a hardware reset or removal of the disk from the system.

Note – SCSI-2 and SCSI-3 reservations cannot exist on the same disk.

If certain procedures are not followed correctly, extraneous DID paths to the same device can be created. When this occurs the Sun Cluster framework is fooled into thinking it has a SCSI-3 device and will place a simulated SCSI-3 reservation on the disk by writing a key into the header area on the disk.

The cluster node will in most cases start correctly, file systems will mount, and data services will start. However, starting the device group and mounting the global file systems will fail if the device does not support SCSI-3 PGR.

Actions known to create these additional paths are:

- Re-cabling of storage devices
- Panic during installation – make sure all required Sun Cluster and Volume Manager patches are installed
- Upgrading from SC 2.2
- Moving hardware from another system – the keys persist even if the hardware is moved

To determine whether your disks have an erroneous SCSI-3 reservation key, look for an entry in the system's /var/adm/messages file similar to:

```
Oct 30 15:06:35 mars Cluster.CCR: [ID 186524
daemon.warning] reservation error(release_shared_scsi2) -
do_scsi2_release  error for disk /dev/did/rdsk/d5s2
```

To verify the SCSI-3 reservation use the scsi command to query the disk.

```
# /usr/cluster/lib/sc/scsi -c inresv -d /dev/did/rdsk/d17s2
```

or

```
# /usr/cluster/lib/sc/scsi -c inkeys -d /dev/did/rdsk/d17s2
```

Example: Removing a SCSI-3 Reservation

1. Verify that the SCSI-3 reservation exists on the disk.

   # **/usr/cluster/lib/sc/scsi -c inresv -d \ /dev/did/rdsk/d17s2**

   You should see this:

   /dev/did/rdsk/d17s2

   Reservations(1):

2. Use devfsadm to make sure that no erroneous devices entries exist. for the disk.

   # **devfsadm -C**

3. Use scdidadm to clean up any extraneous DID paths that may exist.

   # **scdidadm -C**

4. Use the scdidadm repair procedure to remove the SCSI-3 key.

   # **scdidadm -R 17**

   Alternatively, you may use the scsi command to remove the key.

   # **/usr/cluster/lib/sc/scsi -c scrub /dev/did/rdsk/d17s2**

Clearing the STOP_FAILED Error Flag on Resources

When the Failover_mode resource property is NONE or SOFT and the STOP of a resource fails, the individual resource goes into the STOP_FAILED state and the resource group into the ERROR_STOP_FAILED state. You cannot bring a resource group in this state on any node online, nor can you edit it (create or delete resources, or change resource group or resource properties).

Use the following procedure to clear the STOP_FAILED error flag:

1. Become superuser on a node in the cluster.
2. Identify which resources have gone into the STOP_FAILED state and on which nodes.

```
# scstat -g
```

3. Manually stop the resources and their monitors on the nodes on which they are in STOP_FAILED state.

   This step might require killing processes or running resource type-specific commands or other commands.

4. Manually set the state of these resources to OFFLINE on all the nodes on which they were manually stopped.

```
# scswitch -c -h nodelist -j resource-name -f STOP_FAILED
```

-c	Clears the flag.
-h *nodelist*	Specifies the node names on which the resource was running.
-j *resource-name*	Specifies the name of the resource to take offline.
-f STOP_FAILED	Specifies the flag name.

5. Check the resource group state on the nodes where the STOP_FAILED flag was cleared in Step 4. It should now be OFFLINE or ONLINE.

```
# scstat -g
```

   If the resource group remains in the ERROR_STOP_FAILED state, as shown by scstat -g, take the resource group offline on those nodes where it is still in the ERROR_STOP_FAILED state by using the following scswitch command:

```
# scswitch -F -g resource-group-name
```

-F	Takes the resource group offline on all nodes that can master the group.
-g *resource-group-name*	Specifies the name of the resource group to take offline.

This situation can occur if the resource group was being switched offline when the STOP method failure occurred and the resource that failed to stop had a dependency on other resources in the resource group. Otherwise, the resource group reverts to the ONLINE or OFFLINE state automatically after you have run the command in Step 4 on all STOP_FAILED resources.

6. If needed, switch the resource group online.

```
# scswitch -Z -g resource-group-name
```

Module 6 Labs

The purpose of the following lab is to:

- Simulate a known bug and apply the suggested work-around
- Simulate a STOP_FAILED error state and apply the suggested procedure

Use the material from the *Sun Cluster 3.0 Release Notes* to help you with the lab.

Make sure each student group has a copy of the *Sun Cluster 3.0 Release Notes*. After the students complete the lab, ask them to discuss their thoughts as to why scshutdown fails when `init 0` does not. The problem is caused by the inability of the cluster file system to do forced unmounts.

Lab 1: Simulating a scshutdown Failure

1. Log on to the console of each cluster node. On one of the cluster nodes, cd to a global directory and issue a command that does not complete, such as:

```
# tail -f /var/adm/messages
```

2. On the other cluster node, issue the following command to shut down the cluster:

```
# scshutdown -y -g0
```

3. Does the command succeed? yes/no
4. Why or why not?
5. Follow the suggested Workaround in the *Sun Cluster 3.0 Release Notes* to correct the problem.
6. Repeat the command to shut down the cluster:

```
# scshutdown -y -g0
```

7. Does the command succeed? yes/no
8. Why or why not?

Lab 2: Simulating a STOP_FAILED Error Flag

In this lab, you will create a new resource type which is guaranteed to generate a STOP_FAILED error.

1. Become superuser on both of the cluster nodes.
2. Copy the `TEST.stopfail` file from the classroom file directory into the `/usr/cluster/lib/rgm/rtreg` directory on both cluster nodes.
3. On one of the cluster nodes, register the new resource type:

   # **`scrgadm -a -t TEST.stopfail`**

4. Create a new resource group:

   # **`scrgadm -a -g rg`**

5. Create a new resource of the new `stopfail` resource type in the `rg` resource group:

   # **`scrgadm -a -j r-will-stop-fail -g rg -t stopfail`**

6. Attempt to bring the resource group online:

   # **`scswitch -Z -g rg`**

   a. What error message do you see on the screen?

   _____

   b. Examine the `TEST.stopfail` file. Does the error message make sense? yes / no

   c. Why or why not?

   _____

   _____

   d. Check the status of the resource group:

   # **`scstat -g`**

   e. What is the status of the `r-will-stop-fail` resource?

   _____

7. Attempt to bring the resource group offline:

   # **`scswitch -F -g rg`**

a. What error message do you see on the screen?

_____

b. Examine the `TEST.stopfail` file. Does the error message make sense? yes / no

c. Why or why not?

_____

d. Check the status of the resource group:

scstat -g

e. What is the status of the `r-will-stop-fail` resource?

_____

8. Use the procedure in the course material to clear the STOP_FAILED error flag:

scswitch -c -h *nodename* **-j r-will-stop-fail -f STOP_FAILED**

a. Check the status of the resource group:

scstat -g

b. What is the status of the `r-will-stop-fail` resource?

_____

c. Is the status what you expected? yes / no

d. Why or why not?

_____

9. Remove the resource, resource type, and resource group:

a. Disable the resource:

scswitch -n -j r-will-stop-fail

b. Remove the resource:

scrgadm -r -j r-will-stop-fail

c. Remove the resource type:

scrgadm -r -t stopfail

d. Remove the resource group:

```
# scrgadm -r -g rg
```

e. Verify that the resource, resource type, and resource group have been removed.

```
# scrgadm -p
```

Module 7

Tuning Data Service Fault Monitors

Objectives

This module introduces the data service fault monitors and associated tunables.

Upon completion of this module you should be able to:

- Define the function of data service fault monitors
- Describe the data service fault monitoring process
- Identify data service tunable properties
- Tune data service fault monitors

Additional Resources

Additional resources – The following references can provide additional details on the topics discussed in this module:

- *Sun Cluster 3.0 Data Services Installation and Configuration Guide,* part number 806-1421

Data Service Fault Monitors

The data services supplied by Sun contain fault monitors that are built into the package. The fault monitor (fault probe) probes the health of the data service.

The fault monitor is invoked by the Resource Group Manager (RGM) when you bring a resource group and its resources online. This invocation causes the RGM to call the appropriate `MONITOR_START` method (function) for the data service.

The fault monitor performs two functions:

- Monitors the abnormal exit of the data service server process
- Checks the health of the data service

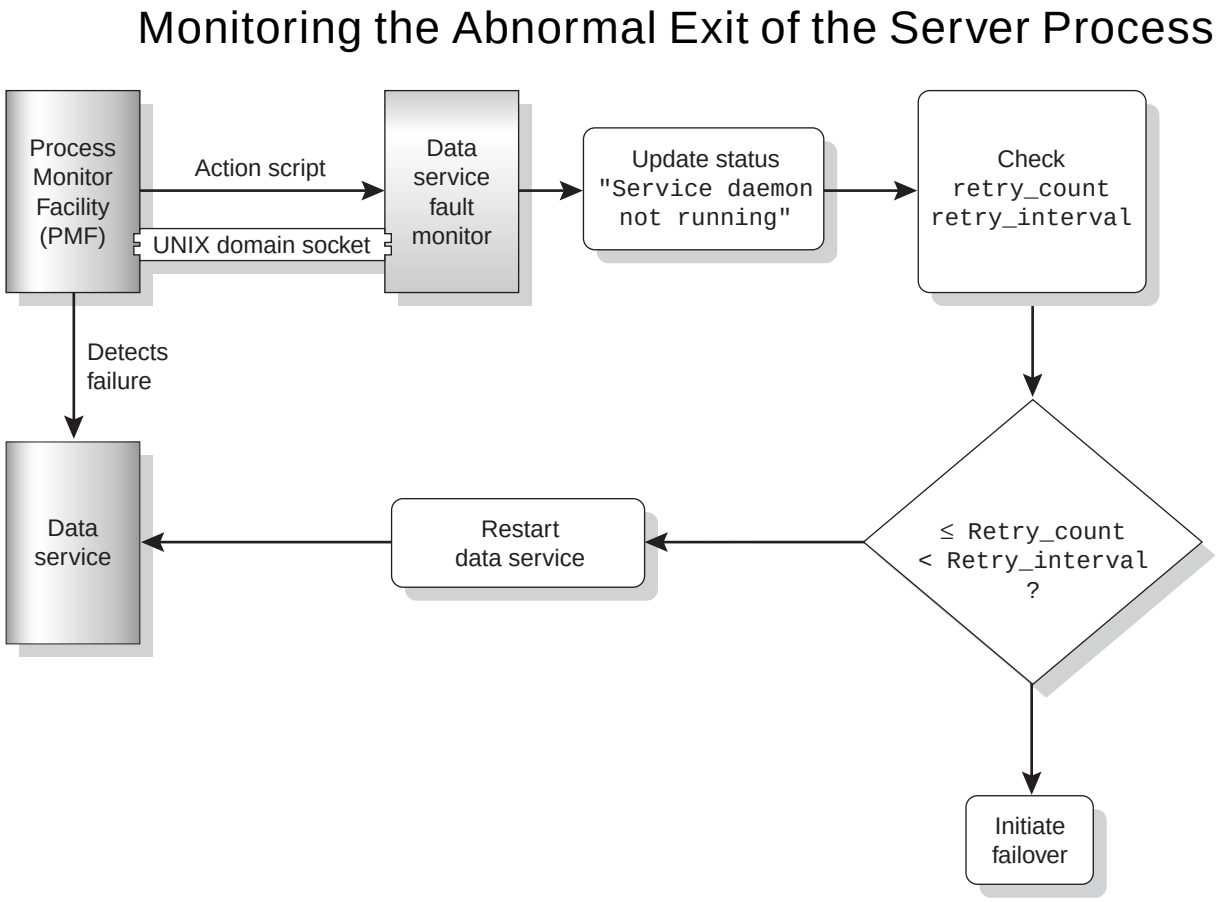

Figure 7-1 Monitoring the Abnormal Exit of the Server Process

The Process Monitor Facility (PMF) monitors the data service process. On abnormal exit, the PMF invokes an action script, supplied by the data service, to communicate the failure to the data service fault probe.

Communication between the PMF action script and the data service fault probe occurs over a UNIX domain socket. The only communication intended to take place through the UNIX domain socket occurs when the PMF informs the probe, through the action script, that the data service has exited abnormally. This event is considered a total failure of the data service.

The data service fault probe polls for messages from the PMF action script. It runs in an infinite loop and sleeps for an adjustable amount of time. Adjust the sleep time using the resource property `Thorough_probe_interval`.

When the probe is notified that the data service has exited abnormally, it updates the status of the data service as `Service daemon not running`, and takes action. The action can involve restarting the data service locally or failing over the data service to a secondary cluster node. The probe decides whether to restart or to fail over the data service by checking the value set in the resource properties `Retry_count` and `Retry_interval` for the data service application resource. The probe will attempt to restart the data service `Retry_count` times over the `Retry_interval` seconds.

Health Checks of the Data Service

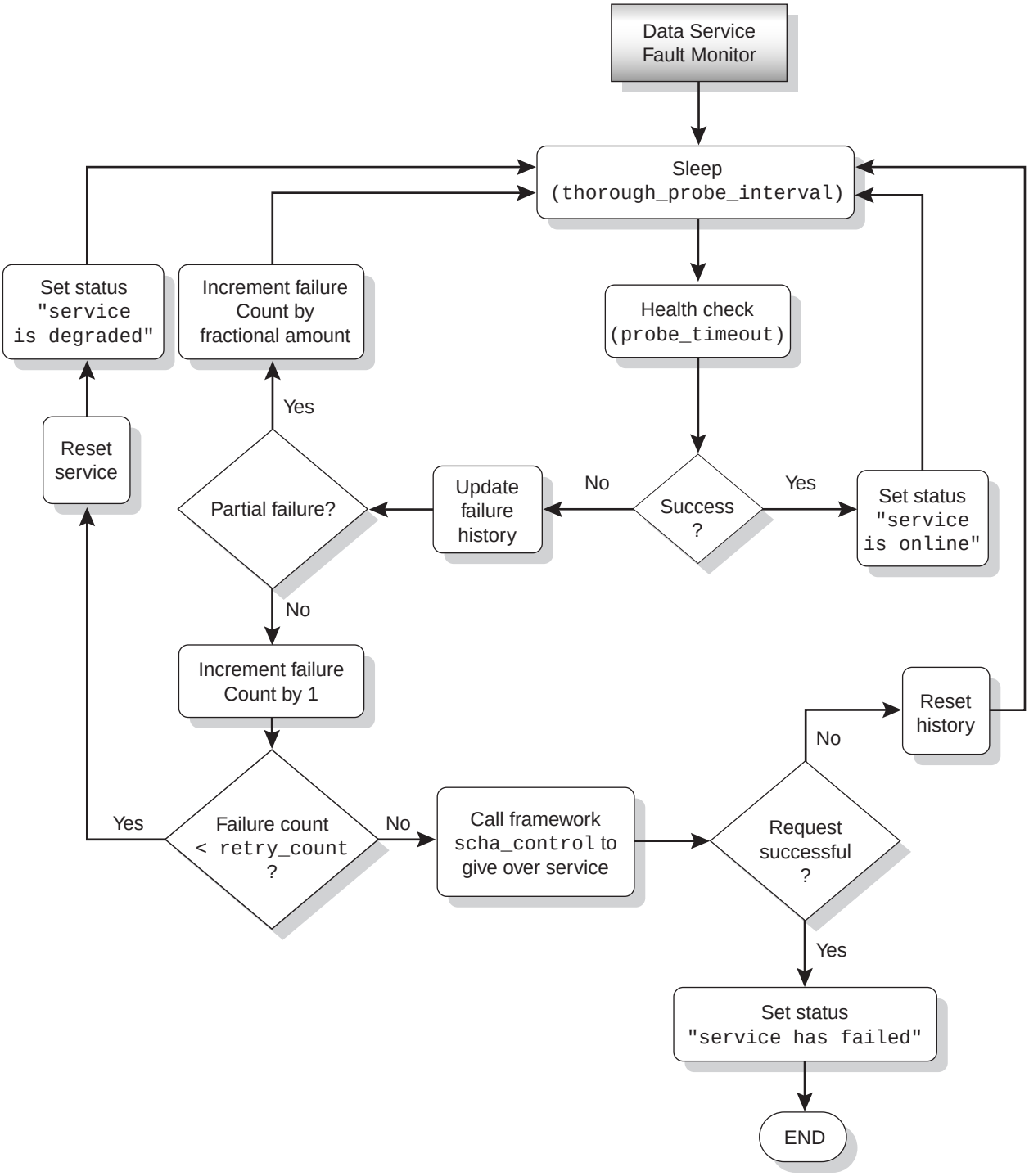

Figure 7-2 Health Checks of the Data Service Process

Typically, communication between the probe and the data service occurs through a dedicated command or a successful connection to the specified data service port.

If no messages have been received on the control socket, the probe, after sleeping for an interval specified by `Thorough_probe_interval`, checks the health of the data service. The logic followed by the probe is illustrated in Figure 7-2 on page 7-6, Health Checks of the Data Service and is also explained below:

The probe will:

1. Sleep for `Thorough_probe_interval` seconds.
2. Perform health checks under a time-out property `Probe_timeout`. This is a resource extension property of each data service that you can set. It indicates the time lapse in seconds after which the probe concludes that the health check has failed.
3. Update the success/failure history by purging any history records that are older than the value set for the resource property `Retry_interval` (If the result of Step 2 is a success, that is, the service is healthy). The probe sets the status message for the resource as `Service is online` and returns to Step 1.

If Step 2 resulted in a failure, the probe updates the failure history. It then computes the total number of times the health check failed. The result of the health check can range from a total failure to success. The interpretation of the result depends on the specific data service.

In some cases, the probe can successfully connect to the server and send a handshake message, but receives only a partial response before timing out. This scenario is most likely a result of system overload. In this case, you would only add to the load if some action is taken (such as restarting the service). Instead, a data service fault monitor can decide not to treat this "partial" failure as fatal. The monitor can simply track this failure as a nonfatal probe of the service. These partial failures are still accumulated over the interval specified in `Retry_interval`.

However, if the probe cannot connect to the server at all, it can be considered a fatal failure. Partial failures lead to incrementing the failure count by a fractional amount. A fatal (total) failure always increments the failure count by 1. Every time the failure count increases by 1 (either by a fatal failure or by accumulation of partial failures), the probe attempts to correct the situation either by restarting or failing over the data service.

4. If the result of the computation in Step 3 (the number of failures in the history interval) is less than the value of the resource property `Retry_count`, the probe attempts to correct the situation locally (for example, by restarting the service). The probe sets the status message of the resource as `Service is degraded` and returns to Step 1.

5. If the number of failures in `Retry_interval` seconds exceeds `Retry_count`, the probe calls `scha_control` in the cluster framework with the `giveover` option. This option requests failover of the service.

If this request succeeds, the fault probe stops on this node. The probe sets the status message for the resource as: `Service has failed`.

The `scha_control` request issued in the previous step can be denied by the Sun Cluster framework because of various reasons; the reason is identified by the return code of `scha_control`. The probe checks the return code. If the `scha_control` is denied, the probe resets the failure / success history and starts afresh.

The history must be reset since the number of failures is already above `Retry_count`, and the fault probe would attempt to issue `scha_control` over and over again. This request would just place additional load on the system. The probe then returns to Step 1.

Fault Monitor Resource Properties

Each data service and its monitor has access to system-defined, standard resource properties that are common to all data services and monitors (for example, `Start_timeout`). In addition, each data service and its monitor has a set of extension properties that is specific to it. Many standard and extension properties are configured with default settings that may or may not be appropriate for a particular environment. Some of these properties may be configured only during the resource or resource group's creation while others may be modified after the resource or resource group has been created or disabled.

Table 7-1 lists the standard resource properties related to data service fault monitoring.

Table 7-1 Data Service Fault Monitor Resources

Resource Property	Description	Updatable	Default
`Cheap_probe_interval`	The number of seconds between invocations of a quick fault probe for the resource	Resource Type Dependent	Resource Type Dependent
`Thorough_probe_interval`	The number of seconds between invocations of a high-overhead fault probe for the resource	Resource Type Dependent	Resource Type Dependent
`Retry_count`	The number of times a fault monitor should attempt to restart a failed resource before giving up	Resource Type Dependent	Resource Type Dependent
`Retry_interval`	The number of seconds over which to count attempts (`Retry_count`) to restart a failed resource	Resource Type Dependent	Resource Type Dependent

Table 7-1 Data Service Fault Monitor Resources

Resource Property	Description	Updatable	Default
<METHOD>_TIMEOUT where <METHOD> is: START, STOP, PRENET_START, POSTNET_STOP, INIT, FINI, BOOT, VALIDATE, UPDATE, MONITOR_START, MONITOR_STOP, MONITOR_CHECK	The time, in seconds, that the RGM will allow a particular method to complete before considering the method invocation a failure	Any time	3600
Failotver_mode	Controls the response of the RGM to resource start or stop failure or timeout. Can be set to NONE, SOFT or HARD: **NONE** - Sets the resource state to a failure state and waits for operator intervention. **SOFT** - For START failure, the RGM relocates the resource's group to another node. For STOP failure, the RGM leaves the resource in STOP_FAILED state and waits for operator intervention. **HARD** - If the resource startup fails, the resource's group will be relocated to another node. If the resource stop fails, the node will be aborted.	Any time	Resource Type Dependent

If a resource property is listed as updatable only when disabled, the associated resource and resource group must be taken offline. Directions to accomplish this are provided in the "Tuning Data Service Fault Monitors" section later in the module.

Resource Type Registration (RTR) File

If a resource property's default is resource type dependent, its default is defined in the resource type's Resource Type Registration (RTR) file. This file is usually located in `/opt/<dataservice_package_name>/etc/SUNW.<resource_type>`.

For example, the RTR file for the HA for DNS data service is: `/opt/SUNWscdns/etc/SUNW.dns`.

The RTR files for the `SUNW.HAStorage`, `SUNW.LogicalHostname`, and `SUNW.SharedAddress` resource types are stored in `/usr/cluster/lib/rgm/rtreg`. This directory also contains symbolic links to the other RTR files.

Note – The RTR file contains not only default resource properties, but also other useful information such as the names of the data service's methods and the description of its extension properties. This file is not meant to be edited directly by the end user. Although the cluster should still operate, the results are unpredictable if the service is later unregistered and reregistered. Changes to resource properties should be accomplished via the `scrgadm` command which is explained later in this module.

Sample Resource Type Registration file

```
#
# Copyright (c) 1998-2000 by Sun Microsystems, Inc.
# All rights reserved.
#
#ident "@(#)SUNW.dns  1.13     00/03/02 SMI"

# Registration information and Paramtable for Domain Name Service (DNS)
#
# NOTE: Keywords are case insensitive, i.e. users may use any
# capitalization style they wish
#

RESOURCE_TYPE = "dns";
VENDOR_ID = SUNW;
RT_DESCRIPTION = "Domain Name Service on Sun Cluster";

RT_VERSION ="1.0";
API_VERSION = 2;
FAILOVER = TRUE;

RT_BASEDIR=/opt/SUNWscdns/bin;

FAILOVER = TRUE;

START       =  dns_svc_start;
STOP        =  dns_svc_stop;

INIT        =  dns_init;
BOOT        =  dns_boot;
FINI        =  dns_fini;

# Although we do define the abort method, since all we need to
# do is same as in STOP. This will, in reality, be a link.

VALIDATE  =  dns_validate;
UPDATE      =  dns_update;

MONITOR_START=dns_monitor_start;
MONITOR_STOP=dns_monitor_stop;
MONITOR_CHECK=dns_monitor_check;

PKGLIST = SUNWscdns;

# The paramtable is a list of bracketed resource property declarations
```

```
# that come after the resource-type declarations
# The property-name declaration must be the first attribute
# after the open curly of a paramtable entry
#
{
    PROPERTY = Start_timeout;
    MIN=0;
    DEFAULT=300;
    TUNABLE = ANYTIME;
}
{
    PROPERTY = Stop_timeout;
    MIN=0;
    DEFAULT=300;
    TUNABLE = ANYTIME;
}
{
    PROPERTY = Validate_timeout;
    MIN=0;
    DEFAULT=300;
    TUNABLE = ANYTIME;
}
{
    PROPERTY = Update_timeout;
    MIN=0;
    DEFAULT=300;
    TUNABLE = ANYTIME;
}
{
    PROPERTY = Init_timeout;
    MIN=0;
    DEFAULT=300;
    TUNABLE = ANYTIME;
}
{
    PROPERTY = Fini_timeout;
    MIN=0;
    DEFAULT=300;
    TUNABLE = ANYTIME;
}
{
    PROPERTY = Boot_timeout;
    MIN=0;
    DEFAULT=300;
    TUNABLE = ANYTIME;
}
```

```
{
    PROPERTY = Monitor_Start_timeout;
    MIN=0;
    DEFAULT=300;
    TUNABLE = ANYTIME;
}
{
    PROPERTY = Monitor_Stop_timeout;
    MIN=0;
    DEFAULT=300;
    TUNABLE = ANYTIME;
}
{
    PROPERTY = Monitor_Check_timeout;
    MIN=0;
    DEFAULT=300;
}
{
    PROPERTY = Thorough_Probe_Interval;
    MIN=0;
    MAX=3600;
    DEFAULT=60;
    TUNABLE = ANYTIME;
}
{
    PROPERTY = Retry_Count;
    MIN=0;
    MAX=10;
    DEFAULT=2;
    TUNABLE = ANYTIME;
}
{
    PROPERTY = Retry_Interval;
    MIN=0;
    MAX=3600;
    DEFAULT=300;
    TUNABLE = ANYTIME;
}
{
    PROPERTY = FailOver_Mode;
    DEFAULT = SOFT;
    TUNABLE = ANYTIME;
}
{
    PROPERTY = Network_resources_used;
    TUNABLE = WHEN_DISABLED;
```

```
    DEFAULT = "";
}
{
        PROPERTY = Port_list;
        DEFAULT = "53/udp";
        TUNABLE = AT_CREATION;
}

#
# Extension Properties
#

# Not to be edited by end user
{
    PROPERTY = Paramtable_version;
    EXTENSION;
    STRING;
    DEFAULT = "1.0";
    DESCRIPTION = "The Paramtable Version for this Resource";
}

# Must specify installation path of DNS (on PXFS)
{
    PROPERTY = Confdir_list;
    EXTENSION;
    STRINGARRAY;
    TUNABLE = AT_CREATION;
    DESCRIPTION = "The Configuration Directory Path(s)";
}

# These two control the restarting of the fault monitor itself
# (not the serer daemon) by PMF.
{
    PROPERTY = Monitor_retry_count;
    EXTENSION;
    INT;
    DEFAULT = 4;
    TUNABLE = ANYTIME;
    DESCRIPTION = "Number of PMF restarts allowed for the fault monitor";
}

{
    PROPERTY = Monitor_retry_interval;
    EXTENSION;
    INT;
    DEFAULT = 2;
```

```
    TUNABLE = ANYTIME;
    DESCRIPTION = "Time window (minutes) for fault monitor restarts";
}

# Time out value for the probe
{
    PROPERTY = Probe_timeout;
    EXTENSION;
    INT;
    DEFAULT = 30;
    TUNABLE = ANYTIME;
    DESCRIPTION = "Time out value for the probe (seconds)";
}

{
    PROPERTY = DNS_mode;
    EXTENSION;
    STRING;
    DEFAULT = "conf";
    TUNABLE = AT_CREATION;
    DESCRIPTION = "Configuration file (named.conf or named.boot) to use";
}
```

The pmfadm Command Line Interface

The pmfadm command provides the administrative command line interface to the process monitor facility. It is normally invoked during the boot process to start such daemons as the public network management daemon (pnmd).

However, you can use it to determine which services are currently being monitored by the process monitor facility by issuing the command:

```
# pmfadm -L
```

Examples:

1. To determine which services are currently monitored, type:

```
# pmfadm -L
```

Your output will appear similar to:

```
tags: scsymon pnmd
```

2. To determine the status of a monitored service by:

```
# pmfadm -l <tag_name>
```

3. To check the status of the Public Network Management daemon:

```
# pmfadm -l pnmd
```

Your output will appear similar to:

```
pmfadm -c pnmd -n -1 /usr/cluster/bin/pnmd
        retries: 0
        owner: root
        monitor children: up to level 1
        pids: 328
```

4. To check the status of all monitored processes:

```
# pmfadm -l ""
```

Your output will appear similar to:

```
STATUS scsymon
pmfadm -c scsymon -n 3 /usr/cluster/lib/scsymon/scsymon_srv
```

```
        retries: 0
        owner: root
        monitor children: all
        pids: 349
STATUS pnmd
pmfadm -c pnmd -n -1 /usr/cluster/bin/pnmd
        retries: 0
        owner: root
        monitor children: up to level 1
        pids: 328
```

Note – If `pmfadm` is monitoring a large number of processes (at least 70 or more), this command will cause `rpc.pmfd` to dump core and panic the node. This is a known bug that will eventually be corrected with a patch or later release of the Sun Cluster software. Use this command with care, particularly on a large cluster.

Data Service Monitors for Sun-Supplied Data Services

Sun supplies data services monitors for the following data services:

- Sun Cluster HA for Apache Fault Monitor
- Sun Cluster HA for DNS Fault Monitor
- Sun Cluster HA for NFS Fault Monitor
- Sun Cluster HA for Oracle Fault Monitor
- Sun Cluster HA for iPlanet Web Server Fault Monitor
- Sun Cluster HA for Netscape Directory Server Fault Monitor
- Sun Cluster HA for Sybase ASE Fault Monitor
- Sun Cluster HA for SAP 4.6d Fault Monitor

Each monitor performs the health checks and takes actions specific to the needs of its data service.

Additional data services with their own fault monitors may be added between releases of the Sun Cluster 3.0 product. Consult the Configuration Guide at: `http://suncluster.eng/products/SC3.0/config` for the latest list of supported data services.

Note – Sun Cluster for Oracle Parallel Server (OPS) uses its own methods for fault monitoring.

Sun Cluster HA for Apache Fault Monitor (apache_probe)

The Sun Cluster HA for Apache fault monitor illustrates the application of the data service fault monitoring process. It is described in detail in the sections that follow.

The Sun Cluster HA for Apache probe sends a request to the server to query the health of the Apache server. Before the probe actually queries the Apache server, it checks to confirm that network resources are configured for this Apache resource. If no network resources are configured, an error message (`No network resources found for resource`) is logged and the probe exits with failure.

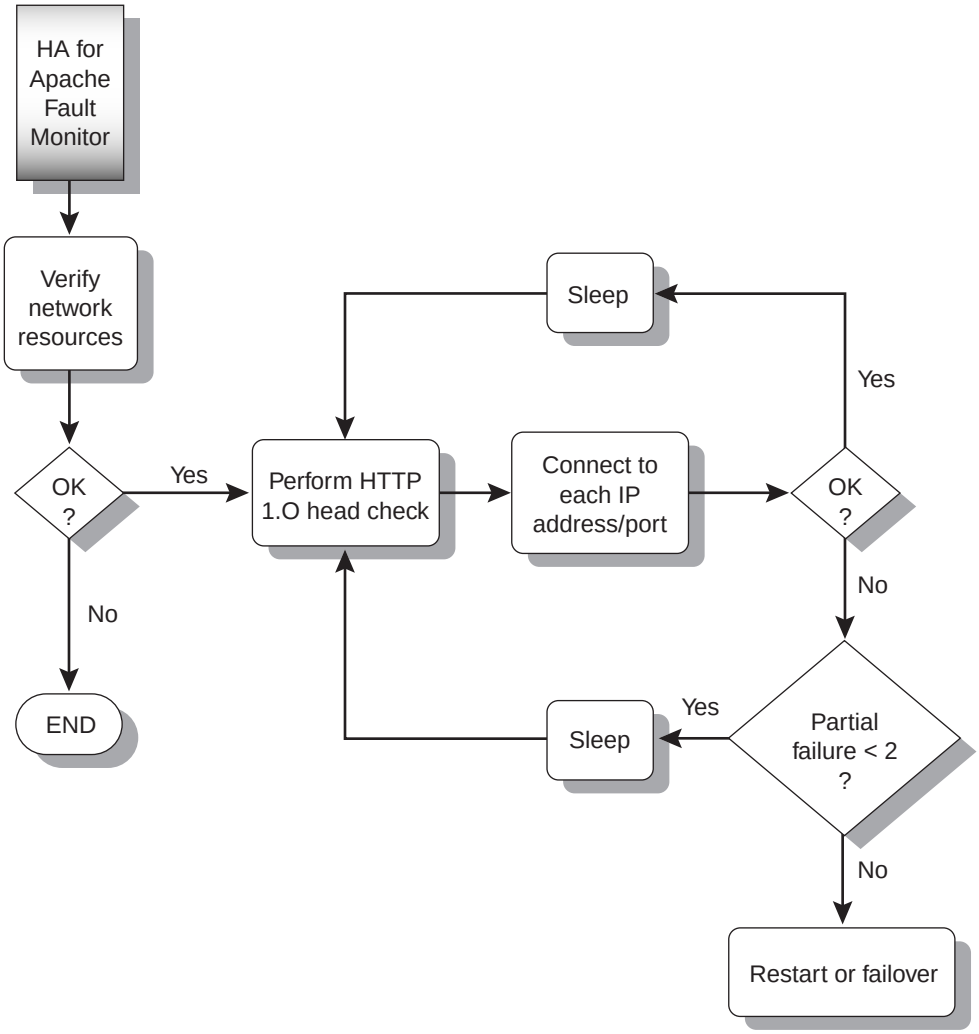

Figure 7-3 Sun Cluster HA for Apache Fault Monitor Process

The probe:

1. Uses the timeout value set by the resource property `Probe_timeout` to limit the time spent trying to successfully probe the Apache server.

2. Connects to the Apache server and performs an HTTP 1.0 HEAD check by sending the HTTP request and receiving a response. In turn, the probe connects to the Apache server on each IP address/port combination.

   The result of this query is either failure or success. If the probe successfully receives a reply from the Apache server, the probe returns to its infinite loop and continues the next cycle of probing and sleeping.

   The query can fail for various reasons, such as heavy network traffic, heavy system load, and misconfiguration. Misconfiguration can occur if the Apache server is not configured to be listening on all IP address/port combinations that are being probed. The Apache server should service every port for every IP address specified for this resource.

   If the reply to the query is not received within the `Probe_timeout` limit, the probe considers this scenario a failure on the part of Apache data service and records the failure in its history. An Apache probe failure can be a total failure or a partial failure.

   Probe failures that are considered total failures are:

   - Failure to connect to the server, as flagged by the error message: `Failed to connect to %s port %d`, with `%s` being the host name and `%d` the port number
   - Running out of time (exceeding the resource property time-out `Probe_timeout`) when trying to connect to the server
   - Failure to successfully send the probe string to the server, as flagged by the error message: `Failed to communicate with server %s port %d: %s`, with the first `%s` being the host name, `%d` the port number, and the second `%s` further details about the error
   - Two partial failures within the resource property interval `Retry_interval`

   Probe failures considered partial failures are:

- Running out of time (exceeding the resource property timeout `Probe_timeout`) while trying to read the reply from the server to the probe's query.
- Failing to read data from the server for other reasons, as flagged by the error message: `Failed to communicate with server %s port %d: %s`, with the first `%s` being the host name and `%d` the port number; the second `%s` further details about the error.

Based on the history of failures, a failure can cause either a local restart or a failover of the data service. This process was described in "Health Checks of the Data Service".

Note – If the restart on the local services fails and there is no available node for failover, the service will continue the restart/failover attempts in an infinite loop. This is a known bug (Bug ID#4340961) that had no fix at initial release.

Sun Cluster HA for Apache Monitor Resource Properties

The following table describes the standard resource properties for the Sun Cluster HA for Apache fault monitor.

Table 7-2 HA for the Apache Monitor Standard Resource Properties

Resource Property	Description	Updatable	Default
`Cheap_probe_interval`	The number of seconds between invocations of a quick fault probe for the resource	At Creation Min: 0 Max: 3600	60 (Seconds)
`Thorough_probe_interval`	The number of seconds between invocations of a high-overhead fault probe for the resource	Any Time Min: 0 Max: 3600	60 (Seconds)
`Retry_count`	The number of times a fault monitor should attempt to restart a failed resource before giving up	Any Time Min: 0 Max: 10	2
`Retry_interval`	The number of seconds over which to count attempts (`Retry_count`) to restart a failed resource	Any Time Min: 0 Max: 3600	300 (seconds)

`<METHOD>_TIMEOUT`
where `<METHOD>` is:
`START, STOP, INIT, FINI, BOOT, VALIDATE, UPDATE, MONITOR_START, MONITOR_STOP, MONITOR_CHECK`
	The time, in seconds, that the RGM will allow a particular method to complete before considering the method invocation a failure	Any time
		Min: 60
			300
			(seconds)

Table 7-2 HA for the Apache Monitor Standard Resource Properties (Continued)

Resource Property	Description	Updatable	Default
`Failover_mode`	Controls the response of the RGM to resource start or stop failure or timeout. Can be set to `NONE`, `SOFT` or `HARD`: **NONE** - Sets the resource state to a failure state and waits for operator intervention. **SOFT** - For `START` failure, the RGM relocates the resource's group to another node. For `STOP` failure, the RGM leaves the resource in `STOP_FAILED` state and waits for operator intervention. **HARD** - If the resource startup fails, the resource's group will be relocated to another node. If the resource stop fails, the node will be aborted	Any time	SOFT

The following table describes the extension properties for the Sun Cluster HA for Apache fault monitor.

Table 7-3 HA for Apache Monitor Extension Properties

Extension Property	Description	Updatable	Default
`Monitor_retry_count`	Number of times to set attempt to restart the fault monitor. When set to -1, the restart will be attempted an infinite number of times.	Any Time Min: -1	4
`Monitor_retry_interval`	The time window (in minutes) to measure fault monitor restarts (See the `Monitor_retry_count` extension property). If the number of failed restarts exceeds the `Monitor_retry_count` within the `Monitor_retry_interval`, the fault monitor is not restarted. When set to -1, the retry_interval is infinite.	Any Time Min: -1	2
`Probe_timeout`	Time out value (in seconds) used for the fault monitor probe	Any Time Min: 15	30

Tuning a Data Service Fault Monitor

Any of the resource properties listed as updatable may be tuned by an administrator using the `scrgadm` command. Some resources may only be tuned at the resource's creation, some may be tuned when the resource is created, and others may be tuned at any time.

Tuning a resource involves the following steps:

1. View the current configuration settings for the resource.
2. If necessary, disable the resource.
3. Modify the resource property using the `scrgadm` command.
4. If necessary, re-enable the resource.

Viewing the Current Configuration Settings for the Resource

Perform this procedure from any node in the cluster.

1. Become superuser on a node in the cluster.
2. Use the `scrgadm` command to determine the name of the resource and its associated resource group.

   # **`scrgadm -pv`**

The `scrgadm` command provides three levels of configuration status information:

With the `-p` option, the output shows a very limited set of property values for resource types, resource groups, and resources.

With the `-pv` option, the output shows more details on other resource type, resource group, and resource properties.

With the `-pvv` option, the output provides a detailed view, including resource type methods, extension properties, and all resource and resource group properties.

You can also view specific resource types, resource groups, and resources by using the `-t`, `-g`, and `-j` (resource type, resource group, and resource, respectively) options, followed by the name of the object you want to view.

For example, the following command specifies that you want to view specific detailed information on the resource apache-1 only.

```
# scrgadm -pvv -j apache-1
```

Disabling the Resource (If Necessary)

If a resource property is listed as updatable only when disabled, you will need to disable the resource before modifying its properties. To do this, you will also need to disable the resource group to which it belongs and any other resources in the resource group. This is rarely necessary for the Sun-supplied data service fault monitor resources.

Note – This will make the resource unavailable to users and should only be done during a scheduled maintenance period.

1. Use the output from the `scrgadm -pv` command in the previous step to determine the name of the resource's resource group and any other resources in the group.
2. Disable the resource.

```
# scswitch -n -j resource-name
```

The `-n` option disables the resource. The `-j` resource-name option specifies the name of the resource to disable.

3. Verify that the resource is disabled.

```
# scrgadm -pv -j resource-name
```

Modifying the Resource Property

A resource property can be modified either through the Sun Management Center GUI or through the command line. The following procedure explains how to modify the property through the command line.

1. Use the `scrgadm -pvv` command to view the current resource property settings.

```
# scrgadm -pvv -j resource-name
```

2. Change the resource property.

```
# scrgadm -c -j resource-name -y property=new-value \
-x extension-property=new-value
```

The `-c` option changes the specified property.

The `-j` resource-name option specifies the name of the resource.

The `-y property=new-value` specifies the name of the standard property to change.

The `-x extension-property=new-value` specifies the name of the extension property to change.

3. Verify that the resource property has been changed.

```
# scrgadm -pvv -j resource-name
```

Re-Enabling the Resource (If Necessary)

If you previously disabled the resource, execute the following steps to bring it back online.

1. Enable the resource.

```
# scswitch -e -j resource-name
```

The `-e` option enables the resource.

The `-j` resource-name specifies the name of the resource.

2. Verify that the resource is enabled.

```
# scrgadm -pv -j resource-name
```

Check the `Res Enabled` field to see if it is enabled.

Example: Changing a Standard Resource Property

The following example lists the steps to change the system-defined `Start_timeout` property for the resource (`r-1`).

1. Get the name of the resource and resource group.

```
# scrgadm -pv
```

```
Res Group name:                                 apache-rg
  (apache-rg) Res Group RG_description:         <NULL>
  (apache-rg) Res Group mode:                   Scalable
  (apache-rg) Res Group management state:       Managed
  (apache-rg) Res Group Failback:               False
  (apache-rg) Res Group Nodelist:               pnode1 pnode2
  (apache-rg) Res Group Maximum_primaries:      2
  (apache-rg) Res Group Desired_primaries:      2
  (apache-rg) Res Group RG_dependencies:        ap-server-sa-rg
  (apache-rg) Res Group network dependencies:   True
  (apache-rg) Res Group Global_resources_used:  All
  (apache-rg) Res Group Pingpong_interval:      3600
  (apache-rg) Res Group Pathprefix:             <NULL>

  (apache-rg) Res name:                         r-1
    (apache-rg:r-1) Res R_description:       <NULL>
    (apache-rg:r-1) Res resource type:       SUNW.apache
    (apache-rg:r-1) Res resource group name: apache-rg
    (apache-rg:r-1) Res enabled:             True
    (apache-rg:r-1) Res monitor enabled:     True
```

2. Check the existing value of `Start_timeout`.

```
# scrgadm -pvv -j r-1 | grep -i start_timeout
```

```
(apache-rg:r-1) Res property name:         START_TIMEOUT
    (apache-rg:r-1:START_TIMEOUT) Res property class: standard
    (apache-rg:r-1:START_TIMEOUT) Res property type: int
    (apache-rg:r-1:START_TIMEOUT) Res property value: 300
```

3. Change the `Start_timeout` property to 60 seconds.

```
# scrgadm -c -j r-1 -y start_timeout=60
```

4. Verify that the change has been made.

```
# scrgadm -pvv -j r-1
```

```
(apache-rg:r-1) Res property name:         START_TIMEOUT
    (apache-rg:r-1:START_TIMEOUT) Res property class: standard
    (apache-rg:r-1:START_TIMEOUT) Res property type: int
    (apache-rg:r-1:START_TIMEOUT) Res property value: 60
```

Example: Changing an Extension Resource Property

The following example lists the steps to change the extension `Probe_timeout` property for the resource (`r-1`).

1. Get the name of the resource and resource group.

```
# scrgadm -pv
```

```
Res Group name:                                 apache-rg
  (apache-rg) Res Group RG_description:         <NULL>
  (apache-rg) Res Group mode:                   Scalable
  (apache-rg) Res Group management state:       Managed
  (apache-rg) Res Group Failback:               False
  (apache-rg) Res Group Nodelist:               pnode1 pnode2
  (apache-rg) Res Group Maximum_primaries:      2
  (apache-rg) Res Group Desired_primaries:      2
  (apache-rg) Res Group RG_dependencies:        ap-server-sa-rg
  (apache-rg) Res Group network dependencies:   True
  (apache-rg) Res Group Global_resources_used:  All
  (apache-rg) Res Group Pingpong_interval:      3600
  (apache-rg) Res Group Pathprefix:             <NULL>

  (apache-rg) Res name:                         r-1
    (apache-rg:r-1) Res R_description:       <NULL>
    (apache-rg:r-1) Res resource type:       SUNW.apache
    (apache-rg:r-1) Res resource group name: apache-rg
    (apache-rg:r-1) Res enabled:             True
    (apache-rg:r-1) Res monitor enabled:     True
```

2. Check the existing value of `Probe_timeout`.

```
# scrgadm -pvv -j r-1 | grep -i probe_timeout
```

```
(apache-rg:j-1) Res property name:         Probe_timeout
    (apache-rg:j-1:Probe_timeout) Res property class: extension
    (apache-rg:j-1:Probe_timeout) Res property description: Time out
value for the probe (seconds)
    (apache-rg:j-1:Probe_timeout) Res property type: int
    (apache-rg:j-1:Probe_timeout) Res property value: 30
```

3. Change the `Probe_timeout` property to 60.

```
# scrgadm -c -j r-1 -x probe_timeout=60
```

4. Verify that the change has been made.

```
            # scrgadm -pvv -j r-1
(apache-rg:j-1) Res property name:         Probe_timeout
    (apache-rg:j-1:Probe_timeout) Res property class: extension
    (apache-rg:j-1:Probe_timeout) Res property description: Time out
value for the probe (seconds)
    (apache-rg:j-1:Probe_timeout) Res property type: int
    (apache-rg:j-1:Probe_timeout) Res property value: 60
```

Module 7 Lab: Tuning a Data Service Fault Monitor

The purpose of the following lab is to tune a data service fault monitor.

Refer to the material in your student guide to help you through the lab.

1. Change the `retry_count` resource property for the Apache web server fault monitor to its maximum value.

   What value did you set it to? _____

   What was its original value? _____

2. Change the `retry_count` back to its original value.

3. Change the `monitor_retry_count` resource property for the Apache web server so that the fault monitor attempts to restart an infinite number of times.

   What value did you set it to? _____

   What was its original value? _____

4. Change the `monitor_retry_count` back to its original value.

Module 8

Tuning the Cluster

Objectives

This module introduces various tunable parameters in Sun Cluster 3.0.

By the end of this module you should be able to:

- Implement scalable service load balancing
- Tune the transport
- Tune PXFS
- Identify `/etc/system` tunables
- Tune Public Network Management

Additional Resources

The following references can provide additional details on the topics discussed in this module:

- *Sun Cluster 3.0 U1 System Administration Guide*, 806-7073-10
- *Sun Cluster 3.0 U1 Installation Guide*, 806-7069-10
- *Sun Cluster 3.0 U1 Data Services Installation and Configuration Guide*, 806-7071-10
- *Sun Cluster 3.0 U1 Data Services Developer Guide*, 806-7072-10
- *Sun Cluster 3.0 U1 Concepts*, 806-7074-10
- Sun Cluster 3.0 Cluster File System (CFS): Making the most of the global file system, a Technical White Paper by Tim Read, `http://galileo.eng/docs/pxfs/cfs-wp.pdf`

Scalable Resource Load Balancing

Service requests come into the cluster through a single network interface (the global interface or GIF) and are distributed to the nodes based on one of several predefined algorithms set by the load-balancing policy. The cluster can use the load-balancing policy to balance the service load between several nodes. Note that there can be multiple GIFs on different nodes hosting other shared addresses.

Load Balancing Policies

Load balancing improves performance of the scalable service, both in response time and in throughput.

There are two classes of scalable data services: pure and sticky. A pure service is one where any instance of it can respond to client requests. A sticky service is one where a client sends requests to the same instance. Those requests are not redirected to other instances.

Pure Service

A pure service uses a weighted load-balancing policy. Under this load-balancing policy, client requests are by default uniformly distributed over the server instances in the cluster. For example, in a three-node cluster, each node has the weight of 1. This means that each node will service one third of the requests from any client on behalf of that service. Weights can be changed at any time by the administrator through the `scrgadm` command interface.

This policy is set using the `LB_WEIGHTED` value for the `Load_balancing_weights` property. If a weight for a node is not explicitly set, the weight for that node defaults to one.

Note that this policy is not round-robin. A round-robin policy would always cause each request from a client to go to a different node: the first request to node 1, the second request to node 2, and so on. The weighted policy guarantees that a certain percentage of the traffic from clients is directed to a particular node. This policy does not address individual requests.

Sticky Service

Sticky services allow concurrent application-level sessions over multiple TCP connections to share in-state memory (application session state). A sticky service has two flavors: ordinary sticky and wildcard sticky.

Ordinary Sticky

Ordinary sticky services permit a client to share state between multiple concurrent TCP connections. The client is said to be "sticky" with respect to that server instance listening on a single port. The client is guaranteed that all of his requests go to the same server instance, provided that instance remains up and accessible and the load balancing policy is not changed while the service is online.

An example of this is a web browser on a client that connects to a shared IP address on port 80 using three different TCP connections. The connections exchange cached session information between them at the service.

A generalization of a sticky policy extends to multiple scalable services exchanging session information behind the scenes at the same instance. When these services exchange session information behind the scenes at the same instance, the client is said to be "sticky" with respect to multiple server instances on the same node listening on different ports.

For example, a customer on an e-commerce site fills his shopping cart with items using ordinary HTTP on port 80, but switches to SSL on port 443 to send secure data in order to pay by credit card for the items in the cart.

In this policy, the set of ports is known at the time the application resources are configured. This policy is set using the `LB_STICKY` value for the `Load_balancing_policy` resource property.

Wildcard Sticky

Wildcard sticky services use dynamically assigned port numbers, but still expect client requests to go to the same node. The client is "sticky wildcard" over ports with respect to the same IP address.

A good example of this policy is passive mode FTP. A client connects to an FTP server on port 21 and is then informed by the server to connect back to a listener port server in the dynamic port range. All requests for this IP address are forwarded to the same node that the server informed the client through the control information.

For a scalable service identified by the IP address, ports are assigned by the server (and are not known in advance). The ports might change. This policy is set using the `LB_STICKY_WILD` value for the `Load_balancing_policy` resource property.

Note – For each of these sticky policies, the weighted load-balancing policy is in effect by default. Thus, a client's initial request is directed to the instance dictated by the load balancer. After the client has established an affinity for the node where the instance is running, future requests are directed to that instance as long as the node is accessible and the load balancing policy is not changed.

Scalable Resource Groups

The Resource Group Manager (RGM) provides a number of properties that support the implementation of a scalable resource. In some cases, it may be necessary to tune these for the purpose of load balancing.

When configuring a scalable resource group, you must first specify the maximum and desired number of primary nodes as well as the dependency between this resource group and the failover resource group.

Table 8-1 Setting Nodes for Scalable Resource Groups

Resource Property	Description
`RG_mode`	Specifies the resource group as failover or scalable. If RG_mode is `scalable`, the RGM allows Maximum_primaries to have a value greater than 1, meaning the group can mastered by multiple nodes simultaneously. The RGM does not allow a resource whose failover property is TRUE to be instantiated in a resource group whose RG_mode is `scalable`.
`Maximum_primaries`	Specifies the number of active primary nodes allowed for this resource group. The default for this value is 1.
`Desired_primaries`	Specifies the desired number of primary nodes allowed for this resource group. The default for this value is 1.

Table 8-1 Setting Nodes for Scalable Resource Groups (Continued)

Resource Property	Description
RG_dependencies	Identifies the resource group that contains the shared address resource on which the resource group depends.

If the data service developer declares the Scalable property to be TRUE in the RTR file for a resource, the RGM automatically creates the following set of properties for that resource:

Table 8-2 Scalable Resource Properties

Resource Property	Description
Network_resource_used	Identifies the shared address resource used by this resource. This property is empty by default so the system administrator must provide the actual list of shared addresses the scalable service will use when creating the resource.
Load_balancing_policy	Specifies the load balancing policy for the resource. You can explicitly set the policy or use the default LB_WEIGHTED. In either case, the system administrator can modify the default when creating the resource.
LB_WEIGHTED	Load is distributed among nodes according to weights set by the Load_balancing_weights property.
LB_STICKY	Allows clients of the scalable service to always connect to the same cluster node.
LB_STICKY_WILD	Allows clients that connect to an IP address of a wildcard sticky service to always connect to the same cluster node regardless of the port it is coming to.

Table 8-2 Scalable Resource Properties (Continued)

Resource Property	Description
`Load_balancing_weights`	This property specifies the load to be sent to each node. The format is `weight@node1`, `weight@node2`, `weight@node3`, where weight is an integer reflecting the portion of load distributed to the specified node. The percentage of the load distributed to a node is determined by dividing the weight by the total of all weights. For example, if `Load_balancing_weights=2@node1,1@node2,1@node3`, then `node1` would receive 2/4 (or 1/2) of the load, and `node2` and `node3` would each receive 1/4.
`Port_list`	This property determines which port the server is listening on. The default is an empty string. The data service developer can provide a list of comma-separated strings in the RTR file. This property may also be changed during resource creation.

Note – It is important to note that if the `Load_balancing_policy` is `LB_STICKY` or `LB_STICKY_WILD`, changing `Load_balancing_weights` while the service is online can cause existing clients to be reset. In that case, a different node might service the client's request even if the client had been previously serviced by another node.

Implementing a Scalable Data Service with Load Balancing

Implementing a scalable data service using load balancing involves the following steps:

1. Create an entry in the `/etc/hosts` file for the logical host address.
2. Register the data service resource type.
3. Create a failover resource group to contain the shared address.
4. Add a shared address resource to the failover resource group.
5. Create a scalable resource group.
6. Add an iPlanet/Apache instance with load balancing enabled.
7. Bring the new failover and scalable resource groups online.

Example: Implementing an iPlanet Data Service with Load Balancing

The following is an example of how to create an iPlanet instance using the command line interface. This procedure should be performed on a single cluster node:

1. Create an entry in the `/etc/hosts` file for the new logical hosts address.

```
ns-server      aaa.bbb.ccc.ddd      #iPlanet logical host
```

2. Register the iPlanet resource type.

```
# scrgadm -a -t SUNW.nshttp
```

3. Create the failover resource group.

```
# scrgadm -a -g ns-server-sa -h mars,venus,earth
```

4. Add a shared address resource to the failover resource group created in Step 3.

```
# scrgadm -a -S -g ns-server-sa -l ns-server -n \
nafo0@mars,nafo0@venus,nafo@earth
```

5. Create a scalable resource group.

```
# scrgadm -a -g netscape-sa \
-y Maximum_primaries=3
-y Desired_primaries=3
-y RG_dependencies=ns-server-sa
```

6. Add an iPlanet instance with load balancing enabled.

```
# scrgadm -a -j netscape-res -g netscape-sa \
-t SUNW.nshttp
-x Confdir_list=/global/nshttp/https-ns-server
-y Scalable=TRUE
-y Network_resource_used=ns-server
-y Load_balancing_policy=LB_WEIGHTED
-y Load_balancing_weights=1@node1,1@node2,1@node3
```

   In this example, each node will receive approximately the same number of requests.

7. Bring the new resource groups online using the `scswitch` command.

```
# scswitch -Z -g ns-server-sa

# scswitch -Z -g netscape-sa
```

8. Verify resource group configuration using scconf.

   # **scrgadm -pvv -j netscape-res**

Modifying a Load Balancing Policy

Load_balancing_policy is set to LB_WEIGHTED by default. This value can only be set resource creation. If you wish to change the Load_balancing_policy to something other than LB_WEIGHTED, you must recreate the resource. You can, however, modify Load_balancing_weights using the following procedure:

1. View the existing resource property settings.

   # **scrgadm -pvv -j netscape-res**

2. Modify the resource property to enable load balancing.

   # **scrgadm -c -j netscape-res -y **
   Load_balancing_weights=2@node1,1@node2,1@node3

   In this example, node 1 will now receive one half of the requests and nodes 2 and 3 will each receive one fourth.

3. Verify the change.

   # **scrgadm -pvv -j netscape-re**s

PXFS Tuning

The following sections describe global file system mount options (syncdir and forcedirectio) and disk device group considerations.

Global File System Mount Options

The following sections describe the syncdir and the forcedirectio mount options.

The syncdir Mount Option

The syncdir mount option can be used for cluster file systems. If you specify syncdir, the writes are guaranteed to be POSIX compliant. If you do not, you will have the same behavior that is seen with UFS file systems.

For example, under some cases, without syncdir, you would not discover an out of space condition until you close a file. With syncdir (and POSIX behavior), the out-of-space condition would have been discovered during the write operation.

However, there is a significant performance improvement if you do not specify syncdir. The cases in which you could have problems if you do not specify syncdir are rare, so it is recommended that you do not specify it and receive the performance benefit.

Note – In the beta releases of the Sun Cluster 3.0, you were required to specify the syncdir option when adding a cluster file system in the /etc/vfstab file. The shipping version does not require this.

The forcedirectio Mount Option

The forcedirectio mount option forces I/O operations to be transferred directly from the user address space to disk, bypassing the kernel page cache. The option is particularly beneficial to applications that do their own caching, such as relational database systems (RDBMSs).

Note – The forcedirectio option applies to all files opened on the specified file system. This can have a negative impact on any applications also using the file system that do not cache their own data.

Recommended mount flag combinations for global UFS mounts

The following table provides recommendations for combinations of mount options to use for various classes of applications.

Table 8-3 Recommended mount options for global UFS mounts

Application class	Mount option flags
NFS	`global, logging`
Web servers	`global, logging`
RDBMS	`global, logging, forcedirectio`

Disk Device Group Considerations

One of Sun Cluster's major improvements over previous releases is the removal of the requirement that a data service must be physically attached to the physical storage storing its data. This means that in a three-or-more node cluster, the data service's resource groups might reside on a node that does not have physical access to the disk device group. File access is done over the interconnect between the data service's node and the node directly connected to the disk device group.

While this provides the benefit of faster failover, it could provide performance degradation for disk-intensive services such as Oracle or NFS.

In addition, during the cluster boot up or failover, a data service may attempt to start before global devices and cluster file systems are online. When this happens, manual intervention is required to reset the state of the resource groups.

To alleviate these problems, the Sun Cluster software includes a `SUNW.HAStorage` Resource Type.

The `SUNW.HAStorage` Resource Type

The resource type `SUNW.HAStorage` serves the following purposes:

- It coordinates the boot order by monitoring the global devices and cluster file systems and causing the `START` methods of the other resources in the same group with the `SUNW.HAStorage` resource to wait until the disk device resources become available.

- With the resource property `AffinityOn` set to `True`, it enforces collocation of resource groups and disk device groups on the same node, thus enhancing the performance of disk-intensive data services.

The `SUNW.HAStorage` Guidelines

On a two-node cluster, configuring `SUNW.HASstorage` is optional. To avoid additional administrative tasks, however, you could set up SUNW.HAStorage for all resource groups whose data service resources depend on global devices or cluster file systems.

You might also want to set the resource property `AffinityOn` to `True` for disk-intensive applications if performance is a concern. Be aware, however, that this increases failover times because the application does not start up on the new node until the device service has been switched over and is online on the new node.

In all cases, consult the documentation for the data service for specific recommendations.

Future PXFS improvements

The product team is researching other ways to improve the performance of PXFS. It is expected that other performance enhancements and/or tunables will be available for future releases of the software.

Sun Cluster Transport Tunables

The following sections describe private interconnect traffic and transport adapter properties.

Private Interconnect Traffic

There are three basic types of traffic that go across the private interconnect:

- Sun Cluster software-generated traffic (for example, infrastructure and CFS)
- Scalable services traffic
- Application-level traffic

SC 3.0 will stripe the first two kinds of traffic across all available interconnects. The third kind of traffic goes through the private interconnect using a special set of logical IP numbers that provide a fault-tolerant connection, but uses only one of the interconnects between each pair of nodes. The SC 3.0 software is written in such a way as to put traffic between different node pairs on different interconnects.

If most of the interconnect communications is CFS or Scalable Services traffic, increasing the number of interconnects in your cluster should increase the bandwidth available. If the traffic is application-level, you may have to configure the cluster with higher-bandwidth interconnects to increase performance.

Transport Adapter Properties

Adapter properties are set in the CCR.

The `dlpi_heartbeat_quantum` and `dlpi_heartbeat_timeout` are undocumented adapter properties. When a new `dlpi` cluster transport adapter is added to the cluster configuration, this property is automatically set to a default value for that adapter. Currently, all adapters use a default of 1000 and 10000 milliseconds.

Table 8-4 Transport Tunables

Tunable	Behavior
`dlpi_heartbeat_quantum`	This value is used to set the heartbeat interval time.
`dlpi_heartbeat_timeout`	This value is used to set the heartbeat timeout value.

This adapter property can be changed using `scconf`. But, its use is not publicly documented. It is currently considered for internal use only and should only be changed under the explicit direction of the Sun Cluster development team.

Example: Modifying a Transport Tunable

To change the `dlpi_heartbeat_timeout`, type the following at the command prompt:

```
# scconf -c -A name=name,node=node,dlpi_heartbeat_timeout=value
```

The /etc/system Tunables

The following are cluster-specific /etc/system tunables. This information is provided to you on a need-to-know basis. It is currently considered available for internal use only, and should only be changed under the explicit direction of the Sun Cluster development team.

The first two tunables control private interconnect behavior. The first global variable, pm_first_path_drop_panic, controls whether the local node should panic in the event of path failure to any node. The next variable pm_ignore_path_drop_mask provides a way to ignore path failures to a select set of nodes. This is to facilitate tests that intentionally bring down a cluster node. The bits in this mask get cleared as soon as the corresponding node establishes its first path with the local node. This mechanism has been provided to support path timeout debugging with non-debug bits.

Table 8-5 Private Interconnect Tunables

Tunable	Behavior
pm_first_path_drop_panic	When this value is set it will panic a node if it loses any path.
pm_ignore_path_drop_mask	This value specifies that the node will not panic if a path is lost to a node with its bit set.

The following parameters are used to tune TCP behavior on pathend and endpoint connections. The first variable, tcp_transport_ep_abort_threshold, is the time that TCP waits for an outstanding ack. If no ack is received within the specified timeframe, the connection is dropped. The next variable, tcp_transport_conn_abort_threshold, is the time TCP waits for a connection establishment to complete. If the connection could not be established in this time period, the connection attempt is abandoned.

Table 8-6 TCP Tunables

Tunable	Behavior
tcp_transport_ep_abort_threshold	Drop TCP connection if ack is pending. Kernel module is cl_dlpitrans. The default is 90 seconds.
tcp_transport_conn_abort_threshold	Abort TCP connection attempt if no ack is received. The default is 30 seconds.

Kernel /etc/system Tunables

Sun Cluster 3.0 systems have access to the same kernel tunables as standalone systems. In many cases, the same tunable rules apply in both a clustered and non-clustered system. For example, the same rules for tuning Oracle parameters apply to CFS as well as UFS/VxFS or raw partitions.

However, in some cases, applying the standalone server rules for a kernel tunable on a clustered system can result in a serious performance degradation. The following table lists the impact of tuning two commonly-tuned parameters.

Table 8-7 The Impact of Tuning Kernel (/etc/system) Variables

Kernel Variable Name	Description	Tuning Advised	Impact
ncsize	Number of directory name entries in the Directory Name Lookup Cache (DNLC)	NO	An increase in ncsize increases the number of CFS files in memory at any time. CFS files consume far more memory than files from local filesystems, so an increase in ncsize can seriously affect performance.
maxusers	Determines the size of various kernel tables such as the process table		ncsize uses the value of maxusers to determine its size: (ncsize = 17 * maxusers + 90). If maxusers must be increased, be sure to set ncsize to its non-tuned value.

If anyone asks, maxusers=Mbytes of RAM - 2 Mbytes used by kernel at boot time. The maximum automatic limit is 1024, but it can be increased manually up to 2048 in /etc/system.

Public Network Management (PNM) Tunables

Public Network Management has been EOL'ed for Sun Cluster 3.0. IP Multipathing will take its place in a future release.

There are five tunable parameters for PNM: `inactive_time`, `ping_timeout`, `repeat_rest`, `slow_network` and `warmup_time`. These parameters allow you to adjust the speed and fault detection behavior of the `pnmd`. The following table describes these values in detail.

Table 8-8 Public Network Management Tunables

Parameter	Description
`inactive_time`	Number of seconds between successive probes of the packet counters of the current active adapter. Default is 5.
`ping_timeout`	Time-out value in seconds for the `ALL_HOST_MULTICAST` and subnet broadcast pings. Default is 4.
`repeat_test`	Number of times to do the ping sequence before declaring that the active adapter is faulty and failover is triggered. Default is 3.
`slow_network`	Number of seconds waited after each ping sequence before checking packet counters for any change. Default is 2.
`warmup_time`	Number of seconds waited after failover to a backup adapter before resuming fault monitoring. This allows extra time for any slow driver or port initialization. Default is 0.

These parameters are set using the `/etc/cluster/pnmparams` configuration file. This file does NOT exist by default and is created only when you need to modify the default values.

Note – Settings in the `/etc/cluster/pnmparams` file apply to all NAFO groups on the node. Lines with a pound sign (#) are ignored. Other lines in the file must be of the format `variable=value`.

Example: Modifying the Public Network Management Tunables

To change the value of `inactive_time` to 3 and `repeat_test` to 5:

1. Become superuser on a cluster node.
2. Open the `/etc/cluster/pnmparams` file for editing. If it does not exist, create it.

```
# vi /etc/cluster/pnmparams
```

3. Add the following lines to the file or modify the lines if they exist:

```
inactive_time=3
repeat_test=5
```

4. Save the file and exit.

Note – Changes will take effect the next time the `pnmd` is restarted.

Module 8 Labs

The purpose of the following labs is to:

- Modify a load balancing policy.
- Modify public network management tunable parameters.

Make sure that the `test-cluster.cgi` script is available in the cgi-bin directory of each student's cluster.

Lab 1 - Modifying a Load Balancing Policy

1. Verify that the web-server is operational. Try connecting to the web server using the browser on the admin workstation. Connect using the URL:

   http://*web-server-name*/cgi-bin/test-cluster.cgi

   Substitute the name of your Web server for *web-server-name*.

   Repeatedly hit the refresh button on the browser. The `test-cluster.cgi` script will display the nodename that is servicing the request. It may take several iterations before the request is transferred to the other node. Notice that the requests are evenly split between the nodes.

2. Run `scstat` to look at the current configuration.

   # **scrgadm -pvv -g apache-rg**

3. Apply load balancing to the Apache data service.

scrgadm -c -j apache-res -y Load_balancing_weights=4@pnode1,1@pnode2

4. Verify that the configuration has been updated.

   # **scrgadm -pvv -g apache-rg |grep Load_balancing_weights**

   Test the configuration by accessing the test-cluster.cgi page on the web server. Once again repeatedly hit the refresh button on the browser. Notice that the requests are now being handled by the nodes based on the load balance weights.

5. Modify the data service load balance weights again, this time with a much bigger weight difference. This will make its effect more obvious.

```
# scrgadm -c -j apache-res -y Load_balancing_weights=9@pnode1,2@pnode2
```

6. Verify that the configuration has been updated.

```
# scrgadm -pvv -g apache-rg |grep Load_balancing_weights
```

7. Again, test the configuration by accessing the `test-cluster.cgi` page on the web server. Observe results.

8. Configure the Web server back to its default with the following command:

```
# scrgadm -c -j apache-res -y Load_balancing_weights=
```

9. Verify that the configuration has been updated.

```
# scrgadm -pvv -g apache-rg |grep Load_balancing_weights
```

Lab 2: Modifying the Public Network Management Tunable Parameters

1. To observe the behavior of the pnmd modifications, you must first run pnmd in the debug and tracefile mode. To turn debugging/trace mode on, open the `/etc/rc3.d/S24pnm` startup file in your favorite text editor.

2. Locate the following line:

```
$PMFADM -c $PNMD_TAG -n -1 -C 1 /usr/cluster/bin/pnmd
```

3. Change the line to:

```
$PMFADM -c $PNMD_TAG -n -1 -C 1 /usr/cluster/bin/pnmd -d -t /tmp/trac.out
```

4. Save the changes and close the file.

5. Restart pnmd to make the changes take effect.

```
# /etc/init.d/pnm stop
# /etc/init.d/pnm start
```

6. Verify that the pnmd is running the default PNM parameters by observing the tracefile output.

```
# cat /tmp/trac.out | grep param
1: parameters: inactive = 5; ping_tmout = 4; rep_test = 3; slow_nw = 2;
extra_arps = 4
```

7. Use a text editor to create the /etc/cluster/pnmparams file.

8. Modify the default parameters by adding the following lines to the file:

```
inactive_time=3
repeat_test=7
```

9. Verify /etc/cluster/pnmparams file settings.

```
# cat /etc/cluster/pnmparams
```

10. Restart the pnmd again.

```
# /etc/init.d/pnm stop
# /etc/init.d/pnm start
```

11. Verify that pnmd is running with the new parameters.

```
# cat /tmp/trac.out |grep param
```

12. Turn off the pnmd debug mode by restoring the /etc/rc3.d/S24pnm script to its original state.

13. To restore pnmd to its original default values, remove the /etc/cluster/pnmparams file and restart pnmd.

```
# rm /etc/cluster/pnmparams
# /etc/init.d/pnm start
# /etc/init.d/pnm stop
```

Appendix A

Cluster Boot Process

Table A-1 describes the high-level steps that take place during the cluster boot process during normal, reconfiguration, non-cluster and non-cluster reconfiguration boots.

These steps are described in greater detail in the following sections.

Table A-1 Cluster Boot Process

Normal Boot	Reconfiguration Boot	Non-Cluster Boot (`boot -x`)	Non-Cluster Reconfiguration Boot (`boot -rx`)
rcS			
Initialize DID using file interfaces	Initialize DID using file interfaces	Initialize DID using file interfaces	Initialize DID using file interfaces
Initialize the transport, ORB, CMM, HA framework, DCS	Initialize the transport, ORB, CMM, HA framework, DCS		
Initialize DID using CCR interfaces	Initialize DID using CCR interfaces		
Mount `/global/.devices/node@<nodeid>` as a UFS filesystem	Mount `/global/.devices/node@<nodeid>` as a UFS filesystem	Mount `/global/.devices/node@<nodeid>` as a UFS filesystem	Mount `/global/.devices/node@<nodeid>` as a UFS filesystem
Start up clexecd	Start up clexecd		
	Base Solaris runs devfsadm		Base Solaris runs devfsadm
	Discover new did devices if they exist		Discover new DID devices if they exist
rc2			
Get access to all local disks	Get access to all local disks		

Table A-1 Cluster Boot Process (Continued)

Normal Boot	Reconfiguration Boot	Non-Cluster Boot (`boot -x`)	Non-Cluster Reconfiguration Boot (`boot -rx`)
Import currently existing PXFS mounts	Import currently existing PXFS mounts		
Remount `/global/.devices/node@<nodeid>` as a PXFS filesystem	Remount `/global/.devices/node@<nodeid>` as a PXFS filesystem		
Mount all PXFS filesystems from `/etc/vfstab`	Mount all PXFS filesystems from `/etc/vfstab`		
Sync up global namespace with the rest of the cluster	Sync up global namespace with the rest of the cluster		
rc3			
Start up `rpc.pmfd`	Start up `rpc.pmfd`		
Start up `pnmd`	Start up `pnmd`		
Start up `rpc.fed`	Start up `rpc.fed`		
Start up `rgmd`	Start up `rgmd`		
Start up `scsymon`	Start up `scsymon`		
Start up Apache server for SunPlex Manager	Start up Apache server for SunPlex Manager	Start up Apache server for SunPlex Manager	Start up Apache server for SunPlex Manager

Kernel Boot Stage

The Kernel Boot Stage is as follows:

1. `main()` calls `startup()` which calls `mod_setup()`.
2. `mod_setup()` loads `misc/cl_bootstrap`.

If `cl_bootstrap` exists, its `_init()` routine sets `cluster_bootflags` to be either `CLUSTER_CONFIGURED` or `CLUSTER_CONFIGURED | CLUSTER_BOOTED` depending on whether the cluster was booted with the `-x` flag or without.

The `CLUSTER_CONFIGURED` configured flag is used by the instance number management code to determine if the `/devices/node@<node_id>` format should be used for the `/devices` directory.

The **rcS** Stage

Steps marked with an asterisk (*) use standard Solaris Operating Environment scripts.

1. * Run `S30rootusr.sh`.

   This script mounts /usr in read-only mode.

2. * Run `S40standardmounts.sh`

   This script mounts `/proc` and remounts `/` and `/usr` in read-write mode.

3. Run `S41bootcluster`.

   This script executes the following steps:

   a. Initializes `did` using regular file interfaces to read the CCR table. This is done so that the quorum algorithm can use `did` devices as quorum devices before the cluster has been formed.

   b. If the node is booting as a cluster, starts up the ORB, transport, CCR, and DCS (`modload misc/cl_comm` and `clconfig -c`). It also disables `TCP SACK`, a workaround for bug #4319441. The bug was fixed, but `TCP SACK` remains in the script.

      If installmode is disabled, it resets the node `quorum vote count` to its default (1).

If there is a failure, the script prints out an error message and halts the node.

c. Initializes `did` using CCR interfaces to read configuration.

Note – There is a possible bug here as the script does not distinguish between cluster and non-cluster boots before this step. Also, there is no handling for SCSI disks that return invalid SCSI IDs.

d. Enables `vxvm` volumes if the root disk has been encapsulated.

e. Mounts `/global/.devices/node@<node_id>` as a UFS filesystem.

f. If the node is booting as a cluster, starts up the `clexecd` mounter daemon.

If there is a failure, the script prints out an error message and halts the node.

4. * Run `S50devfsadm`.

This script runs `devfsadm` if this is a reconfiguration boot.

5. Run `S60initdid`.

This script exits if the node is not booting in cluster mode.

The script runs `scgdevs` if this is a reconfiguration boot.

6. Run `S68did_update_vfstab`.

The script exits if the node is not booting in cluster mode.

If the node is booting in cluster mode and it is the first boot after installation, it modifies the `vfstab` entry to use the did device corresponding to the `/dev/dsk` device for `/global/.devices/node@<nodeid>`.

This can be an area where problems might crop up because there is no handling of exceptions.

This script removes its link to `/etc/init.d/did_update_vfstab` after completion.

The **rc2** Stage

Steps marked with an asterisk (*) use standard Solaris Operating Environment scripts.

1. * Run `S01MOUNTFSYS`.

   The script mounts filesystems from `/etc/vfstab`.

2. * Run `S72inetsvc`.

   The script is the third phase of TCP/IP start up. Many of its functions depend upon the NIS/NIS+ maps. It also starts `inetd`.

3. Run `S75MOUNTGFSYS`.

   This script executes the following steps:

   a. Exits if not booted as a cluster.

   b. Runs the disk fencing program `/usr/cluster/lib/run_reserve` to get access to all connected disks.

   c. Unmounts `/global/node@<node_id>` as a local UFS filesystem.

   d. Runs `clconfig -g` to start up the DSM to start up replicas for active device services, and schedule switchbacks where necessary.

   e. Starts up the mount client to unmount stale versions of global mounts, join the current PXFS namespace, and start up replicas for currently active filesystems.

      If the import of global mounts fail, give the user a root shell to repair.

   f. Mounts `/global/node@<`*`node_id`*`>` as a PXFS filesystem.

      If the mount fails, give the user a root shell to repair.

   g. Mounts all other PXFS filesystems specified in `/etc/vfstab` using the following procedure for each filesystem:

      1. For each global mount in `/etc/vfstab`:

         Serialize the `fsck/mount` across the cluster (`clconfig -l(ock) special_file_name`).

         If this is not a UFS mount, `fsck`, mount and unlock.

         If this is a UFS mount, add the mount information to a list.

2. `fsck` all UFS filesystems in parallel (`fsck -o p`).

   If `fsck` fails, give user a root shell to repair.

   Mount all UFS filesystems globally in parallel (`mount -a`).

   Call `clconfig -u(nlock)` for all the global UFS mountpoints.

h. Run `S76gdevsync`.

The script exits if the node is not booting in cluster mode.

If the node is booting in cluster mode and it is a reconfiguration boot, the script runs `scgdevs`. This command regenerates the global namespace, discovers any new did devices, and runs `scgdevs` on all other cluster nodes if necessary.

The **rc3** Stage

1. Run `S17initpmf`.

   This script exits if the node is not booting in cluster mode.

   It starts the fault monitoring daemon, `rpc.pmfd`, if one is not already running.

2. Run `S23initfed`.

   This script exits if the node is not booting in cluster mode.

   It starts the "fork and exec" daemon, rpc.fed if one is not already running. This daemon handles the fork and exec calls from the rgm and UCMM.

3. Run `S24pnm`.

   This script exits if the node is not booting in cluster mode.

   It starts the public network management daemon, `pnmd`, if one is not already running.

   If there is a failure, the script exits. Otherwise it waits for up to 30 seconds for `pnmd` to start, using `pnmstat` as a poller.

   If `pnmd` hasn't started after 30 seconds, it prints an error and exits.

4. Run `S29initrgm`.

   This script exits if the node is not booting in cluster mode.

It starts the resource group management daemon, `rgmd`, if one is not already running.

5. Run `S30initscsymon`.

   This script exits if the node is not booting in cluster mode and starts the Sun Management Center cache daemon, `scsymon`, if `/usr/cluster/lib/scsymon/scsymon_srv` exists.

6. Run `S79spm`.

   This script starts the Apache server for SunPlex Manager.

7. Run `S80spminstall`.

   This script performs the post-cluster installation tasks for a cluster which has been installed by SunPlex Manager.

8. Finally, if the cause can still not be determined, follow the escalation procedures defined for your group.

Appendix B

Object Request Broker (ORB) Components

This section introduces the major ORB components / features:

- IDL compiler
- CORBA support
- Reference counting
- Invocation support
- ORB Subsystem Management
- Implementation objects
- Handlers
- Xdoors

IDL Compiler

IDL is a language-neutral Interface Definition Language that Sun acquired externally and extensively modified. The IDL compiler translates IDL specifications stored in an `.idl` file into C++ language constructs that become part of the ORB when the ORB is built. There is currently very little documentation of any form on the compiler. Specifically, the compiler generates the following:

- C++ header file
- Stub for client code
- Skel for server code
- Schema file containing data about the class and data types
- Type-specific support functions

The ORB maintains separate lists of IDL-defined object and exception descriptors. The system loads this information during ORB initialization using code generated by the IDL compiler.

CORBA (Common Object Request Brokerage Architecture) Support

CORBA is an industry-standard distributed object model. The CORBA cluster component provides the implementation for CORBA predetermined types, such as strings. The Sun Cluster software only uses a small portion of the CORBA model.

The CORBA component manages the references to an object using "smart" and "dumb" pointers. Dumb pointers do not handle reference counts automatically, and the programmer must specify the functions to handle them. Smart pointers can handle much of the reference counting automatically.

Reference Counting

The system keeps track of the number of referenced objects. It adjusts the reference count any time an action occurs which affects the number of references:

```
get object reference/release object reference
marshal/unmarshal an object reference
```

"Marshal" is a term used to represent the process that writes an in-memory structure of objects to a data stream. "Unmarshal" refers to the process that takes the data stream and reads it into memory.

The reference counting mechanism is used by the ORB to determine when it is safe to delete an object. It can also be used to determine a node failure.

Invocation Support

All ORB messages marshal outbound message data into a "sendstream" and unmarshal inbound message data from a "recstream". Both sendstream and recstream share a similar structure consisting of three principal parts:

- "MarshalStream" MainBuf – Contains the various message headers followed by ordinary marshalled data
- "MarshalStream" XdoorTab (Optional) – Contains marshalled object references
- "Region" (Optional) – Contains offline, typically large, data for which the ORB attempts to reduce data copying

The system supports IDL invocations with "service" class objects that are typically specialized for one-way and two-way invocations. The "service" classes are able to perform the appropriate marshal and unmarshal time operations.

During a node reconfiguration, the ORB rebuilds the object reference counts.

ORB Subsystem Management

The ORB has a component that runs in each domain. In ORB language, a domain is a process that is local or remote. This means each process contains a kernel level component and an ORB component.

The ORB must be explicitly initialized and is modloaded into the kernel. The kernel ORB represents every node that is currently in the cluster using an `ID_node` object. As a node joins the cluster, the ORB also assigns an incarnation number to the object. The ORB uses the combination of the node number and incarnation number to distinguish between orphan data and current data.

When a node fails, the ORB updates its internal data structures. The CMM provides the inter-node synchronization support.

Implementation Objects

Server objects are implemented as C++ templates. The developer takes the basic definition and fills in the functions as needed. Examples include:

- `McServerof` – Reference counted object
- `McNoref` – Non-reference counted object
- `mc_checkpoint` – HA checkpoint object
- `mc_replica_of` – HA replicated object
- `Anchor` – Proxy to well-defined server with fixed address used for solving startup problems

Handlers

Handlers provide supporting methods (functions) for objects manipulated by the ORB. Handler responsibilities include:

- Support invocation operations
- Marshal object representation
- Provide type manipulation
- Support reference operation
- Support unreference operation (garbage collection)

Xdoors

Xdoors are "extended" doors. Doors provide a facility for processes to issue procedure calls to functions in other processes running on the same system.

Xdoors extend the facility to processes on other nodes.

There are a variety of xdoor implementation classes to support different object requirements:

- `solaris_xdoor` – Supports object addresses in user space
- `rxdoor` – Supports features common to kernel addresses and is used for inter-node communications

- `standard_xdoor` – Specialized rxdoor that supports reference counting
- `simple_xdoor` – Specialized rxdoor used to solve cyclical startup issues
- `hxdoor` – Represents an address for a replicated object

Message Transport

The ORB message transport supports all messages exchanged between domains.

Additional References

For copies of TOI slides and links to white papers that discuss the ORB in greater detail, consult the Sun Cluster 3.0 Transfer of Information (TOI) page at: `http://galileo.eng/SC30TOI/index.html`

Appendix C

Sun Cluster 3.0 Diagnostic Tool Kit Selected Man Pages

NAME

ccradm – CCR table files administration command

SYNOPSIS

/usr/cluster/lib/sc/ccradm –i *ccr_table_file* [**–o**]
/usr/cluster/lib/sc/ccradm –r *ccr_table_file* **–f** *input_file*

DESCRIPTION

ccradm is an emergency command line interface designed only to be used when a Cluster Configuration Repository (CCR) table is corrupted. CCR tables are represented in the file system by ASCII text files, in the directory **/etc/cluster/ccr**. **ccradm** is used to make the data and checksum in the CCR table files consistent. Each file starts with a generation number, *ccr_gennum*, and a checksum, *ccr_checksum*. *ccr_gennum* indicates the current generation number of the CCR table file, and is used to keep track of the most current version of the CCR table file among the different nodes in the cluster. *ccr_checksum* indicates the checksum of the CCR table contents, and provides a consistency check of the data in the table.

ccr_table_file is the name of the file representing the CCR table on the local node.

If a CCR table file is manually edited as part of some emergency repair procedure, then the checksum must be re-computed to make the data consistent.

OPTIONS

–i

This option can be used only in non-cluster mode. It re-computes the checksum and updates *ccr_table_file* with the new checksum.
When used without the **–o** option, it sets the generation number to INIT_VERSION. A generation number of INIT_VERSION means that the CCR table file is valid only while recovery is being performed as part of the node boot process in cluster mode. A prerequisite for successful recovery is one of the following: either 1) one of the other nodes in the cluster must have the override version set for the CCR table file, or 2) at least one of the other nodes must have a valid copy of the CCR table file. A CCR table file is valid if it has a valid checksum and its generation number is greater than or equal to zero.
If *ccr_table_file* has a generation number of INIT_VERSION on all nodes, then the CCR table will remain invalid after recovery has completed. Therefore, do not use the **–i** option without the **–o** option on a CCR table file on all nodes in the cluster.
The CCR table file is ignored when the node joins a cluster where this CCR table has already been recovered.

–o The override option is used with the **–i** option. This option can be used only in non-cluster mode. It sets the generation number to OVRD_VERSION. This option is used to designate one CCR table file to be the master copy. This version of the CCR table file will override other versions of the file that are on the remaining nodes during recovery. If a CCR table file has a generation number of OVRD_VERSION on more than one node,

then only one of the files will be selected and a warning message will be printed on the console of one of the nodes. After recovery the table's generation number will be reset to 0.

–r and **–f**
These options can be used only in cluster mode. Do not use these options unless explicitly directed by a Sun Cluster development engineer. These options must always be used together. Contents of the file specified with the **–f** option will replace the contents of the table file specified with the **–r** option. The -r *ccr_table_file* refers to the existing invalid table in the CCR and the -f *input_file* refers to the one it should be replaced with. The existing CCR table file to be restored with the -r option must exist under the directory /etc/cluster/ccr. The checksum will be recomputed and the generation number will be reset to 0. If a table is invalid even after recovery then it can be restored by using these options.

USAGE

ccradm should be used only for emergency CCR table file repair.
–i and **–o** options can be used only in non-cluster mode.
–r and **–f** options can be used only in cluster mode.

REPAIR PROCEDURE

Perform these steps to repair a corrupted RGM CCR table only when directed as part of an emergency repair procedure.

Reboot all nodes in non-cluster mode.

Edit the file on all nodes to contain the correct data. The file must be identical on all nodes.

Because the file is identical on all nodes, it also can be designated as the override version on all nodes. Recompute the checksum and designate this CCR table file to be the override version by running the following command on all nodes. *file* is the name of the CCR table.

/usr/cluster/lib/sc/ccradm –i /etc/cluster/ccr/*file* –o

Reboot all nodes in cluster mode.

EXAMPLES

In non-cluster mode, recompute the checksum and set the generation number to INIT_VERSION for CCR table file *www* on the local node:

ccradm –i /etc/cluster/ccr/www

In non-cluster mode, recompute the checksum and set the generation number to OVRD_VERSION for CCR table file *xxx* on the local node:

ccradm –i /etc/cluster/ccr/xxx –o

In cluster mode, replace the CCR table *yyy* with the contents of its backup version in the file **/etc/cluster/ccr/yyy.bak**:

ccradm –r /etc/cluster/ccr/yyy –f /etc/cluster/ccr/yyy.bak

Here is the generic procedure for fixing corrupted CCR tables. In this example the table name is **directory**:

Reboot all nodes in non-cluster mode.

Edit the file **/etc/cluster/ccr/directory** on all nodes to contain the correct data. The file must be identical on all nodes.

On all nodes, recompute the checksum and designate this CCR table file to be the override version.

phys-schost-1# /usr/cluster/lib/sc/ccradm -i /etc/cluster/ccr/directory -o
phys-schost-2# /usr/cluster/lib/sc/ccradm -i /etc/cluster/ccr/directory -o

```
phys-schost-3# /usr/cluster/lib/sc/ccradm -i /etc/cluster/ccr/directory -o
phys-schost-4# /usr/cluster/lib/sc/ccradm -i /etc/cluster/ccr/directory -o
```

Reboot all nodes in cluster mode.

EXIT STATUS

The following exit values are returned:

0 No errors occurred. Successfully updated or restored the CCR table file

>0 Error occurred.

WARNING

Do not run **ccradm –i** without the **–o** option on a CCR table file on all nodes.

SEE ALSO

chkinfr(1M)

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

NAME

`chkinfr` – validate infrastructure file contents

SYNOPSIS

/usr/cluster/lib/sc/chkinfr [**–F**] [**–i** *filename*] [**–p** *directory*]

DESCRIPTION

chkinfr validates the contents of the **/etc/cluster/ccr/infrastructure** file, which contains information about the configuration of the cluster. This file contains information about the nodes in the cluster, adapters in each node, switches / blackboxes used to connect the nodes (private interconnect), cables that are connected between adapters or between an adapter and a switch / blackbox, and quorum devices. It also contains other information such as name of the cluster and the number of quorum votes given to each node. See **ccr_infrastructure**(4) for more details about the contents of the infrastructure file.

Since the infrastructure information is stored in a file in the file system, it is possible for it to get corrupted or truncated or removed without the knowledge of cluster software. Also the file can be edited manually by the cluster administrator to correct or add information as part of emergency repair procedures. Thus, **chkinfr** is necessary to make sure that the file contents are correct and the system will boot successfully and function as expected.

chkinfr is invoked at boot time before the cluster software is loaded by the rc script **/etc/rcS.d/S41bootcluster.sh**. Any errors detected are displayed on the console and also stored in the **/etc/cluster/chkinfr.err** file. The system is halted if any errors are detected. If this happens, you should boot the system in non-cluster mode and repair the file.

chkinfr checks to see if the information about nodes, adapters, switches, cables, and quorum devices is correct in the infrastructure file. It uses the `.clpl` files in the **/etc/cluster/clpl** directory to validate the supported adapters (for example, `hme`,`qfe`), supported transport types (e.g. `dlpi`,`rsm`) for each type of adapter, and supported blackbox types (e.g. switch).

OPTIONS

–F
Force the command return value to 0. Can be used to continue booting if **chkinfr** in the rc script reports errors.

–i*file*
validates *file* instead of the default **/etc/cluster/ccr/infrastructure**.

–p*directory*
searches for the `.clpl` files in *directory* instead of the default **/etc/cluster/clpl**.

USAGE

chkinfr can be used in cluster mode and non-cluster mode.

EXIT STATUS

The following exit values are returned:

0 No errors were detected, or errors were detected but the **–F** flag was used.

>0 Error(s) were detected.

FILES

/etc/cluster/ccr/infrastructure
The clustering infrastructure file

/etc/cluster/clpl
The `.clpl` files directory

/etc/cluster/chkinfr.err
Error output from **chkinfr**

/etc/rcS.d/S41bootcluster.sh
rc script that runs **chkinfr**

NOTES

Here is the list of possible cases where **chkinfr** reports errors.

The cluster wide check reports errors in the following cases:
- it cannot open `.clpl` files or `.clpl` files are corrupted
- there are no nodes in the cluster
- the node name is missing or duplicated
- there are no ENABLED nodes in the cluster
- the blackbox name is missing or duplicated
- the `quorum_vote` and `quorum_resv_key` are not in at least one node's properties
- the connectivity matrix shows that one of the nodes is not reachable from another node

Each node check reports errors in the following cases:
- node name or state is missing
- missing or duplicate adapter name
- there are no ENABLED adapters in the node
- there is no private hostname property

Each blackbox/switch check can result in errors in the following cases:
- blackbox name or state is missing
- missing or invalid type
- the same blackbox port is used more than once
- blackbox is enabled and there are no ports
- blackbox is enabled and port name is missing
- blackbox is enabled and there are no ports

Each cable check can result in errors in the following cases:
- cable state is missing
- endpoint is not valid (For example, the endpoint does not exist in the infrastructure file)
- cable endpoint is not an adapter or a blackbox
- both the endpoints are adapters and they are of different types
  (e.g. one is hme and the other is sci)

Each adapter check can result in errors in the following cases:
- adapter name or state is missing
- invalid IP address (e.g. 124.367.18.50)
- duplicate IP address
- invalid netmask
- one or more of the ip_address, transport_type, device_name, device_instance,
  or netmask properties are missing
- device name is not supported (using .clpl files)
- adapter_id property is missing when the device name is sci
- device name is sci and trtype is dlpi and dlpi_device_name is missing
- ports are missing and adapter is enabled
- more than one port is mentioned for an adapter

Each quorum device check can result in errors in the following cases:
- the quorum device id is out of range
- the gdevname is missing
- the vote count property is missing

SEE ALSO

ccr_infrastructure(4)

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

NAME

ccr_infrastructure, infrastructure - CCR cluster infrastructure table

SYNOPSIS

/etc/cluster/ccr/infrastructure

DESCRIPTION

The CCR cluster infrastructure table is a CCR table which contains most of the primary configuration data and properties for the low level cluster infrastructure, transport, and quorum devices.

Each line in the table has an entry in the form of a key-value pair. In some cases, the keyword itself includes variable data, such as an identifier, or ID. But, with the exception of node IDs, most IDs embedded within keywords have little or no use outside of defining a branch within the CCR infrastructure tree itself.

The infrastructure table hierarchy is that of a simple tree structure. And, each ID is used to group together branches of that tree.

This table, of course, follows all of the basic syntax rules required of any CCR table. It is stored under /etc/cluster/ccr as a regular ASCII file, with a CCR directory entry. And, it includes the standard **ccr_gennum** and **ccr_checksum** keywords.

As with all CCR table files, the manual editing of this file is strongly discouraged, except under extreme emergency conditions and under the direct instruction from service support. Editing should only occur when all nodes are booted in non-cluster mode and must be performed using the approved procedures for repairing a CCR table (see **ccradm**(1M)).

FORMAT

This section describes the keywords found in the CCR cluster infrastructure table. If the cluster is in a healthy state, most of these data items can be changed using either **scconf**(1M) or **scsetup**(1M).

cluster.name*clustername*

The *clustername* is the name of the cluster. It may be any alphanumeric string of up to 256 characters.

The *clustername* is set by **scinstall**(1M) at the time that the first node is installed in the cluster. And, it can be used again by **scinstall**(1M) to verify that subsequently installed nodes are added to the right cluster. Otherwise, the **Sun Cluster 3.0** software does not use the *clustername* for any purpose other than display. However, additional dependencies may evolve in the future.

If a *clustername* is not given at install time, the default is to set *clustername* to be the name of the first node used to establish the cluster. Both **scsetup**(1M) and **scconf**(1M) can be used later to change the *clustername*.

cluster.state*state*

The cluster *state* must always be set to **enabled**.

cluster.properties.cluster_id*clusterid*

The *clusterid* is a randomly generated 32-bit number represented in this table in hexadecimal format. It is preferred, but not required, for each cluster within the enterprise to have a unique *clusterid*. Currently, **scinstall**(1M) uses the time-of-day clock to generate the *clusterid* at the time that the first node is installed in the cluster.

The *clusterid* is used in forming individual quorum reservation keys.

The *clusterid* cannot be changed by standard procedures or commands.

cluster.properties.installmode*state*

The cluster installmode *state* may be set to either **enabled** or **disabled**. The installmode *state* is usually only **enabled** when the cluster is first installed.

At cluster install time, the first node to be installed is installed with a quorum vote count of one, and all other nodes are installed with a vote count of zero. Normally, when a node boots in cluster mode, its quorum vote count is always set to one. However, if installmode is **enabled**, the vote count is left unchanged.

When **scsetup**(1M) is run, it will detect whether or not installmode is **enabled**. And, if it is set, it will allow the user to configure any required shared quorum devices, reset the vote count of all the nodes to their default of one, and, finally, disable installmode.

cluster.properties.private_net_number*netnumber*
The *netnumber* is the network address (**networks**(1M)) assigned to the private network. This, along with the **private_netmask** property, is used to generate individual transport adapter **netmask** and **ip_address** properties.

The *netnumber* is specified in standard dot notation. And, the last two octets of the *netnumber* must always be zero. The default number is 172.16.0.0.

This private *netnumber* is set by **scinstall**(1M) at the time that the first node is installed into the cluster. And, it cannot be changed later by standard procedures or commands.

cluster.properties.private_netmask*netmask*
The *netmask* is used to define the network portion of the given **private_net_number**.

The *netmask* is specified in standard dot notation. The last two octets of this *netmask* must always be zero, and there may not be any holes in the mask. The default mask is 255.255.0.0.

This private *netmask* is set by **scinstall**(1M) at the time that the first node is installed into the cluster. And, it cannot be changed later by standard procedures or commands.

cluster.nodes.<*nodeid*>**.name***nodename*
The cluster *nodename* for any given *nodeid* should always be the same as its Solaris nodename, as expressed by uname -n (**uname**(1M)).

Each *nodename* is set by **scinstall**(1M). And, neither the *nodeid* nor the *nodename* can be changed later by standard procedures or commands.

cluster.nodes.<*nodeid*>**.state***state*
The *state* of a particular *nodeid* is either **enabled** or **disabled**.

A node will typically only be in the **disabled** state for a very brief period during its installation. Once the first **cable** is established for a new cluster node, the *state* of the node is immediately **enabled**. And, there is no command support for switching the *state* back to **disabled**.

cluster.nodes.<*nodeid*>**.properties.private_hostname***phostname*
The private hostname of a node is the name by which a node can be contacted over the private network. Each private hostname should be unique.

At install time, each node is assigned to a default private hostname of **clusternode**<*nodeid*>**-priv.** And, private hostnames can be changed later using either **scsetup**(1M) or **scconf**(1M).

Note that private hostnames should never be stored in /etc/inet/hosts or other **hosts**(4) databases.

cluster.nodes.<*nodeid*>**.properties.quorum_vote***vcount*
If a node is in quorum maintenance mode, it will have a quorum vote count of zero. Otherwise, its vote count should always be set to one. And, unless cluster installmode is set, the vote count of a node is always reset to one when it boots into the cluster.

scconf(1M) can be used to set and clear the quorum maintenance state of a node. However, there are no supported procedures for changing the vote count of a node to anything other than zero or one.

cluster.nodes.<*nodeid*>**.properties.quorum_resv_key***rkey*
Each node in the cluster is automatically assigned to a quorum reservation key at install time. This is an eight byte value expressed in *rkey* as a single hexadecimal number. The key is generated using the *clusterid* and *nodeid*.

The quorum reservation keys cannot be changed by standard procedures or commands.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.name***adaptername*
Each node in the cluster will typically be configured with two or more cluster transport adapters. Transport adapter names are given here using the same style that physical *interface* names are given to **ifconfig**(1M).

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.state***state*
The state of an individual adapter is either **enabled** or **disabled**.

scconf(1M) can be used to change the state of an adapter.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.properties.device_name***dname*
The **device_name** adapter property is always set to the device name portion of the adapter name. It is automatically set at the time that a new transport adapter is added to the cluster configuration.

This adapter property cannot be changed with **scconf**(1M) or other standard procedures or commands.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.properties.device_instance***dinst*
The **device_instance** adapter property is always set to the device instance portion of the adapter name. It is automatically set at the time that a new transport adapter is added to the cluster configuraion.

This adapter property cannot be changed with **scconf**(1M) or other standard procedures or commands.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.properties.transport_type***trtype*
The only currently supported transport type is **dlpi**. However, **rsm** is expected to be supported in the near future for **sci** adapters. Transport types should not be mixed on the same cluster.

The transport type is set at the time that a new transport adapter is added to the cluster configuraion. And, if not explicitly set, it will be set to the default, **dlpi**. Once it is set, this adapter property cannot be changed with **scconf**(1M) or other standard procedures or commands.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.properties.netmask***netmask*
The per-adapter **netmask** property is internally generated by the transport software at the time that a new adapter is added to the cluster configuration.

Users should not make any assumptions about how this property is used by the clustering software and should have no reason to even look up the setting of this property.

This adapter property cannot be changed with **scconf**(1M) or other standard procedures or commands.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.properties.ip_address***ip_address*
The per-adapter **ip_address** property is internally generated by the transport software at the time that a new adapter is added to the cluster configuration.

Users should not make any assumptions about how this property is used by the clustering software and should have no reason to even look up the setting of this property.

This adapter property cannot be changed with **scconf**(1M) or other standard procedures or commands.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.properties.dlpi_heartbeat_quantum***quantum*
The **dlpi_heartbeat_quantum** is an undocumented adapter property. When a new **dlpi** cluster transport adapter is added to the cluster configuration, this property is automatically set to a default value for that adapter. Currently, all adapters use a default of 2000 milliseconds.

This adapter property can be changed using **scconf**(1M). But, its use is not publicly documented. It is currently considered for internal use only and should only be changed under the explicit direction of the Sun Cluster development team.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.properties.dlpi_heartbeat_timeout***tout*
The **dlpi_heartbeat_timeout** is an undocumented adapter property. When a new **dlpi** cluster transport adapter is added to the cluster

configuration, this property is automatically set to a default value for that adapter. Currently, all adapters use a default of 10000 milliseconds.

This adapter property can be changed using **scconf**(1M). But, its use is not publicly documented. It is currently considered for internal use only and should only be changed under the explicit direction of the Sun Cluster development team.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.properties.rsm_heartbeat_quantum***quantum***

The **rsm_heartbeat_quantum** is an undocumented adapter property. When a new **rsm** cluster transport adapter is added to the cluster configuration, this property is automatically set to a default value for that adapter. Currently, all adapters use a default of 2000 milliseconds.

This adapter property can be changed using **scconf**(1M). But, its use is not publicly documented. It is currently considered for internal use only and should only be changed under the explicit direction of the Sun Cluster development team.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.properties.rsm_heartbeat_timeout***tout*

The **rsm_heartbeat_timeout** is an undocumented adapter property. When a new **rsm** cluster transport adapter is added to the cluster configuration, this property is automatically set to a default value for that adapter. Currently, all adapters use a default of 1000 milliseconds.

This adapter property can be changed using **scconf**(1M). But, its use is not publicly documented. It is currently considered for internal use only and should only be changed under the explicit direction of the Sun Cluster development team.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.properties.bandwidth***bwidth*

The **bandwidth** is property an undocumented adapter property. When a new cluster transport adapter is added to the cluster configuration, this property is automatically set to a default value for that adapter. The current defaults are as follows:

ge70
hme10
qfe10
sci30

Values are in units of MB/sec.

This adapter property can be changed using **scconf**(1M). But, its use is not publicly documented. It is currently considered for internal use only and should only be changed under the explicit direction of the Sun Cluster development team.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.properties.nw_bandwidth***nw_bwidth*

The **nw_bandwidth** is property an undocumented adapter property. When a new cluster transport adapter is added to the cluster configuration, this property is automatically set to a default value for that adapter. All adapters currently have the same default value of 80%.

This adapter property can be changed using **scconf**(1M). But, its use is not publicly documented. It is currently considered for internal use only and should only be changed under the explicit direction of the Sun Cluster development team.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.ports.**<*id*>**.name***portname*

Whenever a cable is configured for a particular adapter, a new port is added. And, conversely, the port is automatically removed whenever the cable is removed. All currently supported adapters support only a single port. And, so, the adapter port *id* is always 1.

The default *portname* for an adapter is always 0. And, while both **scconf**(1M) and the non-interactive form of **scinstall**(1M) allow the user to specify a different *portname* with the cable suboptions, the default is usually taken. The adapter *portname* has no special meaning for any of the currently supported adapters. However, the command set disallows the inclusion of the characters '@', ':', and ',' in the name.

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.ports**<*id*>**.state***state*

The *state* of an adapter port is either **enabled** or **disabled**.

Whenever a cable is enabled using either **scinstall**(1M), **scsetup**(1M), or **scconf**(1M), both of its ports, and their parent objects (adapters or junctions/blackboxes) are also enabled. However, when a cable is disabled, only the state of the cable itself is changed. The command set does not provide for the disabling of an individual port once it has been enabled.

cluster.blackboxes.<*id*>**.name***bbname*

Blackbox is an obsolete term for cluster transport junction. However, the old term is still used in the CCR infrastructure table.

Each junction name must be unique within the cluster. But, none of the currently support junction types have any special requirements of the name.

A junction is named whenever it is added to the cluster configuration using **scinstall**(1M), **scsetup**(1M), or **scconf**(1M). The default name used by the command set is switch<*N*>. The command set does not support changing the name once it has been established.

cluster.blackboxes.<*id*>**.state***state*

Again, blackbox is an obsolete term for cluster transport junction. However, the old term is still used in the CCR infrastructure table.

The *state* of a particular junction is either **enabled** or **disabled**.

Whenever a cable is enabled using either **scinstall**(1M), **scsetup**(1M), or **scconf**(1M), both of its ports, and their parent objects (adapters or junctions), are also enabled. The **scconf (1M)** command also provides for expicit enabling and disabling of an entire junction under certain conditions.

cluster.blackboxes.<*id*>**.properties.type***bbtype*

Again, blackbox is an obsolete term for cluster transport junction. However, the old term is still used in the CCR infrastructure table.

The only junction type currently supported is *bbtype* **switch**. However the non-interactive form of **scinstall**(1M) and the **scconf**(1M) command do support options for expansion to other junction types. But, the command set does not support changing the type once it has been established.

cluster.blackboxes.<*id*>**.ports.**<*id*>**.name***portname*

Again, blackbox is an obsolete term for cluster transport junction. However, the old term is still used in the CCR infrastructure table.

Whenever a cable is configured to a port on a particular junction, a new port object is added. And, conversely, the port is automatically removed whenever the cable is removed.

scinstall(1M), **scsetup**(1M), and **scconf**(1M) all provide for *portname* specification as part of adding cables. The default junction *portname* used by the command set is the same as the *nodeid* number found at the other end of the cable. However, some junction types, such as the **sci** switch, do have special naming requirements. But, the command set does not support changing the *portname* once it has been established at cable-add time.

cluster.blackboxes.<*id*>**.ports.**<*id*>**.state***state*

Again, blackbox is an obsolete term for cluster transport junction. However, the old term is still used in the CCR infrastructure table.

The *state* of an individual junction port is either **enabled** or **disabled**.

Whenever a cable is enabled using either **scinstall**(1M), **scsetup**(1M), or **scconf**(1M), both of its ports, and their parent objects (adapters or junctions/blackboxes), are also enabled. However, when a cable is disabled, only the state of the cable itself is changed. The command set does not provide for the disabling of an individual port once it has been enabled.

cluster.cables.<*id*>**.properties.end1***endpoint1*
cluster.cables.<*id*>**.properties.end2***endpoint2*

Each cluster transport cable has two endpoints, *endpoint1* and *endpoint2*.

At least one of the two endpoints must be an adapter port endpoint. And, the other endpoint is either that of a port on another adapter or that of a port on a cluster transport junction.

An adapter port *endpoint* takes the following form:

cluster.nodes.<*nodeid*>**.adapters.**<*id*>**.ports.**<*id*>

And, a junction endpoint looks like this:

cluster.blackboxes.<*id*>**.ports.**<*id*>

scinstall(1M), **scsetup**(1M), and **scconf**(1M) can all be used to add cables to the cluster. But, the command set does not support changing endpoint data for already established cables. Cable definitions may be added to or removed from the configuration, but they cannot be changed in place.

cluster.cables.<*id*>**.state***state*

The state of a particular cable is either **enabled** or **disabled**.

And, whenever a cable is enabled using either **scinstall**(1M), **scsetup**(1M), or **scconf**(1M), both of its ports, and their parent objects (adapters or junctions), are also enabled. However, when a cable is disabled, only the state of the cable itself is changed. Both **scsetup**(1M) and **scconf**(1M) support changing the state of a cable to either one of the two states, conditions permitting.

cluster.quorum_devices.<*id*>**.name***name*

In the current release, some number of shared quorum disk devices may be added using either **scsetup**(1M) or **scconf**(1M).

The minimum requirement recognized and enforced by the command set is that all two-node clusters be configured with at least one dual-ported shared quorum disk device. However, the **Sun Cluster 3.0 AnswerBook** documentation should be carefully inspected for additional requirements and recommendations which might need to be met before a cluster can be considered supportable.

The quorum device *name* is always set to the relative name of a DID, or global, disk. In other words, the quorum device *name* will always have the format of **d**<*N*>. Both an absolute path prefix and slice number suffix will always be missing.

Both **scsetup**(1M) and **scconf**(1M) can be used to add and remove shared quorum devices to and from the cluster configuration.

cluster.quorum_devices.<*id*>**.state***state*

When a shared quorum device is in quorum maintenance state, its *state* is set to **disabled**, and its quorum vote count is set to zero.

Note that the meaning of and operation against this field may change in the **beta2** release of **SunCluster3.0**.

scconf(1M) can be used to set or re-set the quorum maintenance state of a shared quorum device.

cluster.quorum_devices.<*id*>**.properties.votecount***vcount*

When a shared quorum device is in quorum maintenance state, its *state* is set to **disabled**, and its quorum vote count is set to zero. But, if the device is not in maintenance state, its *state* is set to **enabled**, and its vote count, or *vcount*, is set to zero or <*N*>-1, whichever is greater. <*N*> is the number of enabled paths (see the **path_**<*nodeid*> property, below).

scconf(1M) can be used to set or re-set the quorum maintenance state of a shared quorum device.

cluster.quorum_devices.<*id*>**.properties.gdevname***gdevname*

The *gdevname* for a shared quorum device directly names the absolute path of the DID disk name to use for accessing this device as a shared quorum device. For example, if the **quorum_devices** *name* is **d4**, the *gdevname* property for the device should likely be set to **/dev/did/rdsk/d4s2**.

When either **scsetup**(1M) or **scconf**(1M) are used to add a new shared quorum disk device to the configuration, only the relative device *name* is required. The *gdevname* is automatically set.

cluster.quorum_devices.<*id*>**.properties.path_**<*nodeid*>*state*

A **path_**<*nodeid*> entry should exist for each configured node-to-shared-quorum-device. If the node is in quorum maintenance state, the *state* value for this entry should be set to **disabled**. Otherwise, it should be set to **enabled**.

Note that the meaning of and operation against this field may change in the **beta2** release of **SunCluster3.0**.

EXAMPLE

The following is an example of an infrastructure CCR table:

```
ccr_gennum6
ccr_checksumA43DCF0929777EF8FFC6FDC1B629CAD0
cluster.namesuncluster
cluster.stateenabled
cluster.properties.cluster_id0x377274A2
cluster.properties.installmodedisabled
cluster.properties.private_net_number172.16.0.0
cluster.properties.private_netmask255.255.0.0
cluster.nodes.1.namered
cluster.nodes.1.stateenabled
cluster.nodes.1.properties.private_hostnameclusternode1-priv
cluster.nodes.1.properties.quorum_vote1
cluster.nodes.1.properties.quorum_resv_key0x377274A200000001
cluster.nodes.1.adapters.1.namehme1
cluster.nodes.1.adapters.1.stateenabled
cluster.nodes.1.adapters.1.properties.device_namehme
cluster.nodes.1.adapters.1.properties.device_instance1
cluster.nodes.1.adapters.1.properties.transport_typedlpi
cluster.nodes.1.adapters.1.properties.netmask255.255.255.128
cluster.nodes.1.adapters.1.properties.ip_address172.16.0.129
cluster.nodes.1.adapters.1.properties.dlpi_heartbeat_quantum2000
cluster.nodes.1.adapters.1.properties.dlpi_heartbeat_timeout10000
cluster.nodes.1.adapters.1.ports.1.name0
cluster.nodes.1.adapters.1.ports.1.stateenabled
cluster.nodes.1.adapters.2.namehme2
cluster.nodes.1.adapters.2.stateenabled
cluster.nodes.1.adapters.2.properties.device_namehme
cluster.nodes.1.adapters.2.properties.device_instance2
cluster.nodes.1.adapters.2.properties.transport_typedlpi
cluster.nodes.1.adapters.2.properties.netmask255.255.255.128
cluster.nodes.1.adapters.2.properties.ip_address172.16.1.1
cluster.nodes.1.adapters.2.properties.dlpi_heartbeat_quantum2000
cluster.nodes.1.adapters.2.properties.dlpi_heartbeat_timeout10000
cluster.nodes.1.adapters.2.ports.1.name0
cluster.nodes.1.adapters.2.ports.1.stateenabled
cluster.nodes.2.namegreen
cluster.nodes.2.stateenabled
cluster.nodes.2.properties.quorum_vote1
cluster.nodes.2.properties.quorum_resv_key0x377274A200000002
cluster.nodes.2.properties.private_hostnameclusternode2-priv
cluster.nodes.2.adapters.1.namehme1
cluster.nodes.2.adapters.1.stateenabled
cluster.nodes.2.adapters.1.properties.device_namehme
cluster.nodes.2.adapters.1.properties.device_instance1
cluster.nodes.2.adapters.1.properties.transport_typedlpi
cluster.nodes.2.adapters.1.properties.netmask255.255.255.128
cluster.nodes.2.adapters.1.properties.ip_address172.16.0.130
cluster.nodes.2.adapters.1.properties.dlpi_heartbeat_quantum2000
cluster.nodes.2.adapters.1.properties.dlpi_heartbeat_timeout10000
```

```
cluster.nodes.2.adapters.1.namehme2
cluster.nodes.2.adapters.1.ports.1.name0
cluster.nodes.2.adapters.1.ports.1.stateenabled
cluster.nodes.2.adapters.2.namehme2
cluster.nodes.2.adapters.2.stateenabled
cluster.nodes.2.adapters.2.properties.device_namehme
cluster.nodes.2.adapters.2.properties.device_instance2
cluster.nodes.2.adapters.2.properties.transport_typedlpi
cluster.nodes.2.adapters.2.properties.netmask255.255.255.128
cluster.nodes.2.adapters.2.properties.ip_address172.16.1.2
cluster.nodes.2.adapters.2.properties.dlpi_heartbeat_quantum2000
cluster.nodes.2.adapters.2.properties.dlpi_heartbeat_timeout10000
cluster.nodes.2.adapters.2.ports.1.name0
cluster.nodes.2.adapters.2.ports.1.stateenabled
cluster.blackboxes.1.nameswitch1
cluster.blackboxes.1.stateenabled
cluster.blackboxes.1.properties.typeswitch
cluster.blackboxes.1.ports.1.name1
cluster.blackboxes.1.ports.1.stateenabled
cluster.blackboxes.1.ports.2.name2
cluster.blackboxes.1.ports.2.stateenabled
cluster.blackboxes.2.nameswitch2
cluster.blackboxes.2.stateenabled
cluster.blackboxes.2.properties.typeswitch
cluster.blackboxes.2.ports.1.name1
cluster.blackboxes.2.ports.1.stateenabled
cluster.blackboxes.2.ports.2.name2
cluster.blackboxes.2.ports.2.stateenabled
cluster.cables.1.properties.end1
cluster.nodes.1.adapters.1.ports.1
cluster.cables.1.properties.end2cluster.blackboxes.1.ports.1
cluster.cables.1.stateenabled
cluster.cables.2.properties.end1
cluster.nodes.1.adapters.2.ports.1
cluster.cables.2.properties.end2cluster.blackboxes.2.ports.1
cluster.cables.2.stateenabled
cluster.cables.3.properties.end1
cluster.nodes.2.adapters.1.ports.1
cluster.cables.3.properties.end2cluster.blackboxes.1.ports.2
cluster.cables.3.stateenabled
cluster.cables.4.properties.end1
cluster.nodes.2.adapters.2.ports.1
cluster.cables.4.properties.end2cluster.blackboxes.2.ports.2
cluster.cables.4.stateenabled
cluster.quorum_devices.1.named4
cluster.quorum_devices.1.stateenabled
cluster.quorum_devices.1.properties.votecount1
cluster.quorum_devices.1.properties.gdevname/dev/did/rdsk/d4s2
cluster.quorum_devices.1.properties.path_1enabled
cluster.quorum_devices.1.properties.path_2enabled
cluster.quorum_devices.2.named5
cluster.quorum_devices.2.stateenabled
cluster.quorum_devices.2.properties.votecount1
cluster.quorum_devices.2.properties.gdevname/dev/did/rdsk/d5s2
```

```
cluster.quorum_devices.2.properties.path_1enabled
cluster.quorum_devices.2.properties.path_2enabled
```

SEE ALSO

scconf(1M), **ccradm**(1M), **chkinfr**(1M)

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscr
Interface Stability	Evolving

NAME

cmm_ctl – cluster membership control

SYNOPSIS

/usr/cluster/dtk/bin/cmm_ctl [**–gdefFr**] [**–k***nodeid*] [**–s***nodeid*] [**–a***nodeid*] [**–t***nodeid*]

DESCRIPTION

cmm_ctl(1M) is used to control and get information about cluster membership.

OPTIONS

–g
print the current state of the cluster, including details about CMM reconfiguration, the various CMM configuration timeouts, and quorum-related information.

–d disable transport path monitoring in the cluster. Disabling monitoring allows the CMM to be stopped for debugging purposes without it being declared down. The caveat to this is that no new CMMs should be started unless all the CMMs on the cluster can participate (i.e. are not stopped). Otherwise, the CMM will timeout the reconfiguration and will abort the stopped CMM as soon as the latter is resumed.

–e enable transport path monitoring.

–f disable failfast on all the cluster nodes.

–F enable failfast on all the cluster nodes.

–k *nodeid*
effect an immediate failfast of the node with id *nodeid.*

–r force a reconfiguration of the CMM.

–s *nodeid*
stop the node with id *nodeid.*

–a *nodeid*
abort the node with id *nodeid.*

–t *nodeid*
determine if it is safe to take over data services from the node with id *nodeid* that has been declared down. The command will block until it is safe to perform the take over.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdtk
Interface Stability	Unstable

NAME

dcs_config – query DCS

SYNOPSIS

/usr/cluster/dtk/bin/dcs_config –c info [**–s** *service-name*] | [**–C** *service-class*] | [**–d** *device-path*]
/usr/cluster/dtk/bin/dcs_config –c status [**–s** *service-name*]

DESCRIPTION

dcs_config(1M) provides a more direct way to query the DCS than the standard **scconf**(1M) interfaces. While update interfaces are available in this command, they are not supported or recommended. All changes to the state of the DCS should be done using **scconf**(1M).

To query device services, use the "info" or "status" forms of the command. The info form gives general configuration information about the service, and the status form gives information about the service's current state. "info" and "status" commands without additional qualifying options shows all service classes and services in use.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdtk
Interface Stability	Unstable

NAME

/etc/cluster/ccr/dcs_service - Device Service CCR table

SYNOPSIS

/etc/cluster/ccr/dcs_service_*service_number*

DESCRIPTION

This table contains information for a device service. The name of the file includes the number of the device service, e.g. dcs_service_10. This device service number is also contained in the dcs_service_keys(4) CCR table. There is one dcs_service CCR table for each device service in the cluster.

Do *not* edit this table unless you have been instructed to as part of an emergency repair procedure.

FORMAT

The dcs_service tables are represented as ASCII text files. They begin with the standard CCR generation and checksum, and are followed by a number of TAB delimited keyword-value pairs. The keywords are defined for each type of device service.

The possible keys for each service are as follows:

DCS_ServiceName

The logical name of the service. This is usually the DID name for disk or tapes, or the name of the volume if this device service manages a volume manager device. This key-value pair is required for all device services.

DCS_ServiceClass

The class of device service, which defines the properties for that type of service. This service class is looked up in the dcs_service_classes(4) table to determine the specific properties. This key-value pair is required for all device services.

DCS_FailbackEnabled

The value associated with this key defines whether the device service should attempt to failback to a node which is joining the cluster, and has a higher preference for this device service than the node which is currently primary for the service. This key-value pair is required for all device services.

DCS_Suspended

The value associated with this key (yes,no) defines whether the device service is currently in maintenance mode.

DCS_NumberOfSecondaries

This is unused in this release, and the value should be set to 0.

DCS_Nodes

A definition of the nodes and priorities which are assigned to those nodes. Each node/priority pair is listed as (node number, priority). This list determines the possible number of primaries and secondaries for this device service. This key-value pair is required for all device services.

DCS_Devices

This is the list of devices which the device service manages. The first number in each comma-separated pair is the major number of the set of devices, the second is the minor numbers, or range of minor numbers which are managed. This key-value pair is required for all device services.

index (SDS Specific)

The data for the index key is the Solstice Disk Suite set number that this service manages.

node (SDS Specific)

This entry is the node number that created this SDS device service.

incarnation (SDS Specific)

A timestamp of when this device service was created.

state (SDS Specific)

The value for this pair is set to either "create" or "commit". The create state is a transient state, and is modified by the SDS binaries. For a service to be operational, this entry must be set to "commit".

vol_*minor_number* (VxVM Specific)

There is one key of this format for each VxVM volume under the diskgroup which this device service is managing. The data that corresponds to this key is a comma separated list of the VxVM volume name, major number for that volume, owner, group, and access permissions.

global_id (VxVM Specific)

The value associated with this key is retrieved from a VxVM library and is VxVM implementation specific. It contains a globally unique identifier for the VxVM volume.

EXAMPLES

The following is a sample dcs_service CCR table. This table shows a disk service with 2 possible primaries.

```
ccr_gennum      2
ccr_checksum
CD6CBDC4AE53CE8B6D0ACFD35D653182
DCS_ServiceName dsk/d3
DCS_ServiceClass        DISK
DCS_FailbackEnabled     no
DCS_Suspended   no
DCS_NumberOfSecondaries 0
DCS_Nodes       (1,0) (2,0)
DCS_Devices     (149,96-103)
```

Another example shows a sample SDS service.

```
ccr_gennum      4
ccr_checksum    17E9016673707F75256A102A325FFE9D
DCS_ServiceName galileo-tools
DCS_ServiceClass        SUNWmd
DCS_FailbackEnabled     no
DCS_Suspended   no
DCS_NumberOfSecondaries 0
DCS_Nodes       (1,64) (2,64)
DCS_Devices     (85,8192-16383)
index   1
node    1
incarnation     942823017
state   commit
```

And another examle showing a dcs_service table for a VxVM service.

```
ccr_gennum      1
ccr_checksum
8806A90565CD06007580DC1CC88CEED1
DCS_ServiceName dg3
DCS_ServiceClass        SUNWvxvm
DCS_FailbackEnabled     no
DCS_Suspended   no
```

```
DCS_NumberOfSecondaries 0
DCS_Nodes       (1,64) (2,64)
DCS_Devices     (54,84000-84001)
vol_84000       dg3v1, 54, 0, 0, 8576
vol_84001       dg3v2, 54, 0, 0, 8576
global_id       944725735.1062.phys-corvus-1
```

SEE ALSO

ccradm(1M), **scconf**(1M), **scswitch**(1M), **dcs_service_classes**(4), **dcs_service_keys**(4).

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscr
Interface Stability	Evolving

NAME

/etc/cluster/ccr/dcs_service_classes - Device Service Classes CCR table

SYNOPSIS

/etc/cluster/ccr/dcs_service_classes

DESCRIPTION

This table contains meta-data about device services. Device services are divided into a number of classes. Current examples of classes are the Veritas Volume Manager class, the raw disk class, the tape class, and the Solstice Disk Suite class.

Do *not* edit this table unless you have been instructed to as part of an emergency repair procedure. This table is provided as part of the Sun Cluster product.

FORMAT

Each class has an entry in this table with the name of the class as the key. The data segment of each entry consists of the program which is run to do service failovers, then a ':', and then the type of failover which that service class will do.

If the failover will be done automatically, without administrator intervention, the type will be "transparent". A failover type of "eio" means that this class of service cannot be failed over.

EXAMPLES

The following is a sample dcs_service_classes CCR table.

```
ccr_gennum     1
ccr_checksum
666FDA4A2CE19A69263600C8ACBD977D
SUNWmd  /usr/cluster/lib/run_reserve -C
SUNWmd:transparent
DISK    /usr/cluster/lib/run_reserve -C DISK:transparent
TAPE    /bin/true:eio
SUNWvxvm        /usr/cluster/lib/run_reserve -C
SUNWvxvm:transparent
```

SEE ALSO

ccradm(1M).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscr
Interface Stability	Evolving

NAME

/etc/cluster/ccr/dcs_service_keys - Device Service Keys CCR table

SYNOPSIS

/etc/cluster/ccr/dcs_service_keys

DESCRIPTION

This table contains a list of device services configured in the cluster. It is organized as a list of integer keys.

Each of these keys will correspond to a CCR table named dcs_service_*service_number*. That secondary table will contain all specific information about the device service.

Do *not* edit this table unless you have been instructed to as part of an emergency repair procedure.

EXAMPLES

The following is a sample dcs_service_keys CCR table.

```
ccr_gennum     3
ccr_checksum
202CB962AC59075B964B07152D234B70
1
2
3
```

SEE ALSO

ccradm(1M), **dcs_service**(4).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscr
Interface Stability	Evolving

NAME

/etc/cluster/ccr/did_instances - DID Device Information CCR table

SYNOPSIS

/etc/cluster/ccr/did_instances

DESCRIPTION

This CCR table describes the configuration of all DID devices in the cluster. It is a replacement for the /etc/did.conf file that was present in the SC2.2 release.

The did_instances table contains information about all DID devices in the cluster.

Do *not* edit this table unless you have been instructed to as part of an emergency repair procedure.

FORMAT

The did_instances table is represented as an ASCII text file. It begins with the standard CCR generation and checksum, and is followed by a number of TAB delimited keyword-value pairs. The keys for this table refer to the DID instance number for each device. Each DID device will have exactly one key-value pair. The data portion of the entry for a DID device is pipe-delimited with the following fields:

Device Type

This type matches one of the device type keys in the did_types(4) CCR table, and defines which type of DID device this instance corresponds to.

Device ID Class

Defines the type of device ID that this device provides. If none is given by the device, this field will be empty.

ASCII Device ID

An ascii representation of the device ID. For some devices, such as Fiber Channel disks, this field will be empty.

Device ID

A representation of the device ID in hex. This is the definitive representation, and the ID used to compare devices amongst nodes. For some classes of devices (e.g. tapes or CDROMs), there will be no device ID.

Subpath

A subpath for this device. The subpath is in the format <hostname>:<device path>. There is one subpath entry per physical path to the device.

EXAMPLES

The following is a sample did_instances table.

```
ccr_gennum     29
ccr_checksum
94772D7BB2419270AC31BAD299A63278
1      disk|DEVID_SCSI_SERIAL|SEAGATE
04572667|5345414741544520303435373236363700000000
0|phys-metcalf-1:/dev/rdsk/c0t0d0
2      disk|DEVID_SCSI_SERIAL|SEAGATE
00856075|5345414741544520303038353630373500000000
0|phys-metcalf-1:/dev/rdsk/c0t3d0
3      disk|DEVID_SCSI_SERIAL|FUJITSU
9711630312|46554a4954535520393731313633303331320
000|phys-metcalf-1:/dev/rdsk/c0t4d0
4      disk||||phys-metcalf-1:/dev/rdsk/c0t6d0
8191   tape||||phys-metcalf-1:/dev/rmt/0
```

```
5       disk|DEVID_SCSI_SERIAL|SEAGATE
04633667|53454147415445203034363333363637000000000|phys-metcalf-3:/dev/rdsk/c0t0d0
6       disk|DEVID_SCSI_SERIAL|FUJITSU
9709628338|46554a4954535520393730393632383333380000|phys-metcalf-3:/dev/rdsk/c0t4d0
7       disk||||phys-metcalf-3:/dev/rdsk/c0t6d0
8       disk|DEVID_SCSI_SERIAL|SEAGATE
9716E29051|534541474154452039373136453239303531000|phys-metcalf-2:/dev/rdsk/c1t2d0|phys-metcalf-3:/dev/rdsk/c1t2d0
9       disk|DEVID_SCSI_SERIAL|SEAGATE
9716E23723|534541474154452039373136453233373233000|phys-metcalf-2:/dev/rdsk/c1t3d0|phys-metcalf-3:/dev/rdsk/c1t3d0
10      disk|DEVID_SCSI_SERIAL|SEAGATE
04582876|53454147415445203034353823837360000000000|phys-metcalf-2:/dev/rdsk/c0t0d0
11      disk|DEVID_SCSI_SERIAL|FUJITSU
9710628852|46554a4954535520393731303632383835320000|phys-metcalf-2:/dev/rdsk/c0t4d0
12      disk||||phys-metcalf-2:/dev/rdsk/c0t6d0
```

SEE ALSO

ccradm(1M), **scdidadm**(1M), **did_types**(4).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscr
Interface Stability	Evolving

NAME

/etc/cluster/ccr/did_types - DID Metadata CCR table

SYNOPSIS

/etc/cluster/ccr/did_types

DESCRIPTION

This CCR table describes the metadata for DID device classes. This metadata is used to determine behavior characteristics for each individual device.

Do *not* edit this table unless you have been instructed to as part of an emergency repair procedure. This table is supplied with the Sun Cluster software.

FORMAT

The did_types table is represented as an ASCII text file. It begins with the standard CCR generation and checksum, and is followed by a number of TAB delimited keyword-value pairs. The keys for this table refer to the DID device type. Each DID type will have exactly one key-value pair. The data portion of the entry for a DID device is pipe-delimited with the following fields:

Device Type Number

An integer which is matched uniqely to a device type name key.

Unused

This field is unused in this release.

Discovery Subdirectory

The directory under /dev to discover this type of device in the Solaris device namespace.

Discovery Minor

The minor component of the logical device name. This is used as part of the search criteria in discovering new devices. Many devices have multiple minors for each device. The discovery minor is the minor name that would be found in order to guarantee existence of a physical device.

Unused

This field is unused in this release.

Default Minor Nodes

DID creates instance numbers (and their corresponding minor nodes) in blocks based on the type of device. This number defines how many minor nodes to create at a time for this device class.

Start Default Minor Nodes

This integer defines at what number to start creating instances for this type of device. Note that currently (due to a restriction of Solaris to 32-bit minor numbers) DID is limited to a maximum of 1024 physical devices in the cluster.

Next Default Minor Node

This integer defines what to add to the current minor node in order to arrive at the next minor node for this device type. Usually, this is either 1 or -1.

Minor Encoding

This is a colon-separated list of the details that define a minor number. This minor number is generated for each instance of this device type. It contains the minor number, the minor name, the corresponding minor name for the original device, and the device type (block or character, defined as b and c, respectively). The minor name is defined as the minor component of the path, as is seen in the physical device name.

EXAMPLES

The following is a sample did_types table.

```
ccr_gennum      0
ccr_checksum    E1515A7592728B2A3E79BC685340125F
disk    1|1-
3|rdsk|s2|0|100|1|1|0:s0,raw:a,raw:c|1:s1,raw:b,raw:c|2:s2,r
aw:c,raw:c|3:s3,raw:d,raw:c|4:s4,raw:e,raw:c|5:s5,raw:f,ra
w:c|6:s6,raw:g,raw:c|7:s7,raw:h,raw:c|0:s0,blk:a:b|1:s1,blk:
b:b|2:s2,blk:c:b|3:s3,blk:d:b|4:s4,blk:e:b|5:s5,blk:f:b|6:s6,bl
k:g:b|7:s7,blk:h:b
tape    2|1-3|rmt|l|1|10|8191|-
1|0:,tp::c|1:b,tp:b:c|2:bn,tp:bn:c|3:c,tp:c:c|4:cb,tp:cb:c|5:cbn
,tp:cbn:c|6:cn,tp:cn:c|7:h,tp:h:c|8:hb,tp:hb:c|9:hbn,tp:hbn:c|
10:hn,tp:hn:c|11:l,tp:l:c|12:lb,tp:lb:c|13:lbn,tp:lbn:c|14:ln,t
p:ln:c|15:m,tp:m:c|16:mb,tp:mb:c|17:mbn,tp:mbn:c|18:mn,t
p:mn:c|19:n,tp:n:c|20:u,tp:u:c|21:ub,tp:ub:c|22:ubn,tp:ubn:
c|23:un,tp:un:c
```

SEE ALSO

ccradm(1M), **scdidadm**(1M), **did_instances**(4).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscr
Interface Stability	Evolving

NAME

ddb – dump debug buffers

SYNOPSIS

/usr/cluster/dtk/bin/ddb *kernel-symbols kernel-memory*

DESCRIPTION

ddb(1M) displays the threadlist and the contents of the kernel debugging buffers from either a running system or a system crash dump.

OPERANDS

The following operands are supported:

kernel-symbols
the kernel symbols, typically /dev/ksyms for a running system or the unix file from a system crash dump.

kernel-memory
the kernel memory, typically /dev/mem for a running system or the vmcore file from a system crash dump.

EXAMPLES

To get information from the running system:

```
# /usr/cluster/dtk/bin/ddb /dev/ksyms /dev/mem
```

To get information from a system crash dump:

```
# /usr/cluster/dtk/bin/ddb unix.0 vmcore.0
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdtk
Interface Stability	Unstable

NOTES

ddb(1M) uses **adb**(1) macros and adb aborts macros when a symbol is not found, so if the output ends with "symbol not found" the /usr/cluster/dtk/lib/adb/dump_all macro contains a symbol not found in the kernel symbols. This could be because a module was not loaded, such as when scalable services are not being used and misc/cl_net has not been loaded and ddb can't find the the net_dbg symbol. Or it could be because dump_all macro is not matched to the version of SC3.0 being run.

SEE ALSO

dump_all(5), **dump_one**(5)

NAME

dump_all – adb macro for dumping all kernel debugging buffers

DESCRIPTION

dump_all is an **adb**(1) macro for dumping all the kernel debugging buffers on a running system or in a system crash dump. dump_all is called from **ddb**(1M), after printing a threadlist, so ddb is probably a simpler command to use. To use the dump_all macro the -I option should be given to adb to find the DTK macros in /usr/cluster/dtk/lib/adb.

EXAMPLES

To dump all the debugging buffers from the running system:

```
# adb -I /usr/cluster/dtk/lib/adb -k /dev/ksyms /dev/mem
$<dump_all
$q
```

To dump all the debugging buffers from a system crash dump:

```
# adb -I /usr/cluster/dtk/lib/adb -k unix.0 vmcore.0
$<dump_all
$q
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdtk
Interface Stability	Unstable

NOTES

adb aborts macros when a symbol is not found, so if the output ends with "symbol not found" the dump_all macro contains a symbol not found in the the kernel symbols. This could be because a module was not loaded, such as when scalable services are not being used and misc/cl_net has not been loaded and adb can't find the the net_dbg symbol. Or it could be because the dump_all macro is not matched to the version of SC3.0 being run.

SEE ALSO

ddb(1M), **dump_one**(5)

NAME

dump_one – adb macro for dumping a kernel debugging buffer

DESCRIPTION

dump_one is an **adb**(1) macro for dumping a kernel debugging buffer on a running system or in a system crash dump. dump_one is used by the **dump_all**(5) macro. To use the dump_one macro the -I option should be given to adb to find the DTK macros in /usr/cluster/dtk/lib/adb.

EXAMPLES

To dump all the debugging buffers from the running system:

To dump the mount debugging buffer from the running system:

.

adb -I /usr/cluster/dtk/lib/adb -k /dev/ksyms /dev/mem mount_dbg

$<dump_one

$q

.

To dump the mount debugging buffer from a system crash dump:

.

adb -I /usr/cluster/dtk/lib/adb -k unix.0 vmcore.0 mount_dbg

$<dump_one

$q

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdtk
Interface Stability	Unstable

NOTES

The **dump_all**(5) macro contains a list of the buffers that can be dumped with dump_one.

SEE ALSO

ddb(1M), **dump_all**(5)

NAME

orbadmin – display ORB state

SYNOPSIS

/usr/cluster/dtk/bin/orbadmin [**–U**] [**–R** *opts*] [**–P** *opts*] [**–B** *opts*] [**–m** *opts*] [**–f** *opts*] [**–u** *opts*] [**–c** *opts*] [**–r** *opts*] [**–n** *node*] [*interval* [*count*]]

DESCRIPTION

orbadmin(1M) is a general purpose tool for observing behaviour in the orb. *interval* is the number of seconds between iterations of the state being displayed. *count* is the number iterations and defaults to one when *interval* is not specified. When interval is specified without a count, the command repeats forever. State commands report current values each interval. Statistics commands report current values the first time, and the difference on subsequent intervals.

OPTIONS

–U
display the length of the unref_threadpool's work queue, or in other words the number of unreference notications that have not been processed.

–R *opts*
display state refcount, where *opts* is one of:

all
display all refcount state options

ref
number of reference jobs

ack
number of ack jobs

–P *opts*
display state resource_pool, where *opts* is one of:

pool=*xxx*
specify resource_pool number, defaults to DEFAULT_POOL

all
display all resource_pool state options

total
total number of threads

free
number of free threads

needed
number of needed threads

torecall
number of threads to recall

torelease
number of threads to release

active
number of active threads, convenience function for (total - free)

–B *opts*
display state resource_balancer, where *opts* is one of:

pool=*xxx*
specify resource_pool number, defaults to DEFAULT_POOL

all
display all resource_balancer state options

total
total number of threads

unallocated
number of unallocated threads

–m *opts*
display message statistics, where *opts* is one of:

all
display all message statistic options

msg_types
display msg_types_out + msg_types_in

msg_types_out

msg_types_in

orphans

memory
display both memory_out + memory_in

memory_out
the number of bytes used by messages going out of this node

memory_in
the number of bytes used by messages coming in to this node

size_histogram
display for both out and in

size_histogram_out

size_histogram_in

–f *opts*
display flow control statistics, where *opts* is one of:

all
display all flow control statistic options

policy
statistics - msg_by_policy

block
statistics - blocked, unblocked, reject

pool
statistics - resource_pool actions

balancer
statistics - resource_balancer actions

–u *opts*
display unref_threadpool statistics, where *opts* is one of:

all
display all unref statistic options

hist_wait_handler

hist_wait_xdoor

hist_queue_length

hist_work_handler

hist_work_xdoor

unref threadpool statistics software is optional, and the command reports an error when not supported.

–c *opts*

change flow control settings, where *opts* is one of:

pool=*xxx*
resource_pool number, defaults to DEFAULT_POOL

low=*xxx*
new value for threads low

mod=*xxx*
new value for threads moderate

high=*xxx*
new value for threads high 10

min=*xxx*
new value for threads minimum 10

Any unspecified thread value remains unchanged.

–r *opts*

read flow control settings, where *opts* is one of:

pool=*xxx*
resource_pool number, defaults to DEFAULT_POOL

all

–n *node*

restricts display of information to *node* instead of the default of displaying information for all nodes.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdtk
Interface Stability	Unstable

NAME

print_net_state – print networking state

SYNOPSIS

/usr/cluster/dtk/bin/print_net_state –p [**–v**] [**–P** *node*] [**–G** *node*] **–S** *service*
/usr/cluster/dtk/bin/print_net_state –s [**–P** *node*] [**–G** *node*] [**–S** *service*]
/usr/cluster/dtk/bin/print_net_state –f *node* [**–v**] **–G** *node* **–S** *service*
/usr/cluster/dtk/bin/print_net_state –h –G *node*

DESCRIPTION

print_net_state displays state information about clustering networking. There are four variations to get information about PDT tables, services, forwarding lists, and hash tables. A GIF node must be specified to display forwarding lists and hash tables and can be specified with **–G.** A service name must also be specified to display fowarding lists and can be specified with **–S.**

OPTIONS

–p
print PDT table, requires **–S.**

–s print service(s). An invocation of **–s** without specifying a service prints all the services, however, it is currently not possible to "wildcard" the service for printing out the contents of the forwarding lists or PDT tables. A specific service must be given for those options.

–f *node*
print forwarding lists for node *node* A *node* of 0 indicates all nodes. Requires **–G** and **–S.**

–h print hash tables, requires **–G.**

–v print verbosely.

–P *node*
specify PDT server node. Currently not implemented.

–G *node*
specify GIF node.

–S *service*
specify service name.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdtk
Interface Stability	Unstable

NAME

replctl – interface to the replica manager

SYNOPSIS

/usr/cluster/dtk/bin/replctl [**–d** *serv-desc* [*repl-desc*]]
/usr/cluster/dtk/bin/replctl [**–f**] **–c** *serv-desc repl-desc state*
/usr/cluster/dtk/bin/replctl –x *serv-desc*

DESCRIPTION

The **replctl** command provides an interface to the replica manager. The replica manager is an internal component of SC3.0 which manages kernel-level HA services that are implemented using the replica framework.

replctl allows the state of HA services to be viewed and changed. It is primarily a test utility for the replica manager and services that are written using it.

Each HA service has a name and a number of "providers" or "replicas". Services have one of the following states UNINIT, UP, DOWN, UNSTABLE.

UNINIT
means that the service exists, but has never had a primary. This is usually a short lived state during service initialization.

UP means that the service has a primary.

DOWN
means that the service was once up, but the primary died and there were no secondaries capable of taking over.

UNSTABLE
is a transitory state which indicates that a change (for example, a switchover) is occuring to the state of the service.

Within a service each provider has a state of PRIMARY, SECONDARY or SPARE. When a service is in the UP state, one provider is in the PRIMARY state and the others are in SECONDARY state. The concept of SPARE is not fully supported in SC3.0, so it is rare to find a provider in SPARE state (there are small windows during service startup or addition of new providers where providers can be in SPARE state). This will only occur when the service is UNSTABLE.

replctl without any arguments displays the state of all HA services.

OPTIONS

–d *serv-desc* [*repl-desc*]
Display the state of the service with the description *serv-desc* and optionally limit the output to the replica with the description *repl-desc.* replctl without arguments will display all the service and replica descriptions in use.

–c *serv-desc repl-desc state*
Change the state of service *serv-desc*'s replica *repl-desc* to state *state*. States may be **primary**, **secondary**, **spare**or **remove**. Legal transitions are:

SECONDARY to PRIMARY: this triggers a switchover.

PRIMARY to SECONDARY: this triggers a switchover.

SECONDARY to SPARE: this removes state from a secondary, but the spare is immediately promoted back to secondary again.

SECONDARY to remove: this converts the secondary to spare and then removes the provider from the configuration. There is no interface to add the spare back again, so this should not usually be used, except for testing purposes.

SPARE to remove: this removes a spare but since we rarely have long lived spares, this is not very practical for SC3.0.

–f Force a state change, ignored in SC3.0.

–x *serv-desc*
performs a shutdown of a service *serv-desc*. This asks the primary to perform a shutdown and if it agrees, the service and its supporting data structures are removed from the replica manager.

NOTES

Note that replctl supports some options that are not currently product requirements, and using it to perform unsupported operations may trigger failures. Specifically, only the HA device service officially supports switchovers. None of the services officially support the "remove" request. None of the services support the -x option directly (some services support it, but only in the context of a number of other operations, such as unmount of a file system). This command should be used with care. In general, it is recommended that neither the -c nor -x option be used outside the SC3.0 development group.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdtk
Interface Stability	Unstable

NAME

reserve – query and clear SCSI reservations

SYNOPSIS

/usr/cluster/lib/sc/reserve –c *subcommand* **–d** *disk*

DESCRIPTION

reserve(1M) provides a method of querying and clearing the SCSI reservations on cluster disks.

There are two different types of SCSI reservations. There is the old form, SCSI-2 reservations, and a newer form termed Persistent Group Reservations (PGR). **reserve**(1M) is capable of dealing with both reservation types.

OPTIONS

–c *subcommand*

perform the subcommand:

scrub scrub SCSI-3 reservations from the disk. This should not be done during normal cluster operation. Doing so may compromise data integrity.

release

remove SCSI-2 reservations from the disk. This should not be done during normal cluster operation. Doing so may compromise data integrity.

status

determine if this host has access to the disk. Prints 0 if the host has access, 1 if it does not.

inkeys

print PGR registration keys on the disk.

inresv

print PGR reservations on the disk.

–d *disk*

specifies *disk* as the disk to which the command is applied.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Unstable

NAME

rgmd_debug – RGM daemon debugging

SYNOPSIS

/usr/cluster/lib/sc/rgmd_debug on | **off** | **printbuf**

DESCRIPTION

rgmd_debug(1M) is a utility which facilitates debugging of the rgmd. This script permits the verbose debugging mode of the rgmd to be turned on or off on the local node. To enable or disable verbose debugging mode on multiple nodes, the script has to be invoked on each node.

It also allows the trace buffer in the rgmd to be dumped out. The trace buffer is a ring buffer that has essentially the same information as the verbose debugging messages generated when debugging mode is turned on. When the trace buffer gets full, the oldest messages get dropped.

Each trace buffer entry has the thread ID of the rgmd thread that generated the message and the last 8 digits of a microsecond timestamp.

When verbose debugging mode is turned on, the rgmd will syslog a large number of messages which will appear on the console and in /var/adm/messages. These messages are intended for the use of product sustaining engineers within Sun; they are not intended to be used by customers.

OPERANDS

on
enable verbose debugging mode

off disable verbose debugging mode

printbuf
dump the contents of the rgmd debugging buffer

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

SEE ALSO

scdtk_enable(5), **scdtk_collect**(5)

NAME

/etc/cluster/ccr/rgm_r*x_xxx* - Resource Group Manager CCR tables

SYNOPSIS

/etc/cluster/ccr/rgm_rt_directory
/etc/cluster/ccr/rgm_rt_*resource_type*
/etc/cluster/ccr/rgm_rg_*resource_group*

DESCRIPTION

The Resource Group Manager (RGM) stores configuration data in Cluster Configuration Repository (CCR) tables. There is one resource type directory table (**rgm_rt_directory**), one table for each resource type (**rgm_rt_***resource_type*), and one table for each resource group (**rgm_rg_***resource_group*).

Do *not* edit these tables unless you have been instructed to as part of an emergency repair procedure.

FORMAT

The RGM CCR tables are represented in the file system by ASCII text files. Each file starts with a generation number and checksum, after which the content depends on whether it is a directory or holds the configuration information for a resource type or resource group. The content of each file type is as follows:

rgm_rt_directory

The RGM resource type directory table contains the name of each resource type on a separate line.

rgm_rt_*resource_type*

A resource type CCR table contains the properties of the resource type and the resource property parameters. Each property is listed on a separate line as a TAB delimited keyword-value pair. The value for each parameter table entry is a semicolon separated list of attributes, with no spaces. Within this list, each attribute is specified by keyword=value.

rgm_rg_*resource_group*

A resource group CCR table contains the properties of the resource group and an entry for each of its resources. Each line in this table contains a keyword and a value separated by a TAB character. The value for each resource entry is a semicolon separated list of properties, with no spaces. Within this list, each property is specified by keyword=value.

REPAIR PROCEDURE

Please refer to the **ccradm**(1M) man page for instructions on how to repair a corrupted CCR table.

EXAMPLES

The following is a sample **rgm_rt_directory** CCR table.

```
ccr_gennum34
ccr_checksum9385C16A51122BA7906E81EB82B50098
SUNW.LogicalHostname
SUNW.SharedAddress
SUNW.test_svc
```

The **rgm_rt_SUNW.LogicalHostname** resource type CCR table follows.

```
ccr_gennum0
ccr_checksumFC0366D28AE5E17E8FE1EAB7943FF664
RT_basedir/usr/cluster/lib/rgm/rt/hafoip
RT_descriptionLogical Hostname Resource Type
STARThafoip_start
STOPhafoip_stop
```

PRIMARIES_CHANGED
VALIDATEhafoip_validate
UPDATEhafoip_update
MONITOR_STARThafoip_monitor_start
MONITOR_STOPhafoip_monitor_stop
MONITOR_CHECKhafoip_monitor_check
PRENET_STARThafoip_prenet_start
Single_instanceFALSE
Init_nodesRG_PRIMARIES
SysdefinedTypeLogical_hostname
Installed_nodes*
FailoverFALSE
API_version2
RT_version1.0
PkglistSUNWscu
p.START_TIMEOUT
Tunable=ANYTIME;Type=INT;Default=300;Min=0;Max
=3600;
p.STOP_TIMEOUT
Tunable=ANYTIME;Type=INT;Default=300;Min=0;Max
=3600;
p.VALIDATE_TIMEOUT
Tunable=ANYTIME;Type=INT;Default=300;Min=0;Max
=3600;
p.UPDATE_TIMEOUT
Tunable=ANYTIME;Type=INT;Default=300;Min=0;Max
=3600;
p.MONITOR_START_TIMEOUT
Tunable=ANYTIME;Type=INT;Default=300;Min=0;Max
=3600;
p.MONITOR_STOP_TIMEOUT
Tunable=ANYTIME;Type=INT;Default=300;Min=0;Max
=3600;
p.MONITOR_CHECK_TIMEOUT
Tunable=ANYTIME;Type=INT;Default=300;Min=0;Max
=3600;
p.PRENET_START_TIMEOUT
Tunable=ANYTIME;Type=INT;Default=300;Min=0;Max
=3600;
p.Failover_mode
Tunable=ANYTIME;Type=ENUM;Default=HARD;Enum
list=NONE,HARD,SOFT;
p.Cheap_probe_interval
Tunable=AT_CREATION;Type=INT;Default=60;Min=0;
Max=3600;
p.Thorough_probe_interval
Tunable=AT_CREATION;Type=INT;Default=60;Min=0;
Max=3600;
p.Retry_count
Tunable=AT_CREATION;Type=INT;Default=2;Min=0;
Max=10;
p.Retry_interval
Tunable=AT_CREATION;Type=INT;Default=60;Min=0;
Max=3600;

```
x.NetIfList
Tunable=AT_CREATION;Type=STRINGARRAY;Defaul
t=;Max=1024;\
Description="List of NAFO interfaces on each node";
x.HostnameList
Tunable=AT_CREATION;Type=STRINGARRAY;Defaul
t=;Max=1024;\
Description="List of hostnames this resource manages";
```

The following is an example of a resource group CCR table:

```
ccr_gennum0
ccr_checksumD7A6AEF698039A2D36F41E845978756A
UnmanagedFALSE
Nodelistphys-schost-1,phys-schost-2
Maximum_primaries1
Desired_primaries1
FailbackFALSE
Resource_listR1,R2
RG_dependencies
Global_resources_used*
Logical_hostFALSE
Pathprefix
RG_description
Pingpong_interval3600
R1
Type=SUNW.test_svc;R_description=;On_off_switch=0;
Monitored_switch=0;\
Resource_dependencies=;Resource_dependencies_weak=;
START_TIMEOUT=300;\
STOP_TIMEOUT=300;VALIDATE_TIMEOUT=3600;IN
IT_TIMEOUT=3600;Failover_mode=NONE;\
Extension
R2
Type=SUNW.test_svc;R_description=;On_off_switch=0;
Monitored_switch=0;\
Resource_dependencies=;Resource_dependencies_weak=;
START_TIMEOUT=300;\
STOP_TIMEOUT=300;VALIDATE_TIMEOUT=3600;IN
IT_TIMEOUT=3600;Failover_mode=NONE;\
Extension
```

Here is an example of how to repair a resource group CCR table named *rgm_rg_xxx*.

Reboot all nodes in non-cluster mode.

Edit the file */etc/cluster/ccr/rgm_rg_xxx* on all nodes to contain the correct data. The file must be identical on all nodes.

On all nodes, recompute the checksum and designate this CCR file to be the override version.

```
phys-schost-1# /usr/cluster/lib/sc/ccradm –i
/etc/cluster/ccr/rgm_rg_xxx –o
phys-schost-2# /usr/cluster/lib/sc/ccradm –i
/etc/cluster/ccr/rgm_rg_xxx –o
phys-schost-3# /usr/cluster/lib/sc/ccradm –i
```

```
/etc/cluster/ccr/rgm_rg_xxx –o
phys-schost-4# /usr/cluster/lib/sc/ccradm –i
/etc/cluster/ccr/rgm_rg_xxx –o
```

Reboot all nodes in cluster mode.

SEE ALSO

ccradm(1M), **scrgadm**(1M), **rt_reg**(4).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

NAME

scdtk_enable - information for enabling diagnostic information collection

DESCRIPTION

This man page provides information for enabling the collection of diagnostic information. Diagnosis can be made easier in several ways including:

- logging of events and state transitions
- saving of state information on failure
- execution under control of debugging facilities

Following are sections describing various ways to enable diagnostic information.

After enabling the collection of diagnostic information, the information can be collected using the methods described in **scdtk_collect**(5).

Enabling syslog Messages for Data Services

Various levels of debugging output are available for data services through syslog. To enable this, entries for local7.debug need to be added to /etc/syslog.conf:

```
local7.debug    /dev/console
local7.debug    /var/adm/messages
```

The columns must be separated by tab(s). The file can be edited or appended with the commands:

```
# echo local7.debug\\t/dev/console >> /etc/syslog.conf
# echo local7.debug\\t/var/adm/messages >> /etc/syslog.conf
```

Restart syslogd with:

```
# pkill –HUP syslogd
```

Test the addition to /etc/syslog.conf with:

```
# logger –p local7.debug test
```

You should see output with "test" on the console and at the end of /var/adm/messages.

The file that controls the debug level is called "loglevel" and is stored in a directory specific to a resource type. We currently use the Vendor ID and resource type name in the path for the debug file:

/var/cluster/rgm/rt/*rtname*/loglevel

To turn on debugging for a resource type, you would do the following:

```
# mkdir –p /var/cluster/rgm/rt/rtname
# echo debuglevel > /var/cluster/rgm/rt/rtname/loglevel
```

For example:

```
# mkdir –p /var/cluster/rgm/rt/SUNW.test_svc
# echo 9 > /var/cluster/rgm/rt/SUNW.test_svc/loglevel
```

Debug levels:

0 is for no debug messages – error and information messages only

1 is for the fewest debug messages, as well as all error and information messages

2-8 is for increasingly more debug messages

9 is for the most debug messages

Notes:

The debug level is read at the start of each method invocation. Changing the debug level will take effect on the next call of a given method.

Enabling Resource Group Manager syslog Debug Messages

If you see unexpected messages prefixed with "Cluster.RGM" it may be helpful to enable additional Resource Group Manager (RGM) syslog messages. If you have a reproducible problem it would make sense to reboot and enable debug output to gather more information. If you see some strange behavior while the system is running you could enable debug output to try to diagnose the problem.

To enable RGM syslog debug messages run:

```
# /usr/cluster/lib/sc/rgmd_debug on
```

This will cause RGM debugging messages to be written to the console and /var/adm/messages.

To disable RGM syslog debug messages run:

```
# /usr/cluster/lib/sc/rgmd_debug off
```

Component Debug Buffers

Some of the clustering components use debug buffers to log diagnostic information. To enable all the debug buffers on a running system use adb:

```
# adb -wk /dev/ksyms /dev/mem
haci_dbg_option/WFFFFFFF
orb_trace_options/WFFFFFFFF
clnet_net_dbg_fwd/W1
$q
```

These settings will not persist across reboot, but that can be done by adding these lines to /etc/system:

```
set cl_comm:haci_dbg_option = 0xffffffff
set cl_comm:orb_trace_options = 0xffffffff
set cl_net:clnet_net_dbg_fwd = 1
```

Specific bits in haci_dbg_option can be set using the OR form of set. These are cumulative so:

```
set cl_comm:haci_dbg_option | 0x4
 set cl_comm:haci_dbg_option | 0x80
```

will set both bits. orb_trace_options has bits that can be set similarly.

Operating System Crash Dumps

When the operating system encounters a fatal error it will, by default, produce a crash dump. At times it is necessary to force a crash dump. The **dumpadm**(1M) command tells you the current dump settings, for example:

```
# dumpadm
Dump content: kernel pages
Dump device: /dev/dsk/c0t3d0s1 (swap)
Savecore directory: /var/crash/zoon
Savecore enabled: yes
```

By default the dump device is the swap partition. **savecore**(1M), which runs during boot to extract the crash data from the dump device, stores the data in /var/crash/'uname–n'. You can use dumpadm to specify a different dump device or savecore directory.

Under certain circumstances it is beneficial to get a "live" crash dump of the system. This dumps state while the system keeps running. To be able to get a live crash dump it is necessary to configure a dedicated dump device with dumpadm.

For maximum debugability we recommend:

- if possible, configure a dedicated dump device that is as large as physical memory
- if the dump device is sufficiently large, set dump content to "all"

In practice, dump compression ratios are typically greater than 2.0 so it's usually safe to have a dump device that is half the size of physical memory.

As an example, if /dev/dsk/c7t3d0s2 is unused and available as a dedicated dump device and we want to save all pages and store it in /space/crash we would run dumpadm with these options and get this output:

```
# dumpadm –c all –d /dev/dsk/c7t3d0s2 –s /space/crash
Dump content: all pages
Dump device: /dev/dsk/c7t3d0s2 (dedicated)
Savecore directory: /space/crash
Savecore enabled: yes
```

Process Core Files

When a process terminates abnormally it may produce a core file that can be helpful in determining what went wrong. By default, a file called "core" is created in the process's current directory. To prevent core files from being overwritten by subsequent core dumps and to make it easier to identify where core files came from, **coreadm**(1M) is used to change the default. We recommend:

- enable global core file generation
- enable global set-ID cores
- use "%f.%p.%t" in the core file name pattern

Choose an appropriate directory for core dumps and make sure the directory exists. As an example, if we want core dumps to go in /var/core we run these commands:

```
# mkdir /var/core
# coreadm –e global –e global-setid
# coreadm –g '/var/core/core.%f.%p.%t'
# coreadm
global core file pattern: /var/core/core.%f.%p.%t
init core file pattern: core
global core dumps: enabled
per-process core dumps: enabled
```

```
global setid core dumps: enabled
per-process setid core dumps: disabled
global core dump logging: disabled
```

To test the settings, abort a command with ctrl–\ (control backslash), such as:

```
# sleep 13
^\Quit – core dumped
# ls –l /var/core
total 134 -rw-------   1 root     other      67640 Feb 10 15:37 core.sleep.350.950225870
```

This will produce two core files, one named core in the current directory and the other shown above. If you only want core files to be generated in the directory specified with -g, disable per-process core files with:

```
# coreadm -d process
```

Booting With kadb

During normal cluster operation it is not necessary to run kadb, the kernel debugger. However there may be times when it is necessary to boot the kernel under the control of kadb in order to obtain diagnostic information.

To configure a machine to boot kadb automatically, use the eeprom command to set the boot file to kadb:

```
# eeprom boot-file=kadb
```

With this setting of boot-file, kadb will boot the appropriate kernel for your system, either 32-bit or 64-bit.

To configure a machine to boot automatically a 32-bit kernel under kadb:

```
# eeprom boot-file="kadb –D kernel/unix"
```

Setting boot-file establishes a default file (and optionally arguments) that will be used when no arguments are given to the boot command at the PROM's ok prompt. To unset boot-file:

```
# eeprom boot-file=""
```

This will restore the machine to its default behavior, which does not run the kernel under control of kadb.

To boot one instance of the kernel under kadb, arguments are given to the boot command. For example to boot the default kernel with kadb:

```
ok boot kadb
```

To boot the 32-bit kernel:

```
ok boot kadb –D kernel/unix
```

The arguments to boot temporarily override the boot-file setting.

SEE ALSO

coreadm(1M), **dumpadm**(1M), **eeprom**(1M), **kadb**(1M), **rgmd_debug**(1M), **savecore**(1M), **scdtk_collect**(1M), **scshutdown**(1M), **syslog.conf**(4)

SEE ALSO

ocean(1)

NAME

scdtk_collect - information for diagnostic information collection

DESCRIPTION

This man page provides information for the collection of diagnostic information. Before certain information can be collected the collection needs to be enabled by the methods described in **scdtk_enable**(5).

Following are sections describing various information that can be collected and how to collect it.

Collecting syslog Messages for Data Services

If you have enabled syslog messages for data services (as described in **scdtk_enable**(5)) the messages will appear on the console and in /var/adm/messages. Any method to collect /var/adm/messages, such as using Explorer, will suffice for collecting the messages.

Collecting Resource Group Manager Debug Messages

To display Resource Group Manager diagnostic information, run:

/usr/cluster/lib/sc/rgmd_debug printbuf

This will dump the debugging buffers. The debugging buffers are always used regardless of whether syslog debugging has been turned on with rgmd_debug.

See the gcore section below for more info about rgmd.

Collecting Component Debug Buffers

To dump the contents of debugging buffers used by kernel-level components of SC3.0, the ddb script is run with arguments for a symbol table and memory image. To run this on the currently running system:

/usr/cluster/dtk/bin/ddb /dev/ksyms /dev/mem

If you have a crash dump you can:

/usr/cluster/dtk/bin/ddb unix.0 vmcore.0

ddb prints a threadlist for the kernel and then dumps all the debugging buffers, so it can produce a lot of output.

If you would like to dump a single debugging buffer it can be done with the dump_one adb macro. For example, let's say you want to dump the mount_dbg buffer on the running system:

adb -I /usr/cluster/dtk/lib/adb -k /dev/ksyms /dev/mem

mount_dbg$<dump_one

$q

As with ddb, the unix and vmcore files from a crash dump can be substituted for /dev/ksyms and /dev/mem.

Generating Operating System Crash Dumps

If the kernel encounters a fatal error it will "panic" and produce a crash dump. In most situations the dump will happen automatically. If the kernel is booted under the control of kadb, however, the debugger will get control and you tell the debugger to continue with ":c" to let the dump happen. For example:

panic[cpu0]/thread=f73c2b20: BAD TRAP: ...
...
panic: entering debugger (continue to save dump)

stopped at 0xfbd01028: ta 0x7d

```
kadb[0]: :c
syncing file systems... 1 done
dumping to /dev/dsk/c0t3d0s1, offset 26935296
...
```

If the system has become unresponsive and you want to get a crash dump you should first get to the PROM's ok prompt. Typing L1-A or STOP-A at a Sun keyboard or sending a BREAK with telnet or tip when using a serial connection will interrupt the kernel. From the ok prompt run sync:

```
ok sync
```

If the kernel was booted with kadb you need to exit kadb first before running sync:

```
kadb[0]: $q
ok sync
```

(Alternatively you can use $<systemdump at the kadb prompt, which will exit kadb and remind you to run sync to get a crash dump.)

After the kernel's data is dumped to the dump device the system will reboot and during boot, savecore will copy the data from the dump device into the directory specified with dumpadm.

If one node has crashed and you would like to get crash dumps from the other nodes without bringing them down you can take a "live" crash dump with savecore -L. This requires a dedicated dump device. The kernel's data will be written to the dump device and immediately read and put into the savecore directory. For example:

```
# savecore -L
dumping to /dev/dsk/c0t3d0s4, offset 65536
100% done: 10217 pages dumped, compression ratio 2.51, dump succeeded System dump time: Wed Mar 22 16:15:49 2000
Constructing namelist /export/home/crash/unix.0
Constructing corefile /export/home/crash/vmcore.0 100% done: 10217 of 10217 pages saved
```

Because savecore -L dumps the kernel state while the kernel is running, it will almost certainly contain inconsistent state. If you collect live crash dumps tell the person examining the dumps that they are live crash dumps.

Generating Process Core Files

If a user-level process appears to be stuck or is misbehaving, a core file of the process state can be generated with the gcore utility. By default gcore will create a file in the current directory named "core.pid", where pid is the process-id of the target process. gcore does not use the core directory specified with coreadm so you may want to use -o to specify where gcore will put the core file. For example to get a core of the RGM daemon:

```
# ps -ef | grep rgmd
root 8159     1  0 13:55:51 ?       0:04 /usr/cluster/lib/sc/rgmd
# gcore -o /var/core/core.rgmd 815
gcore: /var/core/core.rgmd.8159 dumped
```

Getting a process's core file with gcore does not disturb the process.

If you have a core file from rgmd you can use the dump_one macro to dump the rgmd's debugging buffer:

```
# adb -I /usr/cluster/dtk/lib/adb /usr/cluster/lib/sc/rgmd core.rgmd.8159
rgm_dbg_buf$<dump_one
```

Other User-Level Tools

Data from misbehaving processes can also be collected using truss(1) and the proc(1) tools such as pstack and pfiles. Some of the proc tools can also be used on core files.

SEE ALSO

adb(1), **ddb**(1M), **dump_one**(5), **rgmd_debug**(1M), **scdtk_enable**(5)

Appendix D

Sun Cluster 3.0 7/01 (Update 1) Support Readiness Training

In ES-333, Sun Cluster 3.0 Administration, you learned how to install and configure a cluster. The IES-SC33 course, Advanced Diagnostics & Troubleshooting, provides you with tools and techniques to troubleshoot and tune a cluster. There is also a Web-based course that includes additional material that you may find useful in your jobs.

WZI-2747: Sun Cluster 3.0 7/01 (Update 1) Support Readiness Training

The WZI-2747, Sun Cluster 3.0 7/01 Support Readiness Training, includes material on all new features that were added to Sun Cluster 3.0 for Sun Cluster 3.0 7/01. This course is self-paced and is free to all internal audiences. The topics in this course include:

Module 1: New Features Support

- Topic 1: Overview of New Supported Features
- Topic 2: VxVM with SDS Root Mirror
- Topic 3: Solaris Resource Manager (SRM) 1.2 Coexistence
- Topic 4: Cluster Volume Manager 32-bit (with OPS 8.1.6)
- Topic 5: Cluster File System Performance Improvements

Module 2: New Tools

- Topic 1: Overview of New Tools
- Topic 2: SunPlex Manager
- Topic 3: scvxinstall

- Topic 4: RGM Extension to scsetup
- Topic 5: Data Service Development Library (DSDL)
- Topic 6: SunPlex Agent Builder

Module 3: New Hardware Support

- Topic 1: Overview of New Supported Hardware
- Topic 2: New Supported Servers
  - Unit 1: Sun Fire 280R (Littleneck)
  - Unit 2: Sun Fire 3800 / 4800 / 4810 / 6800 Servers
  - Unit 3: Sun Enterprise 420R Server
  - Unit 4: Netra t 1400 / 1405 Server
  - Unit 5: Netra t 1120 / 1125 Server
  - Unit 6: Netra T1 AC200 / DC200 Server
- Topic 3: New Supported Storage
  - Unit 1: Sun StorEdge(TM) T3 Array
  - Unit 2: A3500FC Array
  - Unit 3: Netra st D1000 Storage Array
  - Unit 4: Netra st D130 Storage Enclosure
- Topic 4: Gigabit Ethernet Support for Public Networks

Module 4: New Data Service Support

- Topic 1: New Supported Data Services
- Topic 2: Installation and Configuration of Sun Cluster HA for SAP
- Topic 3: Sun Cluster HA for Sybase ASE 12.0 (32-bit)
- Topic 4: Oracle Parallel Server (OPS) 8.1.7 (32-bit)
- Topic 5: Sun Cluster HA for Oracle
- Topic 6: Sun Cluster HA for Netscape LDAP

Module 5: New Supported Software Revisions

- Topic 1: Overview of New Supported Software Revisions

- Topic 2: Patching Requirements for New Supported Software Revisions
- Topic 3: Applying Patches to an Existing Cluster
- Topic 4: Monitoring a Cluster with Sun Management Center 3.0

Module 6: Installation of Sun Cluster 3.0 7/01

- Topic 1: Changes to the Existing Sun Cluster 3.0 Installation Process
- Topic 2: Installing a Cluster Using SunPlex Manager
- Topic 3: Upgrading a Cluster from Sun Cluster 3.0 to Sun Cluster 3.0 7/01 (Update 1)

Module 7: Useful Sources of Sun Cluster 3.x Information

Registering for WZI-2747

You may register for this or any other ITT Web Learning Course (WLC) at:
`http://education.central/ITT/wlc.html`

You will get a Registration Notification back from the wlc registration system with your login information. The course is listed under the ITT Curriculum Bundle.