



# SiCortex<sup>®</sup> System Administration Guide

SC5832, SC1458, SC648

---



For Software Version 3.0

---

### ***Trademarks***

SiCortex is a registered trademark, and the SiCortex logo, SC5832, SC648, SC072 and FabriCache are trademarks of SiCortex, Incorporated.

PathScale is a wholly owned subsidiary of SiCortex. All PathScale trademarks are the property of SiCortex.

Linux is a registered trademark of Linus Torvalds.

MIPS and MIPS64 are registered trademarks of MIPS Technologies, Inc.

Lustre is the registered trademark of Cluster File Systems, Inc.

NIST is a registered trademark of the National Institute of Standards and Technology, U.S. Department of Commerce.

OpenMP is a trademark of Silicon Graphics, Inc.

PCI, PCI Express and PCIe are registered trademarks, and PCI ExpressModule is a trademark, of PCI-SIG.

Perl is the registered trademark of The Perl Foundation

TAU Performance System is a trademark of the joint developers: University of Oregon Performance Research Lab; Los Alamos National Laboratory Advanced Computing Laboratory; and the Research Centre Jülich, ZAM, Germany.

TotalView is the registered trademark of TotalView Technologies, LLC.

Vampir, Vampir-NG, VampirServer, VNG Server, and Visualization Client vng are registered trademarks of Wolfgang E. Nagel.

InfiniBand is the registered trademark of the Infiniband Trade Association.

Intel is the registered trademark of Intel Corporation.

All other trademarks are the property of their respective owners.

### ***Copyrights***

Copyright© 2008 SiCortex Incorporated. All rights reserved.

### ***Disclaimer***

The content of this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by SiCortex, Inc.

**Part Number:** 2905-03 Rev. 01

Published September 19, 2008

## About this Guide

This book provides the full range of information and procedures needed to administer a SiCortex system.

### **For the SC5832, SC1458, and SC648 Only**

The procedures in this guide apply only to the SC5832, SC1458, and SC648.

For all system administration procedures for the SC072-PDS, refer to the SC072-PDS documentation.

## Getting Technical and Customer Support

### **Contacting SiCortex**

For technical and customer support, contact SiCortex Customer Support at:

**Toll free number:** +1 877 742 6783 x289 or +1 877 SICORTE x289

**Main phone number:** 978-897-0214

**Email:** [support@sicortex.com](mailto:support@sicortex.com)

### **SiCortex Website**

The SiCortex website provides information about the company and its products:

**Website:** [www.sicortex.com](http://www.sicortex.com)

## Finding Documentation

SiCortex provides the following information resources:

## Books

All books listed in this section can be found installed on the SSP in one of these two directories:

- `/opt/sicortex/doc/hardware`
- `/opt/sicortex/doc/software`

### Accessing Apache web server

You can access any of the books installed on the SSP, using the installed Apache web server. From any machine, open a web browser and supply the IP or hostname assigned to the SSP. You see a web page with links to the documentation.

For example, if your SSP were `ssp016`, you would use this URL:

```
http://ssp016.sicortex.com/
```

Enter the URL in the web browser, and you see a page that provides links to the PDF versions of all the manuals listed below.

If this doesn't work, you probably need to restart the Apache web server. At a command prompt on the SSP, type:

```
/etc/init.d/apache2 restart
```

### Site Preparation Guides

The *Site Preparation Guides* provide information on the system's requirements for space, power, cooling, move-in facilities, and other information required to plan in advance for your SiCortex system.

- *SC5832 Site Preparation Guide*
- *SC648 Site Preparation Guide* (covers the SC1458 as well)

The SC072-PDS is a desktop model and does not have a separate *Site Preparation Guide*.

### Hardware Installation Guides

The *Hardware Installation Guides* describe how to unpack, assemble, and install your SiCortex system.

- *SC5832 Hardware Installation Guide*
- *SC648 Hardware Installation Guide* (covers the SC1458)

### Quick Start Guides

- *SC072-PDS Catapult Quick Start Guide*

***System Administration Guide***

The *System Administration Guide* provides information on how to boot and configure the system, how to manage user accounts and system resources, how to configure storage, and other issues of interest to the system administrator.

***Programming Guide***

The *Programming Guide* describes the programming environment on a SiCortex system. It provides information on how to compile programs, both native compile and cross-compile, how to debug programs, how to run programs, how to use performance tools and optimize application performance, and other information about running applications.

***man Pages***

man pages are available on the SiCortex system for all standard Gentoo Linux commands, and for all special-purpose SiCortex commands and utilities as well.

On a conventional Linux system, there is a single system and all the software is installed on it. In such a normal Linux system, there are man pages describing things that are installed on that system (programs, libraries, and so forth), and you access them with the `man` command.

SiCortex system systems are more complicated. Even though in the literal sense all the software lives on the SSP, there are actually three conceptually separate systems in a SiCortex system, each with its own set of programs and corresponding man pages:

- SSP (System Service Processor)
- nodes
- MSP (Module Service Processor)

In some cases there is more than one man page for the same command or item, because that item resides in more than one system.

For example, the `ls` command is installed on all three systems (though the MSP one may not have a man page installed to go with it).

If you type `man <command-name>`, it displays the man page for `<command-name>` on the current system, whatever system that may be.

Commands and other items that are unique to a system have only one man page. For example, SSP system administration tools reside only on the

SSP, and therefore they have `man` pages that are accessible only on the SSP.

However, some commands and items reside on more than one of the three sub-systems. Therefore, if you type `man ls` from the SSP, you get the `man` page for the `ls` program that's installed on the SSP. (If you type `man ls` from a node, you get the `man` page for the `ls` program that's installed on the nodes, which may not be exactly the same version).

If you can't find a `man` page in one location, try the others.

### **SSP and MSP man pages**

You can access SSP and MSP `man` pages only from the SSP. Since SSP and MSP programs are administrative in nature, only the system administrator needs access to those `man` pages.

For example, to see the SSP version of the `ls` `man` page, type the usual command on the SSP:

```
man ls
```

### **Node man pages**

You can access the `man` pages for node programs in *two* ways:

- From a node: By typing `man <command-name>`
- From the SSP: By typing `scman <command-name>`

#### **scman**

The `scman` command allows you to be on the SSP, but to look at `man` pages that reside elsewhere.

Type `scman <command>` on the SSP to see:

- `man` pages on the nodes
- `man` pages on the MSPs

### **MSP man pages, from the SSP only**




The MSP `man` pages are visible only on the SSP. The MSP is too space-limited to install the MSP `man` pages on the MSP itself.

If you can't guess where a `man` page should be, try both locations.

`man` and `scman apropos` works for SSP `man` pages.

Use `scman -k <string>` to find the `man` pages that might be relevant to `<string>`.

# Manual Conventions

- Italics* Identifies a term or a cross reference.
-  Indicates a tip, hint, or reminder.
-  Indicates a caution, such as a dependency that must be satisfied before continuing a process.
-  Indicates a warning, such as danger of injury or damage to equipment.







# Table of Contents

---

About this Guide .....	iii
For the SC5832, SC1458, and SC648 <b>Only</b> .....	<b>iii</b>
Getting Technical and Customer Support.....	iii
Contacting SiCortex.....	iii
SiCortex Website .....	iii
Finding Documentation.....	iii
Books .....	iv
Manual Conventions.....	vii
<b>Chapter 1 — SiCortex System Description .....</b>	<b>1</b>
Administering a SiCortex System .....	1
SiCortex System Architecture .....	2
Tour of the Nodes .....	4
The Interconnect Fabric .....	5
Placeholder Card Configurations .....	6
Software on the SiCortex Nodes.....	8
System I/O.....	9
The SSP — the Management Appliance .....	9
Protecting the SSP from User Logins .....	10
System Hardware .....	10
SSP .....	11
Environment Sensors and Controls.....	11
Intracabinet Networking .....	11
SSP Software.....	13

---

Overview of MSP .....	15
<b>Chapter 2 — System Software Concepts .....</b>	<b>19</b>
Concepts—Installing, Configuring and Booting .....	19
Concept—New System Configuration and Boot .....	19
Concept—SSP Ships with Software Pre-installed .....	19
Concept—Log of Booted SSP Software Version .....	20
Concepts—Security on the SiCortex System .....	20
Concept—Using iptables as the SSP Firewall .....	20
Concept—Root Passwords on the SSP and Nodes .....	21
Concept—SSP Access for SysAdmin Only .....	21
Concept—Document Configuration Changes .....	21
Concepts—What to Back up, and Why .....	22
Concept—Back up Configuration Files You Edit.....	22
Concept—Back up the Node Root File System.....	23
Concepts—IP Addressing on the SiCortex System.....	23
Concept—Names and Addresses .....	23
Concept—Choosing Netblock in GRUB Screen .....	23
Concept—Time Zones on the System .....	24
Concept—SSP Networking .....	25
Concept—Choosing DHCP (Default) or Static Addresses .....	26
Concepts—Customizing and Configuring Your System .....	26
Concept—Configuring System Parameters .....	26
Concept—Use sicortex-system.conf.example.....	27
Concept—Basic Cluster and Boot Parameters.....	27
Concepts—Root File System.....	28
Concept—Two Copies of Root File System on SSP .....	28
Concept—Serving the Node Root File System.....	29
Concept—Default Methods for Serving the Root File System .....	29
Concept—Modifying the Root File System.....	30
NFS Provides Read/Write Rootfs on SC648.....	31
NBD Rootfs Is Read-Only on SC1458 and SC5832.....	33
Concepts—Modifying Kernel Behavior.....	34
Concept—Kernel Log Level .....	34
Concept—Node Kernel Arguments .....	35
Concepts—Nodes as Network Devices .....	35
Concept—Configuring Nodes for Networking .....	35
Concept—Head Node .....	38
Concept—Router Nodes .....	39
Concept—Configuring a Default Router .....	39
Avoiding Routing Problems.....	40
Concept—Configuring Network Interfaces .....	41
Concept—Mapping the Interconnect Fabric .....	42
Concept—Declaring Placeholder Cards.....	44

Concept—Declaring Disabled Nodes, Modules, Links .....	45
Debugging Disabled Nodes and Links.....	47
Concept—Overview of Rebooting the System .....	49
<b>Chapter 3 — Configuring and Booting the System.....</b>	<b>51</b>
Steps for Configuring and Booting the System .....	51
.....	52
Installing a New Release on an Existing System .....	52
Configuring and Booting a New System.....	52
Where to Find the Background Concepts .....	52
Step 1—Prepare to Boot the System .....	52
1a—Allocate Names and Addresses .....	52
1b—Verify System Has Power .....	53
1c—Document Your Configuration Changes.....	53
Step 2—Boot the SSP.....	53
2a—Connect a Keyboard and Monitor to the SSP on a New System .....	53
2b—Boot the SSP .....	54
Step 3—Change Root Password on SSP and Nodes .....	55
3a—Change the SSP Root Password .....	55
3b—Change the Node Root Password.....	55
Step 4—Back up the Node Root File System .....	56
Back up the Node Rootfs .....	56
To Unpack the Backup Rootfs Later.....	57
Step 5—Set Your Time Zone.....	57
EST Is the Default Time Zone .....	57
5a—Checking Time Zone Files .....	57
5b—Change the SSP Time Zone .....	58
5c—Prevent Clock Inconsistency on the SSP.....	58
5d—Change the Nodes Time Zone.....	59
5E—Change the MSPs Time Zone.....	59
Guidelines—Editing Configuration Files.....	60
Step 6—Configure SSP Networking .....	60
6a—To Use DHCP.....	62
6b—To Use Static Addresses .....	62
Step 7—Verify That Basic SSP Networking Works.....	63
Step 8—Configure SSP Firewall.....	65
Step 9—Configure System and Networking Settings .....	67
9a—Create Your Own sicortex-system.conf .....	67
9b—Edit the File Header .....	68
9c—Serving the Root File System to the Nodes.....	69
9d—Set the Kernel Log Level for Booting.....	69
9e—Set Additional Node Kernel Arguments .....	70
9f—About Configuring Networking.....	70
9g—Configure a Head Node.....	71

9g—Configure Router Nodes.....	72
9h—Configure a Default Router.....	73
9i—Configuring Extra Routes.....	74
9j—Configure Network Interfaces for Exterior Nodes.....	75
9k—Specify Routing Mode: yes or nat.....	76
9l—Configure Network Interfaces.....	77
9m—Declare Placeholder Cards.....	78
Example—SC5832, 20 Processor Modules.....	80
9n—Declare Disabled Nodes, Modules, Links.....	81
9o—Restoring Disabled Components.....	82
Step 10—Edit Software Installation Settings.....	83
10a—Do Not Edit the File Header.....	83
10b—Using a Non-Default Root File System Image.....	84
10c—Setting a Different Default Kernel.....	84
Step 11—Boot the Nodes.....	85
Booting the Nodes.....	86
Step 12—Test the Head Node.....	89
Step 13—Verify Networking Works on the Nodes.....	90
Step 14—Add User Accounts.....	90
Step 15—Reboot the SSP and Nodes and Test All Changes.....	91
Variations on Booting the Nodes.....	91
Default Kernel Directory.....	92
Booting the Kernel from a Non-default Directory.....	92
Booting a Non-Default File System.....	93
Checking Release and Component Versions Used by sbboot.....	93
Rebooting a Single Node.....	93
Executing Custom Boot Scripts on the Nodes.....	93
<b>Chapter 4 — Installing Software Releases.....</b>	<b>95</b>
Checking Which Release Is Currently Installed.....	95
Rolling Your Custom Configuration Data Forward.....	95
Frequently Customized Files.....	96
Getting Access to a New Release.....	97
Delivery Methods for New Software Releases.....	98
Layout of Release Software Files.....	98
Disk Partitions for Installing Releases.....	98
SSP Ships with Software on Partition 5.....	99
Resetting the Default Boot Partition.....	99
Installing a New Release.....	101
install-ssp Script.....	101
Methods for Installing a New Release.....	101
Installing on a Running SSP from the DVD.....	102
Installing on a Running SSP from the Web.....	103
Installing by Using the DVD to Boot First.....	104

Configuring and Booting with a New Release .....	105
First, Mount the Old Rootfs Partition .....	105
Next, Re-Execute These Configuration and Booting Steps.....	105
Summary of install-ssp Options .....	108
Sample install-ssp from Download Site .....	109
Resolving Conflicts When Installing Updates .....	109
About the n32 and n64 ABI Environments .....	111
<b>Chapter 5 — System Services .....</b>	<b>113</b>
Services Started by Default on the SSP.....	113
What Services Are and Are Not Running?.....	115
Services Started by Default on the Nodes .....	116
Removing a Default System Service.....	117
Notes on DNS Server .....	117
Running Services Under xinetd .....	117
<b>Chapter 6 — Providing Access to System Resources.....</b>	<b>119</b>
Primary System Resources .....	119
Using Nodes for Computing and I/O .....	120
Compute Nodes.....	120
I/O-Ready Nodes.....	120
Separating I/O Nodes from Compute Nodes.....	123
Why Segregate I/O Nodes?.....	123
Designating a Head Node.....	124
Configuring a Head Node .....	125
One Head Node Is Sufficient .....	125
Renaming Your Head Node.....	125
Creating and Using Partitions.....	126
Default SLURM Partitions as Shipped .....	127
Root Partition sc*, for Administrative Use .....	128
Computing Partitions Should Exclude I/O Nodes .....	129
Specifying a Default Partition.....	130
Viewing Existing Partitions .....	131
Defining Partitions .....	131
SLURM File Defines Partitions.....	132
Persistent Partition Changes in Conf File .....	132
Temporary Partition Changes with scontrol .....	132
Node Addressing.....	133
Node Naming Notation .....	134
<b>Chapter 7 — Managing User Accounts.....</b>	<b>137</b>
System Access and User Privileges.....	137
SSP for System Administrator Only .....	137
Users Should Log in to Head Node .....	138
Login Policies .....	138

SSP User Has Total System Access .....	138
Preparing to Configure User Accounts and Passwords .....	139
Brief Introduction to LDAP .....	139
System Uses LDAP on the SSP .....	140
Changing the LDAP Root Password .....	140
Four Methods to Manage User Accounts .....	140
Method 1 — Use an External LDAP Database .....	141
Configuring the LDAP Client, Part One .....	141
Configuring the LDAP Client, Part Two .....	142
Configuring the LDAP Client, Part 3 .....	144
Configuring the LDAP Server .....	145
Restart the LDAP server on the SSP .....	150
Check syncrepl operation .....	151
Reboot the Nodes .....	151
Changing Account Data in an External LDAP Database .....	151
Method 2 — Configure LDAP Stand-Alone .....	152
Restart the LDAP server on the SSP .....	155
Populate the LDAP Database on the SSP .....	155
Reboot the Nodes .....	157
Testing Your LDAP Configuration .....	157
Back up Your LDAP Database .....	159
Changing User Account Data in a Standalone LDAP Database .....	159
Changing Group Data in a Standalone LDAP Database .....	160
Method 3 — Use passwd Files for User Accounts .....	161
Creating User Accounts in /etc/passwd .....	162
Back up Your Passwd Files .....	163
Changing User Passwords .....	163
chrootfs .....	164
scpasswd Command .....	164
scuseradd Command .....	164
Method 4 — Use NIS on the SSP and Nodes .....	165
For the SSP .....	165
For the Nodes .....	166
Logging into the System .....	166
Always Use ssh Instead of telnet .....	167
Changing User Accounts—Command Summary .....	167
<b>Chapter 8 — Managing Storage—Overview .....</b>	<b>169</b>
NFS File Systems .....	169
Direct-Attached File Systems .....	169
Lustre File Systems .....	170
FabriCache File Systems .....	171
<b>Chapter 9 — NFS File Systems .....</b>	<b>173</b>
NFS File System on External Server .....	173

Mounting an NFS File System from the SSP .....	176
<b>Chapter 10 — Direct-Attached File Systems .....</b>	<b>177</b>
Configuring a Direct-attached File System .....	177
Using NFS Server on a SiCortex System .....	178
<b>Chapter 11 — Lustre File Systems.....</b>	<b>183</b>
Lustre Overview .....	183
Lustre File System with External Server.....	184
Lustre File System with SiCortex Nodes as Servers .....	185
Lustre File System Configuration Example .....	185
Creating the Lustre File System .....	186
Mounting the Lustre File System.....	190
Mounting a Lustre File System with a Script .....	191
Unmounting a Lustre File System with a Script .....	194
Unmounting a FabriCache File System .....	196
<b>Chapter 12 — FabriCache File Systems.....</b>	<b>197</b>
FabriCache—A Lustre File System on Nodes, Not Disks .....	197
Lustre on FabriCache .....	198
Requirements and Limitations .....	199
Setting up a FabriCache File System .....	200
fc_create Command .....	200
Usage Notes .....	201
fc_destroy Command .....	203
FabriCache Example .....	204
Using a FabriCache File System .....	205
<b>Chapter 13 — Running and Monitoring Jobs .....</b>	<b>207</b>
Tips for Running SLURM Jobs.....	207
SLURM Job Logging Is On by Default .....	207
Turning SLURM Logging On or Off.....	207
When to Specify the Full Path for an Executable .....	208
Finding Out How Many Files a Job Has Open .....	208
Running SLURM MPI Jobs .....	209
How SLURM_SRUN_COMM_IFHN Works.....	209
SLURM_SRUN_COMM_IFHN Is Set to SSP by Default .....	210
Unset SLURM_SRUN_COMM_IFHN for Batch Jobs.....	210
Batch Job Fails If Variable Is Not Set Correctly .....	210
Raising Ulimits for SLURM Jobs .....	211
<b>Chapter 14 — Monitoring the System.....</b>	<b>213</b>
Overview of System Monitoring Features .....	213
Nagios Alerts.....	214
Log File Entries.....	214
How System Monitoring Is Implemented .....	215
Configuring System Monitoring .....	216

System Monitoring Files .....	216
Specifying Your Nagios Server to Receive Alerts .....	217
Environmental Data Logged to policyd.log .....	218
Raw Environmental Data in RRD Databases .....	219
Locating Log Files .....	219
Log File Handling .....	219
Tailing Any Log File .....	220
Log Files by Purpose and Location .....	221
Other SiCortex Log Files .....	223
<b>Chapter 15 — Checking System Status .....</b>	<b>225</b>
Checking Performance on a Node .....	225
Checking Performance System-Wide .....	226
<b>Chapter 16 — Managing Installed Software .....</b>	<b>227</b>
Gentoo Packaging Concepts .....	227
Introduction .....	227
Portage and Portage Files .....	228
Portage Commands .....	228
Ebuilds, Portage Tree and Overlays .....	229
Ebuild Versions and Selection Criteria .....	230
Searching Packages .....	231
Installing and Customizing Packages .....	232
Querying Packages .....	236
Managing Packages on a Running Node .....	238
Removing Packages .....	238
Installing the n32 ABI Environment .....	239
Installing the Buildroot for the n32 ABI .....	241
Making the n32 ABI Available on the Nodes .....	241
Licensing Third-Party Software .....	242
Using TotalView .....	242
Using Vampir .....	243
<b>Chapter 17 — Installing the Cross-Development Toolkit .....</b>	<b>245</b>
Overview of the Cross-Development Environment .....	245
Installing the Software .....	246
Updating the Cross-Development Software .....	249
<b>Chapter 18 — Shutting Down and Rebooting the System .....</b>	<b>251</b>
Manual System Shutdown .....	251
Rebooting after Manual Shutdown .....	252
Automatic Emergency System Shutdown .....	254
After an Automatic Emergency Shutdown .....	254
Temperature Monitoring Details .....	255
Voltage Monitoring .....	255
<b>Appendix A — Release Profile .....</b>	<b>257</b>



Current Release: Version 3.0 .....	257
Previous Release: Version 2.2 .....	258
Previous Release: Version 2.1 .....	258
<b>Appendix B — Licenses.....</b>	<b>259</b>
<b>Appendix C — Directories and Files.....</b>	<b>261</b>
Location of Files on the SSP .....	261
Status of /var/state Files Across Releases .....	263
Shared across releases, and user editable:.....	263
Shared across releases, created by install: .....	263
Shared across releases, created by system daemons / sboot: .....	263
Finding Nodes .....	264
Node Addressing.....	264
MSPnet Addressing.....	265
I/O-Ready Nodes.....	267
<b>Appendix D — Common Procedures.....</b>	<b>269</b>
Logging into the System.....	269
Logging into the SSP as root.....	269
Logging into the SSP .....	269
Logging into the Head Node as a Non-root User.....	270
Using chrootfs to Edit the Root File System.....	270
Using chrootfs .....	271
Using Portage on a Running Node (chrootfs).....	273
chrootfs Command .....	273
Related Tools .....	274
.....	275
<b>Appendix E — Customized System Files— Checklist.....</b>	<b>277</b>
Backing up Your Modified System Files.....	277
<b>Appendix F — sicortex-system.conf.example.....</b>	<b>281</b>
Applies only to the SC5832, SC1458, and SC648.....	281



# Chapter 1 SiCortex System Description

This introduction provides an overview of administering a SiCortex system, and of the SiCortex system architecture, hardware, and software. Details on the concepts presented in this chapter are provided in the rest of this book. In this chapter:

- Administering a SiCortex System
- SiCortex System Architecture
- The SSP — the Management Appliance
- System Hardware
- SSP Software
- Overview of MSP

Built to support the dominant High Performance Technical Computing (HPTC) software model, the SiCortex system with its MPI/Linux software suite empowers users to quickly develop applications that can tackle the most complex and computationally intensive problems that face the scientific, engineering, and financial communities.

## Administering a SiCortex System

**About this book** This book seeks to provide all the information you need as system administrator, to run a SiCortex system. The chapters in this book provide information on all your areas of responsibility as system administrator:

- SiCortex System Description
- System Software Concepts
- Configuring and Booting the System
- Installing Software Releases
- System Services
- Providing Access to System Resources
- Managing User Accounts
- Managing Storage—Overview
- NFS File Systems

- Direct-Attached File Systems
- Lustre File Systems
- FabriCache File Systems
- Running and Monitoring Jobs
- Monitoring the System
- Checking System Status
- Managing Installed Software
- Installing the Cross-Development Toolkit
- Shutting Down and Rebooting the System

The rest of this chapter provides an overview of the SiCortex system’s architecture, hardware, and software.

## SiCortex System Architecture

As the administrator of a SiCortex system, you spend most of your time logged into the System Service Processor (SSP). The SSP is the management appliance that lets you configure, control, and monitor the nodes. Application users log into the *head node* (sometimes called a *login node*) to run jobs on the nodes.

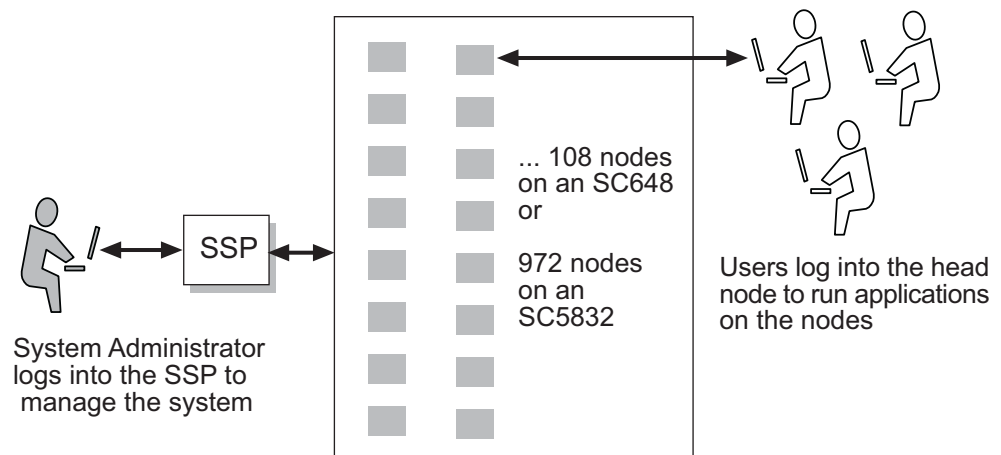


Figure 1. Sysadmin controls SSP—users log into head node

**Nodes** A SiCortex system consists of a number of six-way, symmetric multiprocessing (SMP) compute nodes connected by an Interconnect Fabric. The number of nodes varies with the size of the system.

**SSP - management appliance** You manage the system with the System Service Processor (SSP). On larger systems such as the SC1458 and SC5832, this is a separate server mounted in the same cabinet.

Figure 2 shows how a SiCortex system is laid out, and how it connects to other entities on your network.

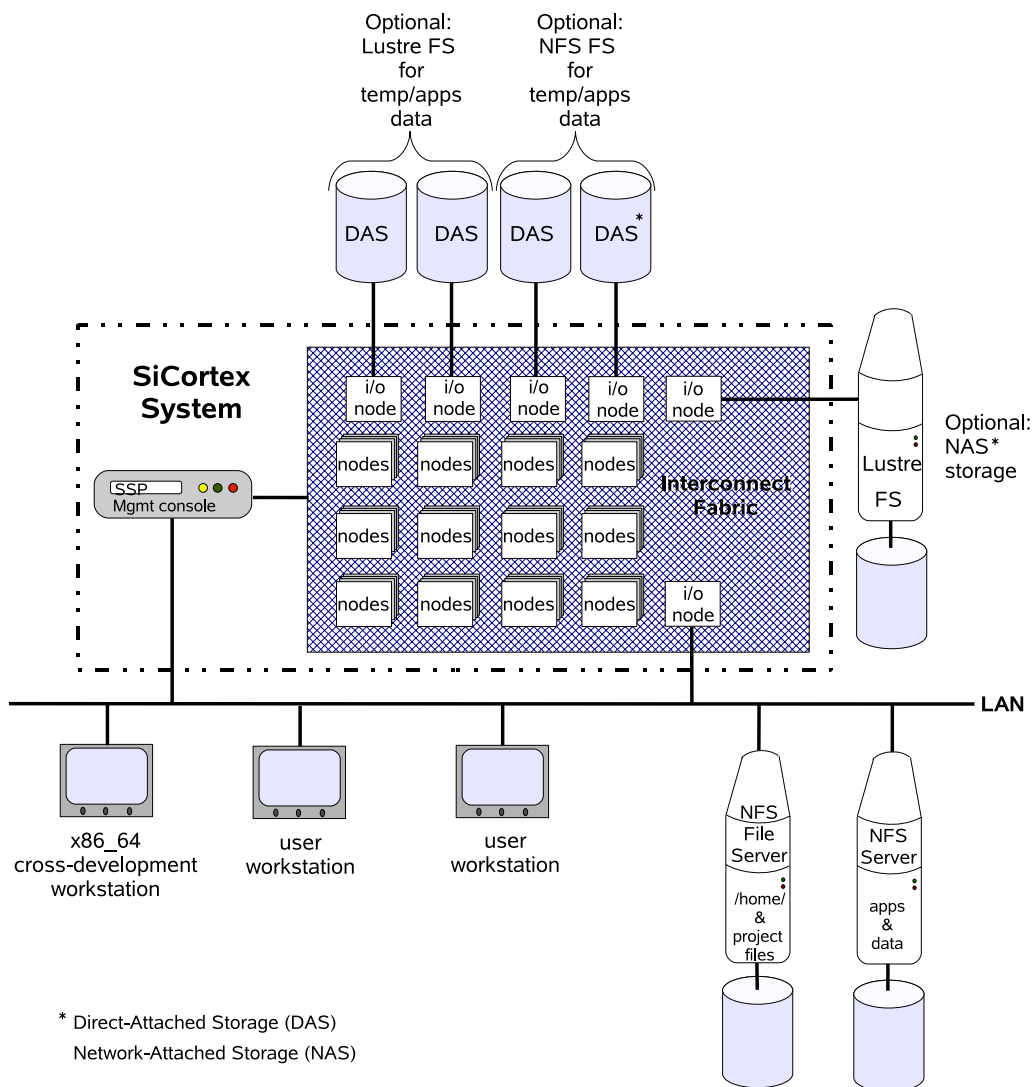


Figure 2. Overview of a SiCortex system

## Tour of the Nodes

Each node consists of one SiCortex node chip (Figure 3 on page 4) and two industry-standard DDR2 memory modules. A node chip contains six 64-bit processors, their L1 and L2 caches, two memory controllers (one for each memory module), the Interconnect Fabric interface components (the Fabric Links, the Fabric Switch, and the DMA engine), and a PCI Express<sup>®</sup> (PCIe<sup>®</sup>) interface. The PCIe controller provides control for external I/O devices only, not for the Interconnect Fabric.

On the node chip, the DMA engine, Fabric Switch, and Fabric Links provide the interface to the Interconnect Fabric. The DMA engine connects the memory system to the Fabric Switch, which forwards traffic between incoming and outgoing links, and to and from the DMA engine.

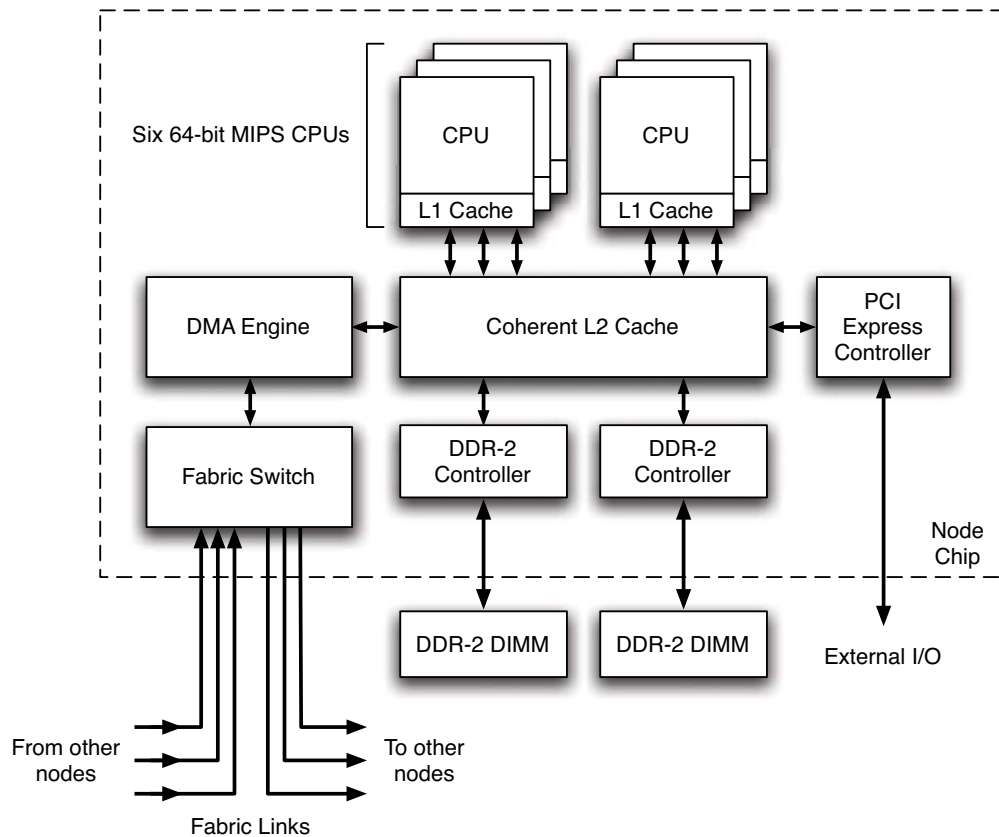


Figure 3. Overview of SiCortex node internals

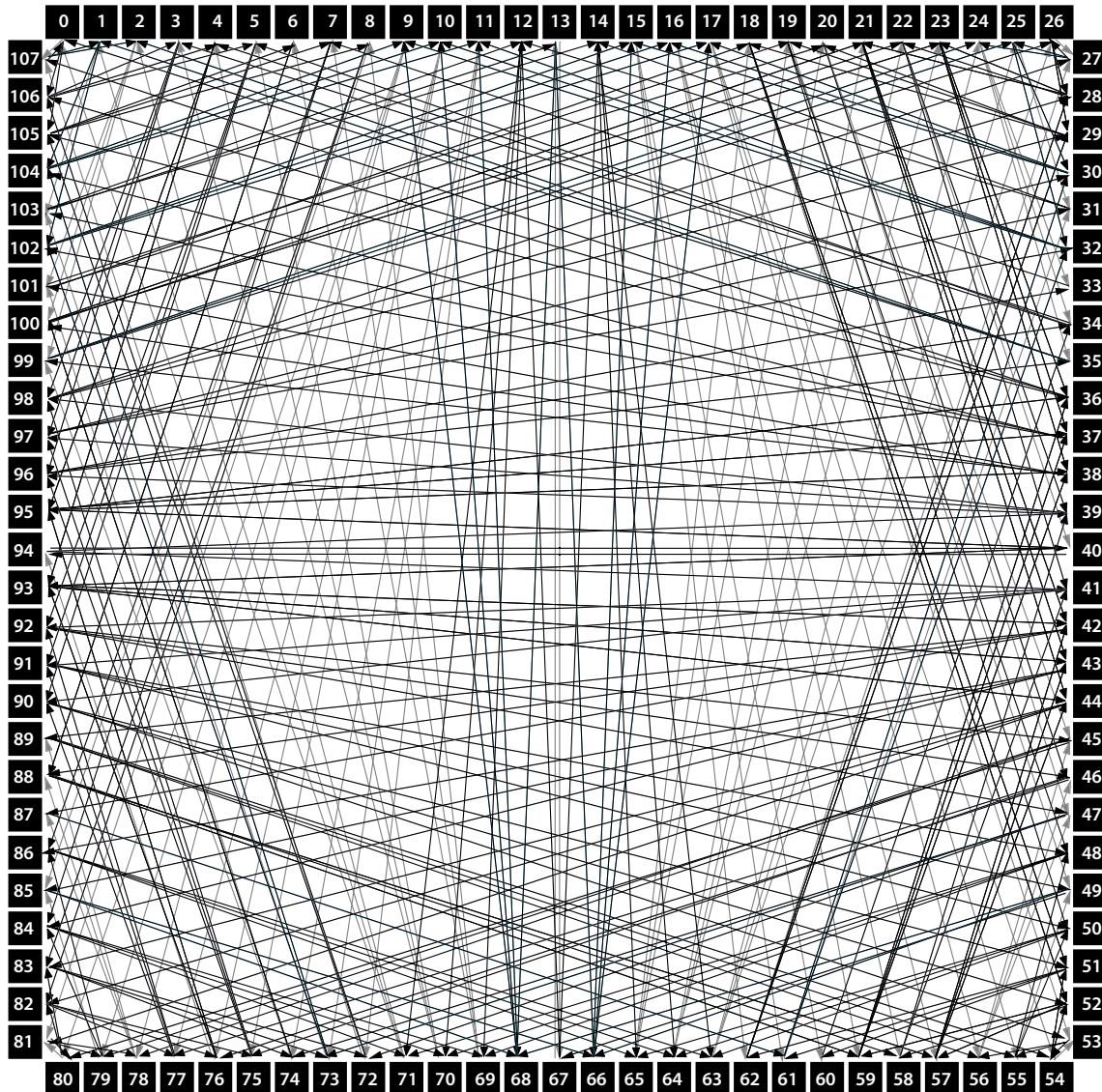
On an SC5832 or SC1458 fully populated with processor modules, all nodes in a system are connected through a degree-3 directed Kautz network. The Kautz graph ensures that every node can communicate with every other node in a small number of hops. Twenty-seven nodes populate a module, and all modules plug into the system's backplane. Of the

twenty-seven nodes on a module, three have their PCIe busses connected to PCI ExpressModule™ slots, and a fourth is attached to an on-module PCIe dual gigabit-Ethernet controller. The PCIe interface on all other nodes is disabled. The nodes on a fully populated SC1458 are connected in a de Bruijn graph rather than a Kautz graph.

## The Interconnect Fabric

The nodes communicate with one another through DMA over the Interconnect Fabric, a fast network used for internode IP networking and direct user-mode communications. The network is based on a degree 3 Kautz graph (Figure 4 on page 5).

Figure 4. 3-degree directed Kautz network for an SC648 system



This network enables internode messages to arrive at their destination within a maximum number of hops. For 108-node systems, the maximum hop count is four, and for 972-node systems, the maximum hop count is six.

Each node transmits to three other nodes and receives from a different three nodes. Not only does this design reduce message latency and network congestion, it also ensures that the failure of one node increases the hop count of a message by no more than one, and that all other nodes remain reachable.

For detailed information on the Kautz graph, see *A New Generation of Cluster Interconnect* posted on the SiCortex web site at <http://www.sicortex.com/>.

The following table displays the number of nodes for a given degree and diameter of Kautz graph.

Table 1. Number of Nodes in Kautz Networks

<b>Diameter</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>
Degree 2	6	12	24	48	96	192
Degree 3	<b>12</b>	36	<b>108</b>	324	<b>972</b>	2916
Degree 4	20	80	320	1280	5129	20480

The diameter of the network (the greatest number of hops a message must take from source to destination) is proportional to the logarithm of the number of nodes in the system. This property results in very small network diameters.

The diameter of the SC5832 is six for 972 nodes, compared with a diameter of at least 15 for a 3-D torus of 1000 nodes, while using half as many links.

Note that the SC1458 is uses a degree 3 diameter 5 de Bruijn graph.

The SC648 has a degree-3 Kautz graph with diameter 4, and therefore has 108 nodes.

## Placeholder Card Configurations

For customers who anticipate increasing computing needs, SiCortex provides systems that contain a subset of the full number of processor modules. The remainder of the slots are filled with *placeholder cards*.



Placeholder cards complete the interconnect path between processor modules in systems that contain fewer than the standard number of processor modules.

The nodes in a system with some processor modules and some placeholder cards are connected to one another via a graph that is neither a classical Kautz graph nor a classical de Bruijn graph, and may have a larger variability in the maximum number of hops between nodes.

For a SiCortex system to function, all slots in it must contain a processor module or a placeholder card. You can replace the placeholder cards with processor modules, as your needs grow for computing power.

Except for the fact that placeholder cards are not cabled and must be installed in certain slots, you install them the same way you install processor modules, as described in the *Hardware Installation Guide*.

**Placeholder Card Slots** For each SiCortex system model, only specific processor module/placeholder card configurations are supported.

**SC5832** The SC5832 supports processor module/placeholder card configurations of 28/8, 20/16, and 12/24 only.

Table 2. SC5832 slot assignments in supported configurations

Processor Modules	Install in these slots...	Placeholder Cards	Install in these slots...
36	Front: All Back: All	0	None
28	Front: 2, 4, 6, 8, 10, 12, 16, 18, 20, 22, 24, 28, 30, 32, 34 Back: 1, 5, 7, 11, 15, 17, 21, 23, 25, 27, 29, 33, 35	8	Front: 0, 14, 26 Back: 3, 9, 13, 19, 31
20	Front: 0, 6, 10, 12, 14, 20, 30, 32 Back: 1, 5, 7, 11, 13, 15, 17, 23, 27, 31, 33, 35	16	Front: 2, 4, 8, 16, 18, 22, 24, 26, 28, 34 Back: 3, 9, 19, 21, 25, 29
12	Front: 6, 10, 20, 22, 24, 28, 30, 34 Back: 1, 7, 11, 27	24	Front: 0, 2, 4, 8, 12, 14, 16, 18, 26, 32 Back: 3, 5, 9, 13, 15, 17, 19, 21, 23, 25, 29, 31, 33, 35

**SC1458** On the SC1458, the following processor module/placeholder card configurations are available. In these configurations, all links are still used:

Table 3. SC1458 slot assignments in supported configurations

Processor Modules	Install in these slots...	Placeholder Cards	Install in these slots...
9	0, 1, 2, 3, 4, 5, 6, 7, 8	0	None
6	0, 2, 3, 5, 6, 8	3	1, 4, 7
4	0, 2, 3, 4	5	1, 5, 6, 7, 8
3	1, 4, 6	6	0, 2, 3, 5, 7, 8
2	1, 4	2	0, 2, 3, 5, 6, 7, 8
1	1	8	0, 2, 3, 4, 5, 6, 7, 8

**SC648** On the SC648, the following processor module/placeholder card configurations are available. In these configurations, all links are still used:

Table 4. SC648 slot assignments in supported configurations

Processor Modules	Install in these slots...	Placeholder Cards	Install in these slots...
4	0, 1, 2, 3	0	None
3	0, 1, 2	1	3
2	0, 2	2	1, 3
1	0	3	1, 2, 3

## Software on the SiCortex Nodes

Each node in the SiCortex architecture is a six-processor SMP (symmetric multiprocessor) system running a single Linux kernel. The system software installed on the nodes includes the conventional utilities available on any Linux system, such as shells (**bash** and **tcsh**), scripting languages (Perl, Python, Tcl/Tk), compiler tools, and libraries.

**ABI's: n32 and n64** The SiCortex nodes support both 32-bit and 64-bit execution modes, implemented by Application Binary Interfaces (ABIs); the compilers can produce object code for either mode. The 32-bit mode is limited to 2 GB. The 64-bit mode, with a 64-bit pointer representation, is the default. It allows process virtual memory sizes that are vastly larger than 2 GB. The 32-bit mode provides for slightly more efficient memory and cache usage with its smaller pointers. Both modes allow equivalent access to processor features, including 64-bit data registers.

**Gentoo Linux** The nodes run a version of Gentoo Linux built for 64-bit MIPS.

## System I/O

The following table summarizes the I/O capabilities on each SiCortex system:

System Size	8-lane PCIe I/O Ports	Gigabit Ethernet I/O Ports
SC5832	108	72
SC1458	27	18
SC648	12	8

These ports provide connections to external networking and storage systems. Storage systems may be either direct or network-attached. Standard direct attached Linux file systems (ext2, ext3, and others) are supported. Transparent access for non-I/O nodes is provided by Lustre and NFS.

With FabriCache™ enabled, a reserved portion of main memory functions as a parallel file system. Managed by Lustre logic, an integral part of each node's kernel, FabriCache provides memory-speed I/O for I/O intensive applications.

The FabriCache is configurable—you can configure a little memory or a lot, on a few nodes or many, to function as a parallel file system, accessible by all processors in the System. Lustre logic ensures data coherency and integrity, and the Interconnect Fabric ensures reliability of data transmission. Like disk file systems, the mechanics of accessing files in the FabriCache is transparent to users and requires no special programming.

For details on the SiCortex implementation of the Lustre files system, see Chapter 11, *Lustre File Systems* on page 183, as well as *The Lustre High Performance File System* white paper on the SiCortex web site at <http://www.sicortex.com/>.

For more information on the FabriCache, see Chapter 12, *FabriCache File Systems* on page 197, as well as *The SiCortex FabriCache™: Measure Its Abilities in Genomes/sec* newsletter on the SiCortex web site at <http://www.sicortex.com/>.

## The SSP — the Management Appliance

The SSP functions as a management appliance for the system. The SSP is a Linux computer that implements monitoring and control of power, cooling, booting, shutdown, and error logging and analysis. It also pro-

vides the management interface to the system, for use by the system administrator.

The following system management protocols are supported in this release:

- Command line access via SSH - The ability to use the command line to invoke a variety of utilities, scripts, and Linux commands.

The SSP collates alarm information from all subsystems and maintains a consolidated error log, accessible via the CLI or the SSP's syslog.

The SSP provides persistent storage for the initial boot image and for diagnostics. When shipped, the boot image is a basic Linux root file system and kernel, capable of bringing the system to operational status.

### **Protecting the SSP from User Logins**

SiCortex recommends that the SSP be shielded from logins by users other than the system administrator. You can accomplish this goal by any of these methods:

- User account access control
- Firewall software
- Wiring the physical network so that the SSP is on a management network which is separate from the user network

The system administrator should have sole access to the SSP. Users should be free to log in to the node designated as a head node but should not be able to log in to the SSP.

The head node is the node that the system administrator designates as the node users will log in to, to run jobs on the nodes. See *Designating a Head Node* on page 124 in Chapter 6, *Providing Access to System Resources* on page 119.

## **System Hardware**

This section summarizes the key points of the SiCortex hardware that are relevant to the software for managing a SiCortex system. More details about the SSP hardware and the internal wiring can be found in the *Hardware Installation Guide*.

## SSP

The SSP is an x86-64 system. The model of the SSP varies by system model:

System Model	SSP Model
SC648	Hewlett-Packard DL320 G5 The DL320 is a 1U unit w/2 500GB SATA drives. The drives can be configured for RAID0 or RAID1. The fans are redundant; the power supply is not.
SC1458	Same SSP as the SC648
SC5832	Hewlett-Packard HP DL380 The DL380 is a 2U unit with 2GB of memory and 8 146GB SAS drives configured for RAID5. The fans and power supplies are redundant.

## Environment Sensors and Controls

The system power supplies and fans are connected to the intracabinet Ethernet. In addition, the module service processors (MSPs) provide access to board-level and node-level temperature sensors and voltage regulators. Software on the MSPs is responsible for monitoring the temperature. The fan controllers control the fans to keep the system temperature in the proper range.

In extreme cases, each MSP can shut down its own module, so that together they shut down the whole system to avoid overheating, or damage from non-standard power conditions.

## Intracabinet Networking

Inside the cabinet are two physical Ethernets, called CTLnet and MGT-net.

The two other networks on the system, SCethernet and MSPnet, are virtual networks.

**MPI and IP networking** The family of SiCortex systems provides full support for both MPI and IP networking.

**SCethernet** SCethernet emulates an Ethernet over the interconnect fabric. It provides high-speed, high-bandwidth TCP/IP communication among all the

nodes on the system. SCethernet addresses the nodes using the `sc*-mynz` addressing scheme, explained in "Node Addressing" on page 133.

**MSP** Each module in the system has a Module Service Processor (MSP). The MSP is a microprocessor resident on the module that initializes the fabric and SCethernet and monitors the module and the fabric. For more information about the MSP, see "Overview of MSP" on page 15.

**MSPnet** MSPnet is a low-bandwidth IP network that connects each node on a module to the MSP on that module. MSPnet is separate from, or out-of-band with respect to, SCethernet. MSPnet is used to initialize the fabric and SCethernet, and to monitor the fabric.

Every processor module has a cable in the MSP port that connects it via MSPnet to the SSP.

**CTLnet** CTLnet is the Ethernet network that includes the devices other than the compute nodes: the MSPs, the system power supplies, and the fan trays. It is built around a 10/100 Ethernet switch with two gigabit ports. The gigabit ports connect to the SSP. CTLnet is used for routing to nodes via the MSPs. This is an IP network and is used for critical logging, monitoring, booting, and control of nodes; it is not used for routine operations.

**MGTnet** MGTnet on the SSP is the Ethernet interface for connecting to a set of nodes using the built-in Ethernet interfaces on the CPU boards. This interface is a quad network interface on an SC5832, and a dual network interface on an SC648. One gigabit Ethernet port connects directly to a single SC648 system.

The number of processor modules with direct connections to the SSP via MGTnet varies with system size. On an SC648, Module 0 Port 2 connects via MGTnet to Port B on the back of the SSP. On an SC5832, on Modules 0, 2, 4, and 6, Port 2 connects to Port A or Port B on the SC5832's SSP.

MGTnet is used for provisioning `root+usr` file systems, access to `root+usr` file systems for modification, system logging, performance data, LDAP directory queries, administrative logins and, generally, the majority of the communication between the SSP and nodes.

**Site LAN** The SSP also has its own direct network connection to the site LAN. This is also an Ethernet network. There may be one or multiple connections to the site LAN. These interfaces provide remote management access to the SSP and to the system as a whole. Normally, users communicate with the system by connecting through IO-ready nodes attached to the site LAN.

For more information about configuring nodes as network devices, see "I/O-Ready Nodes" on page 120.

**Site I/O network** The site I/O network connects the SiCortex system to the outside world.

## SSP Software

The SSP is a Linux server that comes pre-installed with a full, operational installation which includes the base SSP operating system, diagnostics, management software, and kernel and file system images for booting the module service processors (MSPs) and the nodes. For more information on the MSPs, see "Overview of MSP" on page 15.

**SSP Linux** The SSP runs Gentoo Linux built for x86-64, for reasons of familiarity, supportability, and stability. The HP ProLiant server support software is included in addition to the SiCortex provided packages. The hardware is shipped with all of the SSP software pre-installed. In addition, the SSP ships with its original materials, including software disk and HP documentation.

In addition, the package will include a set of DVDs for software installation including the basic Gentoo Linux DVDs, the SiCortex customizations, additional HP and SiCortex software, and a document describing the re-installation process should that be necessary.

**SSP firewall** [Not fully supported in version 3.0] The SSP uses the Linux *iptables* subsystem to restrict access to most of the services used for managing the system. In general, the firewall only permits `ssh` and web access by HTTP and TLS/SSL. Other services described here are specifically blocked from outside access. In addition, the default rule is to block traffic that is not specifically permitted.

**Directory services (LDAP)** The SSP runs an LDAP directory server that contains much of the configuration information used for the system, as described in more detail below. The LDAP server is also the primary source of data for managing user accounts.

**DNS** The SSP runs `dnsmasq` to provide DNS information to the system. The DNS server on the SSP provides the address lookup for those nodes, reverse lookup for their IP addresses, and forwards DNS queries from nodes to other DNS servers as appropriate.

The master host file configuration for the DNS server is stored in a flat file at `/var/state/hosts`, and is read by `dnsmasq`. This file is re-generated as needed.

**SLURM for job management** The SiCortex system uses SLURM, the Simple Linux Utility for Resource Management, from Lawrence Livermore Laboratory, for managing resources and running jobs. This guide refers to a number of SLURM commands, which are described in detail in the standard SLURM online documentation.

For more information about SLURM, see:

<http://www.llnl.gov/linux/slurm/>

**System logging (syslog)** The SSP runs `syslog-ng` and `conserver` for logging most events in the system. This includes:

- Node syslog events - By default, syslogging on the nodes is enabled
- Console logs (one per node)
- MSP syslog events
- Environment events forwarded to syslog by the monitoring daemon
- SSP local events

Normally nodes log over SCethernet to MGTnet, to the SSP. Critical events, such as those concerning the fabric not working properly, are logged over MSPnet/CTLnet.

For details on where to find files, see "Location of Files on the SSP" on page 261, and "Locating Log Files" on page 219.

**Event Manager** The event manager runs on the SSP to receive system events, including the syslog stream, and takes action based on individual events as well as detected trends. Events include exceeding resource limits, loss of communication with other system components, hardware faults, and so on.

Based on analysis of the events, the event manager may take action such as notifying the system operator, displaying a special alert, or forwarding an alert to an external monitoring system.

**DHCP** The SSP runs a DHCP server that provides IP addresses and initial configuration information to the devices on CTLnet and MGTnet. In particular, the DHCP server provides MSPs with the name of the boot image they are to load.



**Network Time Protocol (NTP)** The SSP runs a Network Time Protocol (NTP) daemon, for serving time information to nodes. By default, the NTP server on the SSP is set to synchronize with pool.ntp.org.

**TFTP** The SSP runs an internal TFTP server to provide the MSPs with access to the following:

- MSP kernels
- Node kernels
- Diagnostics
- Configuration information related to the above

The TFTP server will not respond to requests from outside the SiCortex system. Files for the TFTP server are kept in `/tftproot`.

**sshd** The SSP runs `sshd` (`openssh`) for remote access.

**scconserver and conserver** The `scconserver` and `conserver` servers work together to connect to MSPs and to write the console log files.

**Apache web server** The SSP runs an Apache web server. It serves two purposes:

- Copies file images to the nodes. When you boot in NBD mode, the root file system images are propagated to the nodes using the Apache web server.
- Provides access to the entire set of system documentation. The PDF files for all books can be read on the SSP using the Apache web server.

## Overview of MSP

This section introduces the Module Service Processor (MSP).

**Control plane** The control plane of the SC648, the SC1458, and the SC5832 includes those system components responsible for the power subsystem, environmental monitoring (particularly temperature), system booting, and system console. The control plane includes:

- The System Service Processor (SSP)
- The Module Service Processors (MSP)

Each CPU module includes a Module Service Processor in addition to its 27 CPU nodes.

**Control network** The control network, CTLnet, is an Ethernet that interconnects the SSPs and MSPs, power system, and cooling system, using an Ethernet switch mounted in the cabinet.

**Module Service Processor** The Module Service Processor, or MSP, is a microprocessor resident on each module (CPU board). Generally, the MSP is an embedded subsystem that is not intended to be customer visible. This section includes a general description of its functions.

The MSP is generally under direct command from the SSP. It controls the board level power supplies, reports environmental sensor readings to the SSP, and is responsible for initialization of the node chips and initial program load of the nodes. It also handles critical communications between the nodes and the SSP over the control network, CTLnet.

The MSP boots from software located in Flash. The flash resident code does power on self test (POST) then boots uCLinux via TFTP from the SSP. uCLinux is a derivative of the Linux kernel intended for microcontrollers (see <http://www.uclinux.org/description/>).

Once uCLinux is running, software daemons, `attn` for attention protocol processing and `msspcand` for scan logic and environmentals, are launched. Communications with the SSP are established and the MSP software enters normal operation.

`scboot`, the system utility for booting the nodes, will reboot the MSPs and restart their software as necessary to run the correct versions.

The MSP on each module performs the following functions:

- Controls board level power supplies
- Monitors board level environmental sensors
- Communicates with the various nodes via JTAG ports, attn wires, and reset wires
- Communicates with the SSP via CTLnet in support of console, boot, diagnostics, and operations
- Multiplexes access to multiple nodes from multiple SSP connections (many:many relationship)
- Supports node boot, console, and operations

- Controls the module and cabinet LEDs

**MSP addressing** The MSP on each module has an address of the form:

`sc[x|i|1|a]-mspM`

where

x = SC5832

i = SC1458

1 = SC648, mounted in the right (1) side of the rack

a = SC072

M = the number of the module in the system  
(0 on an SC072, 0 - 3 on an SC648, 0 - 8 on an SC1458,  
0 - 35 on an SC5832)

### **Power and environmental monitoring**

The MSP, on command from SSP software, has the ability to turn power on and off to the nodes. The MSP monitors power supply regulator voltage, current, and temperature, board level temperature sensors, and node temperature sensors, and reports these readings via syslog to the SSP on a regular basis (nominally every 10 minutes). On the SSP, this information is collected in a log file.

In addition, the MSP has hardcoded absolute temperature limits beyond which it will shut down the nodes.

For details on where to find log files, see "Locating Log Files" on page 219.

### **Reset and Boot**

The MSP, through the `msscand` daemon, uses JTAG style scan logic to initialize and boot node chips. The detailed sequencing of these operations is controlled from the SSP using an RPC style protocol over CTL-net. On the SSP, this is initiated using the `scboot` program.

Alternatively, the diagnostics supervisor `dash`, running on the SSP, uses the same mechanisms to load and run diagnostics.

Ultimately, to load a test program or to boot the OS, the SSP and MSP work together to load a sequence of ELF format executable files into the memory of a node and start it running.

### **Console**

Each node has a communications register which can be written and read both by the node, using programmed IO, and by the MSP, using scan chain operations. By convention, this register is used in a multiplexed way to carry console traffic between the node and the SSP. This is informally

called “the attention register.” The MSP collects console traffic from all nodes, and passes it along, over CTLnet, to the console server running on the SSP.

The console servers (`scconserver` and `conserver`) create individual log files for each node, and also permit one to connect a terminal to the “console port” of any node. This path uses CTLnet, and therefore does not rely on the SiCortex interconnect fabric.

The log files and their locations are described in "Locating Log Files" on page 219.

**MSPnet** The attention register is also used to carry IP network frames that are part of the control network, CTLnet. This means that each node has an IP network path to the SSP that does not depend on the fabric.

Each node has a network address for this interface of the form:

`sc[x|i|1|a]-msp $y$ -n $z$`

where  $y$  = the module number and  $z$  = the node number

**gdb debugging** In the event of a kernel crash, or for debugging standalone programs, the MSP implements support for `gdb`, the GNU symbolic debugger. This allows `gdb` remote debugging of a node. Each node processor has a MIPS standard EJTAG interface, which is used for this purpose. The MSP creates telnet listeners that support the `gdb` protocol.

To invoke `gdb`, run `scgdb`, the cross-debugger for the node, and connect to the appropriate node for low level debug access. The `gdb` listeners are on MSP ports 2350-2376, for nodes 0-26. For details on using `gdb`, see:

[www.sourceware.org/gdb/onlinedocs/gdb.html](http://www.sourceware.org/gdb/onlinedocs/gdb.html)

# Chapter 2 System Software Concepts

This chapter explains the concepts you need to understand in order to effectively configure, customize, and boot a SiCortex system.

To configure and boot your system, follow the actual procedures, in order, in Chapter 3 - *Configuring and Booting the System* on page 51.

## Concepts—Installing, Configuring and Booting

### Concept—New System Configuration and Boot

The first time you set up your system:

1. Read the system concepts in this chapter.
2. Execute the steps in:

Chapter 3 - *Configuring and Booting the System* on page 51 **in order**.

These steps are required, to customize your system for your site and to configure the system so that all components boot successfully and interoperate fully and correctly. Earlier steps lay the groundwork for later steps.

### Concept—SSP Ships with Software Pre-installed

The SSP is a Linux server that ships pre-installed with a full, operational software installation.

The software pre-installed on the SSP includes:

- the base SSP operating system
- diagnostics
- management software
- kernel images and file system images for booting the compute nodes and the module service processors (MSPs) that support the nodes

## Concept—Log of Booted SSP Software Version


When you boot the SSP, the version of the software that is booted is logged to this log file, where *yyyymmdd* is the current date:

```
/var/log/messages-yyyymmdd
```

## Concepts—Security on the SiCortex System

### Concept—Using iptables as the SSP Firewall

When you boot the SSP, the access to it is initially wide open. The boot process preconfigures the SSP to NAT/forward everything that comes from the SiCortex system's internal networks, to the site LAN.

-  There is no automatic preconfiguration of any SSP firewall security. By default, no access restrictions are provided.
- The SiCortex system supports `iptables`.
  - However, it does *not* support the use of:
    - Third-party `iptables` configuration tools
    - Custom `iptables` configuration files
  - To configure the SSP firewall to restrict access to your SSP, use `iptables`.
  - A simplified sample script for configuring `iptables` is provided in the section *Step 8—Configure SSP Firewall* on page 65.
  - Ensure you do not block out access to the SSP from the netblock you configured for the interior nodes on the GRUB screen in *2b—Boot the SSP* on page 54.
  - For information on configuring your firewall with `iptables`, see the following link or the `iptables` reference of your choice:
    - [http://gentoo-wiki.com/HOWTO\\_Iptables\\_for\\_newbies](http://gentoo-wiki.com/HOWTO_Iptables_for_newbies)

## Concept—Root Passwords on the SSP and Nodes

Both the SSP and the nodes are Linux systems. Therefore, both have root passwords. You can use the root passwords as you normally would, to control who has access at the root level on each system.

## Concept—SSP Access for SysAdmin Only

It is recommended that only the system administrator have privileges to log onto the SSP on the SC648, SC1458, and SC5832. On these systems, the SSP is designed to be the management appliance only. It does not have the capacity to handle application users logging in, compiling programs, and running jobs.

# Concept—Document Configuration Changes

You'll make a number of changes to your SiCortex system, to configure and customize it for your site, your network, your storage, your applications, and your users.

It's strongly recommended that you keep an ongoing record of all the changes and customizations you make to your system. This record will greatly simplify the task of recreating your configuration when you install a new software version, or experience any unforeseen event that makes it necessary to recreate your site configuration from scratch.

**⚠ In general, the SiCortex system does not keep a record of changes you make to system configuration files, or perform any automatic backup of your customized files.**

The only configuration information that is maintained automatically is the configuration contained in `/etc/sicortex-system.conf`. This file is a symlink to `/var/state/etc/sicortex-system.conf`. The contents of `/var/state` are preserved across installs of new releases (unless you repartition the install disk), and are shared between all installed software releases.

Before you begin the process of configuring and booting your system:

1. Establish a method for documenting changes you make to system files, and for backing up copies of your modified versions of these files.

2. Anytime you modify a configuration file anywhere on the system:


- Record how you modified the file from the generic version.
- Back up a copy of your modified configuration file, in a safe place.

Appendix E—*Customized System Files— Checklist* on page 277 lists the files you are likely to modify as part of configuring, booting, and customizing your system.

## Concepts—What to Back up, and Why


### Concept—Back up Configuration Files You Edit

The software on the SiCortex system includes a number of configuration files in different directories, that you are likely to modify to set up and customize your system. Some of these files are standard Linux files, and others are unique to SiCortex systems.

-  The SiCortex system does not keep a record of changes you make to system configuration files, or perform any automatic backup of your customized files.

You are responsible for tracking the customizations you make to configure the system, so that you can recreate them the next time you install a new software release.

Appendix E—*Customized System Files— Checklist* on page 277 lists the files you are likely to modify as part of configuring, booting, and customizing your system.

-  Use the list of commonly customized files in the appendix to:
- Check off each system file you edit.
  - Back up your version of that file in a location that persists across boots and installations of new software releases.
  - Note where you store a backup copy of your version of the file.

The list of the backup locations of all your configuration changes will be very helpful in preserving your customizations, when you install the next software release.



## Concept—Back up the Node Root File System

The SiCortex system is shipped with a complete set of Linux system software, including two root file systems, one for the SSP and a related one for the nodes, that are designed and engineered to work together and to work on the SiCortex system hardware. The system cannot function without these two root file systems.

Early in the process of configuring and booting the system, before you have customized any of the configuration files or settings, it's recommended that you create a backup copy of the node root file system, as a tarball.

This ensures that if your working copy of the node root file system becomes corrupted for any reason, you can always restore the original node root file system.

## Concepts—IP Addressing on the SiCortex System

### Concept—Names and Addresses

In the SiCortex system, internal messages from the SSP to the nodes, and from one node to another, travel via IP. The SSP and the nodes also communicate with other devices on your LAN and the Internet via IP. Therefore, you need to allocate two sets of addresses for your SiCortex system:

- **Unique IP addresses** must be allocated, either statically or via DHCP, for the following entities: the SSP, the head node, and any configured routing nodes.
- In addition, **names matching these addresses** must be defined in the local DNS servers for the SSP and the head node, and for routing nodes if there is a need for other systems to make inbound connections to them.
- **A netblock in the private range**, that is not used anywhere else on your LAN. The system uses this netblock for IP addressing of messages that are internal to the SiCortex system.

### Concept—Choosing Netblock in GRUB Screen

When the SSP boots, it pauses at a GRUB boot prompt, and presents options similar to those shown below.

```
>> Help for boot options <<
Boot sda5, 3.0.0_rc11-r52 (R1), internal netblock 172.31.x.x/16
Boot sda5, 3.0.0_rc11-r52 (R1), internal netblock 10.137.x.x/16
Boot sda5, 3.0.0_rc11-r52 (R1), internal netblock 192.168.x.x/16
Placeholder for partition 6, netblock 172.31.x.x/16
Placeholder for partition 6, netblock 10.137.x.x/16
Placeholder for partition 6, netblock 192.168.x.x/16
Placeholder for partition 7, netblock 172.31.x.x/16
Placeholder for partition 7, netblock 10.137.x.x/16
Placeholder for partition 7, netblock 192.168.x.x/16
```

The GRUB screen shows:

- Which of the three disk partitions you will be booting from (above, sda5)
- The version of software you are using (above, R1)
- The three internal netblocks from which you can choose. You must select the internal IP netblock *that cannot overlap with anything on your site LAN*, to be used for IP networks that are inside the SiCortex system.

**⚠ To avoid conflicts, you must select an IP address range in the GRUB boot menu that is not already in use on your LAN and is not part of a network already accessible on your LAN.** For example, if your site LAN is 10.A.B.C, you should choose 172.31.x.x.

The internal network block must be a CIDR /16 network (X.Y.0.0/16, roughly equivalent to a class B network), providing the first two IPv4 address octets. This leaves the last two octets available for internal sub-netting.

Once you boot the system with a particular internal IP network block, that block will be the default for successive boots until you change your selection at the GRUB boot menu on a later boot.

## Concept—Time Zones on the System

The SSP, the nodes, and the MSPs each have a time zone setting, which you must set individually. You do this early in the configuration and booting process. Doing so ensures that all messages on the system will have consistent timestamps that reflect the actual time in your location.

Eastern Standard Time (EST and EST5EDT) is the default time zone on the SSP, the nodes, and the MSPs when the system is shipped. If you run on Eastern Standard Time, you do not need to reset your time zones.

The following table shows the main time zones for the continental US:

Abbreviation	Time Zone
EST5EDT	Eastern Standard Time
CST6CDT	Central Standard Time
MST7MDT	Mountain Standard Time
PST8PDT	Pacific Standard Time

For more information on time zones and their corresponding time zone variables, see:

<http://www.cs.berkeley.edu/CT/ag4.0/appendid.htm>

## Concept—SSP Networking


You control and configure networking for the SSP with the file `/etc/conf.d/net`, on the SSP. This is the default network configuration file for the SSP.

After you have booted the SSP and logged in as `root`, the next step is to examine `/etc/conf.d/net`, and make any necessary edits for your local context.

There are two versions of this file in the following locations on the SSP:

<code>/etc/conf.d/net</code>	Default copy of the file that executes when you boot the SSP
<code>/etc/conf.d/net.example</code>	Sample version of the file showing many additional options

**`/etc/conf.d/net`** The SSP boot process automatically reads and processes the `/etc/conf.d/net` file when you boot the SSP. After you boot the SSP for the first time, you need to examine this file to see what commands it executed, and to make any edits that may be necessary to customize it for your site.

 If you are installing a new software release, save your customized `/etc/conf.d/net` file first, to reestablish your site's SSP networking after the installation.

This file resembles a bash script, but is actually a series of parameters that define, for the SSP:

- the interfaces that are available
- the addresses that are to be used
- the location of the NTP server
- the location of the DNS server

### Concept—Choosing DHCP (Default) or Static Addresses

Your existing network setup will dictate whether you configure your SiCortex system to resolve network addresses using DHCP or to use static addressing.

**On SSP** You configure your SSP to use DHCP, or to use static addressing instead, by the edits you make in the file `/etc/conf.d/net`.

**The `/etc/conf.d/net` file as shipped configures your SSP to use DHCP.** The file is reproduced in the procedure, *Step 6—Configure SSP Networking* on page 60.

**On nodes** When you configure nodes as network devices, you also configure them to use DHCP or static addressing by the edits you make in the `si-cortex-system.conf` file. See *Step 9—Configure System and Networking Settings* on page 67.

## Concepts—Customizing and Configuring Your System

### Concept—Configuring System Parameters

You use the `/etc/si-cortex-system.conf` file to configure many system functions.

The `/etc/si-cortex-system.conf` file contains keywords to which you assign values. The utility that boots the nodes, `scboot`, reads these values and seeds them into many areas in the system, to control the following aspects of the system configuration:

- Network configuration for head nodes
- Network configuration for router nodes
- Network configuration for a default router
- Network configuration of optional extra routes


- Nodes/modules to exclude from the boot
- Default node kernel log level
- Default node kernel and node root file system

The filename `/etc/sicortex-system.conf` is a symlink to `/var/state/etc/sicortex-system.conf`. This location is shared between all software releases installed on the system.

You can change the configuration in this file at any time. However, the nodes must be rebooted for the changes to take effect. There are no mandatory settings in the `sicortex-system.conf` file that are required for booting the system, though configuring a head node is required for user access.

### Concept—Use `sicortex-system.conf.example`

The `/etc/sicortex-system.conf.example` file is a sample that documents the current options you can specify in `/etc/sicortex-system.conf`.

 The system ships with a `sicortex-system.conf.example` file. This sample file shows you how to build your own working `sicortex-system.conf` file.

### Concept—Basic Cluster and Boot Parameters

The `Basic cluster and boot parameters` section of the `sicortex-system.conf.example` file is shown below:

```
#-----  
# Basic cluster and boot parameters  
#  
# The way we serve the rootfs to the nodes.  
#  
# Default mode for serving rootfs on Catapult  
# sca.boot.rootfs-mode = nfs  
#  
# Default mode for serving rootfs on SC648  
# scl.boot.rootfs-mode = nfs  
#  
# Default mode for serving rootfs on SC1458  
# sci.boot.rootfs-mode = nfsnbd  
#  
# Default mode for serving rootfs on SC5832  
# scx.boot.rootfs-mode = nfsnbd  
#  
# The node kernel's log level to use while booting  
#  
# scl.boot.log-level = 8  
#  
# Additional arguments to append to the node kernel command line  
#  
# scl.boot.append-kargs = initcall_debug  
# -----
```

This section of the file lets you configure these parameters:

- Method used to serve the root file system to the nodes
- Kernel logging level
- Additional node kernel command line arguments

This section of the file looks similar to the sample above.

## Concepts—Root File System

The SiCortex system has to solve an interesting problem: how to provide the same root file system to all the nodes, without consuming the memory and cycles that would be required to simply copy the complete rootfs to every node.

### Concept—Two Copies of Root File System on SSP

After the SSP boots, it contains two copies of the node root file system. One is the live rootfs file tree. The other is a node rootfs image that is

regenerated by scboot every time the nodes are booted. The node rootfs image is used to serve the node rootfs to the interior nodes.

### Concept—Serving the Node Root File System

Instead of copying the complete root file system to every node, the system sets up a small number of nodes to serve the root file system to the rest of the nodes. This design makes it unnecessary to keep a complete copy of the node rootfs on every node, thereby saving much more memory on each node for compute jobs.

Depending on system size, the SiCortex system uses either NFS or NBD over NFS to serve the node root file system to the nodes.

### Concept—Default Methods for Serving the Root File System

The following table shows the system defaults for serving the root file system to the nodes, in Release 2.1, 2.2, and 3.0:

Table 1. **Method of Serving the Root File System, by Model**

<b>Model</b>	<b>Rootfs Nodes</b>	<b>Release 2.1</b>	<b>Release 2.2</b>	<b>Release 3.0</b>
SC648	m0n6	NBD	NFS	NFS
SC1458	m0n6	NBD	NBD	<b>NFSNBD</b>
SC5832	m0n6 m2n6 m4n6 m6n6	NBD	NBD	<b>NFSNBD</b>

The system configuration sample file `sicortex-system.conf.example` shows the default mode for serving the rootfs to the nodes, by model size:

```

#-----
# Basic cluster and boot parameters
#
# The way we serve the rootfs to the nodes.
#
# Default mode for serving rootfs on Catapult
# sca.boot.rootfs-mode = nfs
#
# Default mode for serving rootfs on SC648
# scl.boot.rootfs-mode = nfs
#
# Default mode for serving rootfs on SC1458
# sci.boot.rootfs-mode = nfsnbd
#
# Default mode for serving rootfs on SC5832
# scx.boot.rootfs-mode = nfsnbd
    
```

Figure 1. `sicortex-system.conf.example` shows rootfs defaults

The `sicortex-system.conf.example` file includes commented lines that show the default mode for serving the rootfs to the nodes on each system model, as shown in Figure 1. The possible values are:

<b>nfs</b>	Specifies NFS to serve the rootfs. Provides a <b>read/write rootfs</b> on the nodes.
<b>nfsnbd</b>	Specifies NBD to serve the rootfs, using NFS protocol for file transfer. Provides a <b>read-only rootfs</b> on the nodes.

**Do Not Override the Default for Serving the Rootfs**

It is strongly recommended that you use the system default method of serving the root file system. The defaults are optimized for the scale of each system model.

NBD is required for larger systems because it can scale to serve all the nodes. NBD’s scaling ability is essential on an SC1458 and an SC5832.

**Concept—Modifying the Root File System**

On systems that use NFS to serve the node root file system, all nodes have read/write access directly to the node root file system on the SSP. On these systems, modifying the root file system is easy.



On systems that use NBD to serve the node root file system, the root file system is read-only, so a special command is required to modify it.

**Use `chrootfs` to edit r/o rootfs**

When using NBD mode you can use the `chrootfs` command on a single node, to modify a copy of the root file system. Afterwards, you must reboot the nodes to propagate the changes to live root file systems of the nodes.

For some changes, such as installing packages using `emerge`, you must use `chrootfs` regardless of whether your system serves the rootfs using NBD or NFS.

For details on using `chrootfs` to modify the root file system, see *Using chrootfs to Edit the Root File System* on page 270.

**NFS Provides Read/Write Rootfs on SC648**

By default the system uses NFS to serve the root file system on SiCortex systems up to the size of the SC648. NFS mode provides read/write access to the live node root file system.

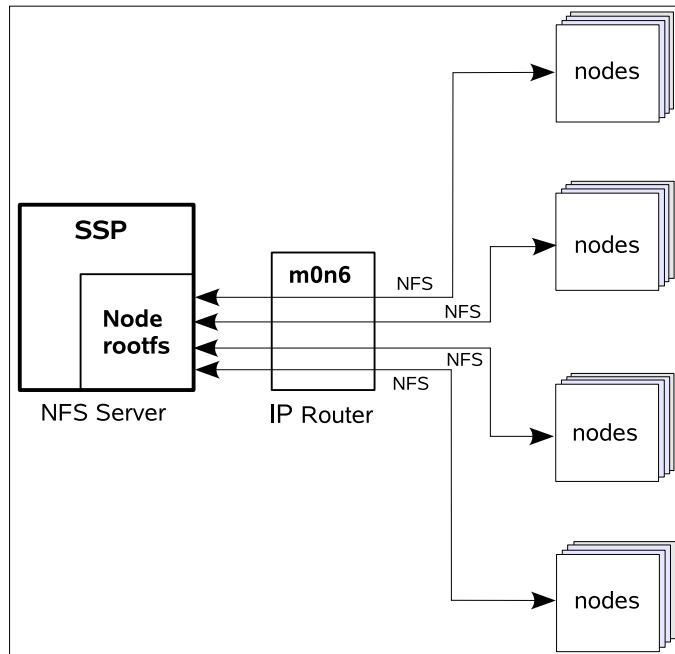


Figure 2. Serving the root file system in NFS mode

Figure 2 shows how the SSP serves the node root file system (*rootfs*) in NFS mode:

- The node rootfs on the SSP is a live file system with a hierarchy.

- The SSP acts as an NFS server, serving the node rootfs to all the nodes on the system.
- The SSP and all nodes have read-write access to the node rootfs on the SSP. The `chrootfs` utility is not needed to modify the node rootfs in this environment.
- One node, by convention `m0n6`, is configured to act as an IP router, routing NFS requests for the node rootfs from the other nodes to the SSP, and routing the replies back.
- The SSP uses NFS protocol to serve the node rootfs to the nodes.
- Every interior node directly and independently mounts the node rootfs that resides on the SSP.
- When any user issues the `scpasswd` command from the SSP to change a user's password, the password change:
  - takes effect in the read/write rootfs on the SSP immediately
  - is visible to the SSP and to all nodes immediately

## NBD Rootfs Is Read-Only on SC1458 and SC5832

The SC1458 and the SC5832 use NBD to serve the root file system to the nodes, as shown in Figure 3.

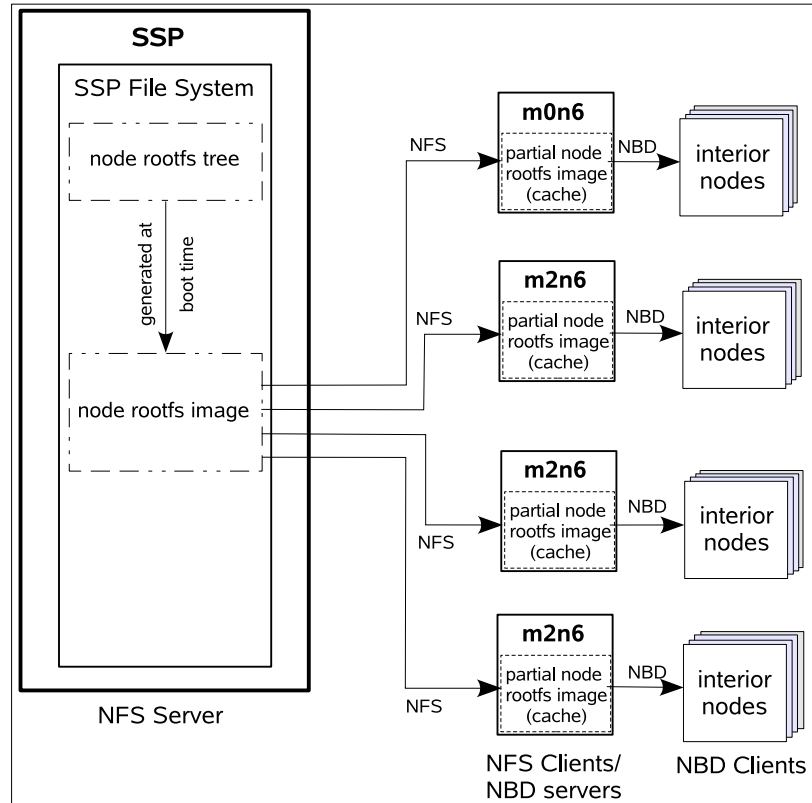


Figure 3. Serving the root file system in NBD mode on an SC5832

- There are two copies of the node roots on the SSP:
  - The live node roots, which is part of the hierarchy of the SSP's root file system. This roots is not accessible from the nodes, except by using chrootfs. See *Using chrootfs to Edit the Root File System* on page 270.
  - An image of the node roots, which `scboot` builds at boot time. The SSP serves this image of the roots to the roots server nodes on demand. Before it boots the nodes, `scboot` rebuilds this image if the live node roots on the SSP has changed.
- The SSP acts as an NFS server, to serve the node roots to the roots server nodes.

- One or four nodes are configured as rootfs servers. The nodes used by default are listed in *Rootfs Server Nodes* on page 38.
- The rootfs server nodes act as NFS clients to the SSP, because the SSP uses NFS to serve the rootfs image to the rootfs server nodes.
- The rootfs server nodes act in turn as NBD servers, to serve the node rootfs image to the interior nodes. The rootfs server nodes maintain a partial, cached, read-only version of the node rootfs image from the SSP. They serve it via NBD protocol to the interior nodes.
- On the SSP, the system administrator has read-write access to the actual node root file system hierarchy.
- The nodes have read-only access to the node rootfs on the SSP.
- The interior nodes never access the rootfs on the SSP, except via `chrootfs`. They have read-only access to the rootfs image on the rootfs server nodes only.
- When any user issues a `passwd` command from the SSP to change a user's password, the password change:
  - `scpasswd` affects only the node root, and is visible only after a reboot.
  - `passwd` on the nodes takes effect immediately if you are using LDAP (no `chrootfs` required).
  - `passwd`, if you are using `passwd` files, requires a reboot of the nodes, if the rootfs is being served with NBD.
- To modify the node rootfs from a node, including `passwd`, `emerge`, etc., you must use the `chrootfs` utility. See *Using chrootfs to Edit the Root File System* on page 270.

## Concepts—Modifying Kernel Behavior

### Concept—Kernel Log Level

**sc1.boot.log-level** You can set the node kernel log level that `scboot` will use while booting the system, according to your preferences:

```
# sc1.boot.log-level = 8
```

The line in the file shows the default log level. To change the log level, uncomment the above line, change “sc1” to the boot partition for your system model, and substitute your chosen log level number.

For more information, see the `syslog` man page.

Also see `/usr/src/linux/Documentation/...`

## Concept—Node Kernel Arguments

**sc1.boot.append-kargs** You can specify additional arguments to append to the node kernel command line.

```
# sc1.boot.append-kargs = initcall_debug
```

The line above shows one argument. To specify additional arguments, uncomment the line, change “sc1” to the boot partition for your system model, and add the arguments to the list, separated by spaces.

# Concepts—Nodes as Network Devices

## Concept—Configuring Nodes for Networking

As part of configuring your SiCortex system, you need to configure some of the nodes as special purpose nodes, with specialized networking roles:

- A *head node* is a node where application users can log in, compile applications, run jobs, and perform other tasks; known on some systems as a login node.

You must configure at least one head node.

- A *router node* is a node that performs routing between the interior nodes on the system and one or more outside networks. Sometimes referred to as a gateway node. (In previous versions of this document, these nodes were referred to as “IO nodes” or “IO gateway nodes.”)

You are not required to configure any router nodes, but you may configure as many as you have the physical and logistical resources to connect, up to the total number of nodes on the system with the ability to support external connections.

- A *storage I/O node* is a node that is used to connect directly to an external storage device such as a disk array.

These special purpose nodes must have external network interfaces, and their network configurations are different from the interior nodes on the system.

Although you could configure a single node with an external network interface as both a head node and a router node, it is not recommended. Each role involves a significant load, so SiCortex recommends that you keep your head node or nodes separate from your router nodes.

**Network-ready nodes**

On each SiCortex system model, only certain nodes have the extra hardware to become networking nodes. Table 2. - Network-Ready Nodes describes these nodes.

Table 2. Network-Ready Nodes

Model	IO-Ready Nodes You May Choose as Head Node, Router Node, or Storage I/O Node
SC648	Node 6 on any module (sc1-m*n6) - has built-in dual gigabit Ethernet interface OR: Node 0, 1, or 3 on any module provided it has a PCIExpress Module installed in the corresponding PCIExpress port.
SC1458	Node 6 on any module (sci-m*n6), or Node 0, 1, or 3 as above.
SC5832	Node 6 on any module (scx-m*n6), or Node 0, 1, or 3 as above.

Table 3. I/O interfaces of PCIExpressModules supported for IP

PCIExpressModule	I/O Interfaces
SysKonnect SK-9122	eth0 and eth1
Myricom Myri-10G	eth0 (only one port)
Broadcom BCM5715	eth0 and eth1


Table 4. PCIe port-to-node connections

PCIe port	Location	Internal connection to node...
1	Top	mxn1

Table 4. PCIe port-to-node connections

PCIe port	Location	Internal connection to node...
2	Middle	m.xn3
3	Bottom	m.xn0

**Eligible nodes** You can configure Node 0, 1, 3, or 6 on any module as a head node or router node, provided you have cabled it appropriately to connect to your site LAN.

 The terms *head node* and *router node* refer only to the Node 0, 1, 3, or 6 nodes you wire and configure to perform IP networking.

### Nodes 0, 1, and 3

Nodes 0, 1, and 3 are connected to PCI Express® ports on the front panel of the module. When connected to your LAN via a PCI Express-Module and cable, Nodes 0, 1, or 3 provide very high-speed, high-bandwidth IO:

PCIExpress Port Number	Port Position on Module Front Panel	Connected to Node Number
PCIExpress 1	Top	sc*-m*n1
PCIExpress 2	Middle	sc*-m*n3
PCIExpress 3	Bottom	sc*-m*n0

### Node 6

Node 6 on each module has an on-board, hard-wired, dual gigabit Ethernet port, which you can wire to your site LAN via the port on the module faceplate as shown in this table:

Ethernet Port on Module Faceplate	Maps to Ethernet Interface	Connected to Node Number
1	eth1 on node 6	sc*-m*n6
2	eth0 on node 6	sc*-m*n6

**Ineligible nodes** Do not use any of the following nodes as a head node or a router node. Do not configure network interfaces in `si:cortex-system.conf` for the following classes of nodes: Storage I/O nodes and FibreChannel nodes.

### Storage I/O and FibreChannel Nodes

Storage I/O and FibreChannel nodes are nodes you have configured to connect directly to disks or disk arrays. They do not perform IP network-

ing or carry network traffic. They only perform storage I/O. *Do not configure network interfaces for your storage I/O or FibreChannel nodes.*

**Rootfs Server Nodes**

Do not choose the node root file system server nodes as the head node or as router nodes. By default, these are:

Model	Default Rootfs Server Node(s)
SC648	m0n6
SC1458	m0n6
SC5832	m0n6, m2n6, m4n6, m6n6

The nodes used to serve the root file system are determined at hardware assembly time, by the modules you connect to MGTnet.

The SC648 / SC1458 *Hardware Installation Guide* specifies that you connect the management net to module 0, which means m0n6 will be used to serve the node root file system.

The SC5832 *Hardware Installation Guide* specifies that you connect MGTnet to modules 0, 2, 4, and 6. This means that m0n6, m2n6, m4n6, and m6n6 will be used to serve the node root file system.

**Cabling networking connections**

For details on how to connect the modules to your network, so you can use any of the eligible nodes above as a head node or router node, see *Cabling the Processor Modules* on page 29 in the SC648 / SC1458 *Hardware Installation Guide*, or on page 28 of the SC5832 *Hardware Installation Guide*.

**Concept—Head Node**

You must configure at least one *head node* (sometimes called a *login node*) on your system.

- Users will log into the head node to run jobs on the system.
- The head node must have external Ethernet interfaces.
- *The head node does not automatically act as a router. Therefore, you must configure the head node to also act as a router node, so that networking will work properly on your system.* The router nodes you select act as the routers for the interior nodes. See 9b—*Configure a Default Router* on page 73.



Establishing a head node requires two edits to `sicortex-system.conf`:

- Edit the `sc1.cluster.head-node` setting to designate your head node.
- **Do** configure the head node as one of your router nodes. This requires that you also create the networking interfaces for the head node (section 9j—*Configure Network Interfaces for Exterior Nodes* on page 75)

**sc1.cluster.head-node** The `sc1.cluster.head-node` keyword configures the head node for the system. This keyword defaults to `sc1-m3n6` if you copy the `sicortex-system.conf.example` file as shipped.

```
# -----
# Configuration for specialized nodes, ie head nodes etc
#
# Which node is the head node, ie has external ethernet interfaces, users are
# expected to log in to it etc. If not set, there is no designated head node.
#
# IMPORTANT! When specifying a head node, you must also configure their
# network interfaces, in the network configuration section, below.

# sc1.cluster.head-node = sc1-m3n6
#
# Which nodes are router nodes. The key name is an historical artifact; this
# list should include only *network* I/O nodes, not storage I/O nodes.
#
# IMPORTANT! When specifying IO node(s), you must also configure
# their network interfaces, in the network configuration section, below.

# sc1.cluster.io-nodes = sc1-m2n1, sc1-m3n1
#-----
```

## Concept—Router Nodes

The latter part of the “Configuration for specialized nodes” section of the `sicortex-system.conf` file lets you select the nodes you will use as router nodes. The router nodes serve as the first-stop routers, providing IP networking services for the interior nodes in the system.

 This section applies to the SC5832, SC1458, and SC648 only.

## Concept—Configuring a Default Router

It’s advisable to configure a default router. Choose an external router that is already connected to your site LAN.

The `<partition>.cluster.default-router` setting configures the selected external router as the default router for your SiCortex system.

- `<external-router>`—Setting it to an external router on your LAN sets up default routes, and causes all traffic to be load-shared among all the router nodes you configure.
- `ssp`—Setting it to `ssp` causes the SSP to be used as your default router. This is not advised on the SC648, SC1458, or SC5832.
- If you fail to specify this setting explicitly, no default routes will be set up.

## Avoiding Routing Problems

To avoid routing problems, observe the instructions for how to configure nodes as network devices above. Also, follow these additional guidelines when configuring your system:

- **Configure a default router.**
  - If you configure the `<partition>.cluster.default-router` setting to `ssp`, the SSP will be used as the default router. This is *not* advised on the SC648, SC1458, or SC5832. WHY: The SSP on the larger SiCortex systems is designed to function as the management appliance only, and will be overwhelmed if used as the default router.
  - To set up default routes for the SiCortex system, you must explicitly configure a default router using the `<partition>.cluster.default-router` setting. This explicit configuration is required, even if you are using DHCP and you have configured your DHCP to provide default routing. **If you fail to explicitly configure a default router, no default routes will be set up on your system.**
- **Configure router nodes.**
  - The router nodes you specify, if any, become the default routers that allow the interior nodes on the SiCortex system to send packets to any location outside the system itself.
  - If there are no router nodes, interior nodes will not be able to route outside the system, except through the SSP.
  - Configuring one or more router nodes, while optional, is very strongly recommended. If you fail to configure any router nodes on your system, you will create other serious networking problems:

- If you have no defined router nodes:
  - Any `scX.cluster.default-router` you designate, other than the SSP, will be invalid.
- **Ensure you configure the network interfaces for:**
  - **The head node** you defined above.
  - **Each of the router nodes** you defined above.
  - **See 9j—*Configure Network Interfaces for Exterior Nodes*** on page 75 below.

## Concept—Configuring Network Interfaces

The `Network configuration for exterior nodes` section of `sicortex-system.conf` specifies which external interface on each configured exterior node (head node, all router nodes) is actually connected to the network.

## Concept—Mapping the Interconnect Fabric

For detailed instructions on connecting the system to networking and storage devices, see the system's *Hardware Installation Guide*.

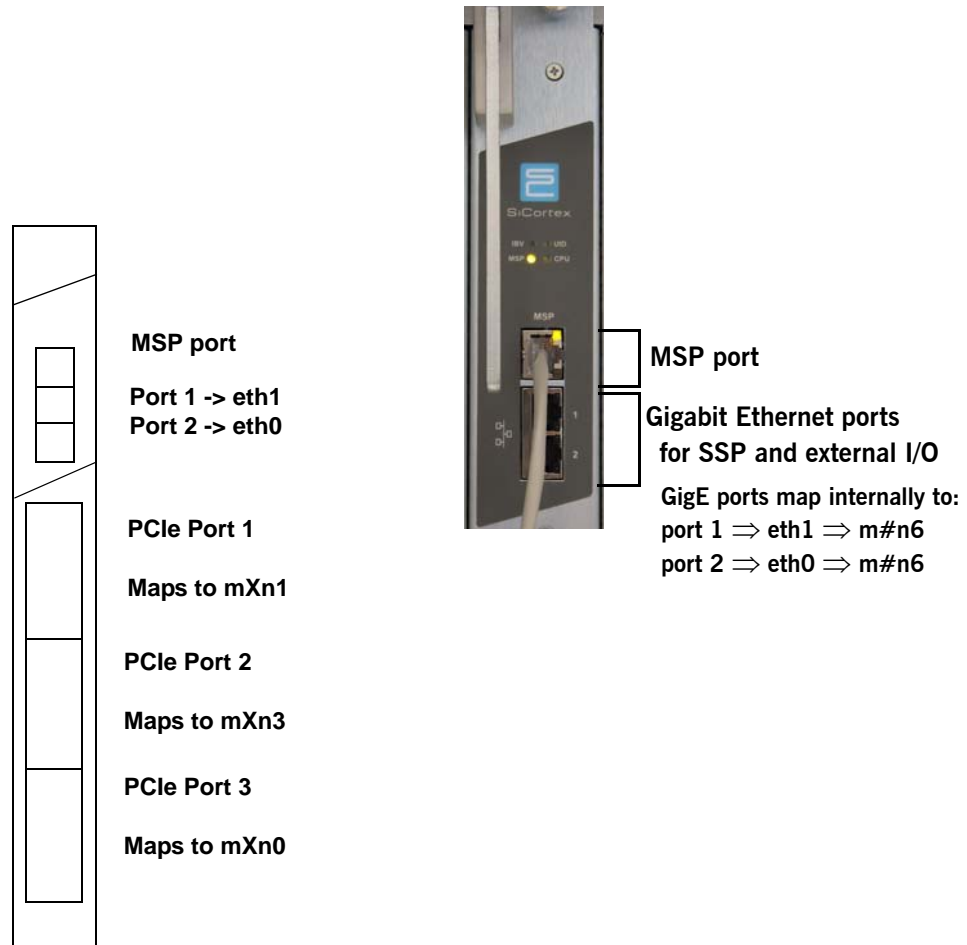


Figure 4. Ports on module faceplate and nodes to which they connect

Figure 4 provides a guide to the ports on the processor module faceplate, and describes the node to which each one connects.

## Concept—Mapping the Interconnect Fabric

The nodes on a SiCortex system communicate with one another across a high-speed fabric interconnect, as described in *The Interconnect Fabric* on page 5.

The default hardware configuration on every model is a fully loaded system, with all slots occupied by processor modules, and all nodes and links active. The nodes on the system are connected via redundant pathways

that form an *interconnect graph* (Kautz or de Bruijn) appropriate to the size of that system model.

When a fully loaded SiCortex system boots with all modules, nodes, and links in service, the boot utility, **scboot**, uses the default interconnect graph to lay out the detailed web of connections across the interconnect fabric, linking all the nodes in the system.

For a variety of reasons, a system may not have the full complement of processor modules, nodes, or the links that connect them. If your system has less than the full complement of processor modules, nodes, or links, **scboot** cannot simply use the default interconnect graph to map the interconnect fabric. In this case, you must notify **scboot** of which modules, nodes, and/or links are absent or not presently usable. This enables **scboot** to build an alternate interconnect graph that can connect all the functioning nodes in the system to one another with sufficient speed and redundancy.

**scboot** will need to build an alternate interconnect graph for your system if any of the following conditions exists:

- Some processor module slots contain placeholder cards.
- One or more processor modules are temporarily out of service.
- One or more nodes are not functioning properly.
- One or more healthy nodes are orphaned, because all of their input or output links go to nodes that are disabled.
- One or more links between nodes are not functioning, because:
  - An individual link is disabled (not in “mission mode”)
  - A healthy link connects two disabled nodes.
  - On a system with placeholder cards, it may be that some links cannot be used due to signal attenuation.

Declaring any of the above conditions triggers **scboot** to map an alternate fabric interconnect graph when booting the nodes, so that all available hardware can communicate and exchange data.

If any of the above conditions exists on your system and you have not declared them to **scboot** in the `sicortex-system.conf` file, **scboot** provides a list of the disabled components, and halts.

### Stranded Nodes

There are numerous combinations of placeholder modules, disabled modules, disable nodes, and disabled links which result in stranded nodes. Because it's hard to notice them, `scboot` automatically detects these cases and alerts you.

Almost always, a node is stranded because either all three input links or all three output links to a given node are either disabled or connect to disabled nodes or modules.

The common cases happen when an entire module is removed from an otherwise complete system, and is not replaced with a placeholder card. On an SC648, there are always three stranded nodes, one on each other board. On an SC5832, there is always one stranded node.

If nodes are stranded, `scboot` alerts you. For example, if you disable module 0 on an SC648, which strands three other nodes, `scboot` displays the following messages:

```
node 50 bad but not explicitly marked bad
node 56 bad but not explicitly marked bad
node 102 bad but not explicitly marked bad
Error, system topology data in /var/state/route_info.sc1 is
inconsistent
```

In the above case, the system administrator should disable those nodes (in `mxn` format) in `/etc/sicortex-system.conf`:

```
sc1.cluster.disabled-nodes = m1n23, m2n2, m3n21
# m1n23 == node 50
# m2n2  == node 56
# m3n21 == node 102
```

*Calculating sequential node number* on page 265 explains how derive the host-name for a node (`scx-mxn` format) starting from the sequential number for that node.

This section explains the concepts behind declaring placeholder cards, and disabled modules, nodes, and links to `scboot`.

## Concept—Declaring Placeholder Cards

If you purchased a SiCortex system that is populated partly with processor modules and partly with placeholder cards, you must declare the placeholder cards in the `sicortex-system.conf` file.

The `Placeholder module configuration` section tells `scboot` to configure the system so that the links in the interconnect graph connect the

active nodes on your processor modules correctly. These links must be mapped differently through placeholder cards.

The sample `sicortex-system.conf.example` file provides, in comments, the default placeholder card declaration for each combination of model and number of placeholder cards. In the file, the following names refer to the system models:

Blizzard	SC5832
Hail	SC1458
Snow	SC648

For information on the valid hardware combinations of processor modules and placeholder cards, see *Placeholder Card Configurations* on page 6.

For detailed instructions on declaring placeholder cards, see *9m—Declare Placeholder Cards* on page 78.

## Concept—Declaring Disabled Nodes, Modules, Links

From time to time, you may have a node, module, or link that has problems, and either malfunctions or does not boot.

Use the “Other cluster-wide configuration” section of the `sicortex-system.conf.example` file to declare to `scboot` any modules, nodes, or links that are disabled. Declaring them as disabled excludes them from the boot process until they can be investigated and repaired. Detailed instructions for how to modify this section of the file appear in *9n—Declare Disabled Nodes, Modules, Links* on page 81.

The “Other cluster-wide configuration” section in the sample file is reproduced below. Check the version on your system for the latest sample file.

```
# -----
# Other cluster-wide configuration

# If some nodes are broken or misbehaving, they can be masked
# out of the system configuration by adding them to this list.
#
# Disabled nodes, modules and links
# sc1.cluster.disabled-nodes = m0n21, m2n23, m3n2
# sc1.cluster.disabled-modules = 1,2,3
# sc1.cluster.disabled-links = m1n20-rx0, m3n12-tx2
# -----
```

The section of `sicortex-system.conf.example` shown above is provided so that you can declare your disabled nodes, modules, and links to `scboot`. However, this is not the file that directly controls which nodes, modules, and links `scboot` excludes from the boot.

**route\_info file** Instead, the values that `scboot` reads for the “disabled” settings govern its input to the `/var/state/route_info.<partition>` file. This is the file that controls how `scboot` constructs the interconnect graph for the system at boot time. However, to preserve system integrity, you should not edit the `route_info` file directly.

When `scboot` finds disabled nodes or modules listed in the `/var/state/route_info.<partition>` file, it excludes them from the system interconnect graph. Excluding the disabled nodes allows `scboot` to configure the remaining good nodes in a re-calculated, complete interconnect graph, so that when the system boots, all booted nodes are fully connected to one another.

**Regenerated on every boot** Each time it’s called, `scboot` automatically regenerates the `/var/state/route_info.<partition>` file, based on the latest set of commands in `sicortex-system.conf`. Then `scboot` uses the `route_info` file to determine which components to exclude from the boot. If there are *no* disabled commands in `sicortex-system.conf`, then `scboot` generates a `route_info` file that includes *all* nodes, modules, and links on the system in the boot.

For instructions on declaring disabled nodes, modules, and links, see *9n—Declare Disabled Nodes, Modules, Links* on page 81.



## Debugging Disabled Nodes and Links

The flow chart in Figure 5 on page 48 shows the process for determining which nodes and Links are down, and need to be disabled in `sicortex-system.conf`.

### Notes on the Figure

Where Figure 5 says “Reboot the node,” it is referring to rebooting a single node on the system while the other nodes are still running. See *Rebooting a Single Node* on page 93.

Two `mfd.log` messages are cited in the flow chart. Their meanings are:

```
fabmon -f1[r|x] mission mode N changed from 1 to 0
```

This message means that the receive [r] or transmit [t] link N (ie. rx0, rx1, rx2, tx0, tx1, tx2) for the node has lost mission mode, which means that the link is no longer operational.

```
[r|t]x link # failed <message> (other end is <node> rx #
```

This message means that receive [rx] or transmit [tx] link # (0, 1, or 2) for the node has failed, with details in <message>, and the other end of this link is [r|t]x # (0, 1, or 2).

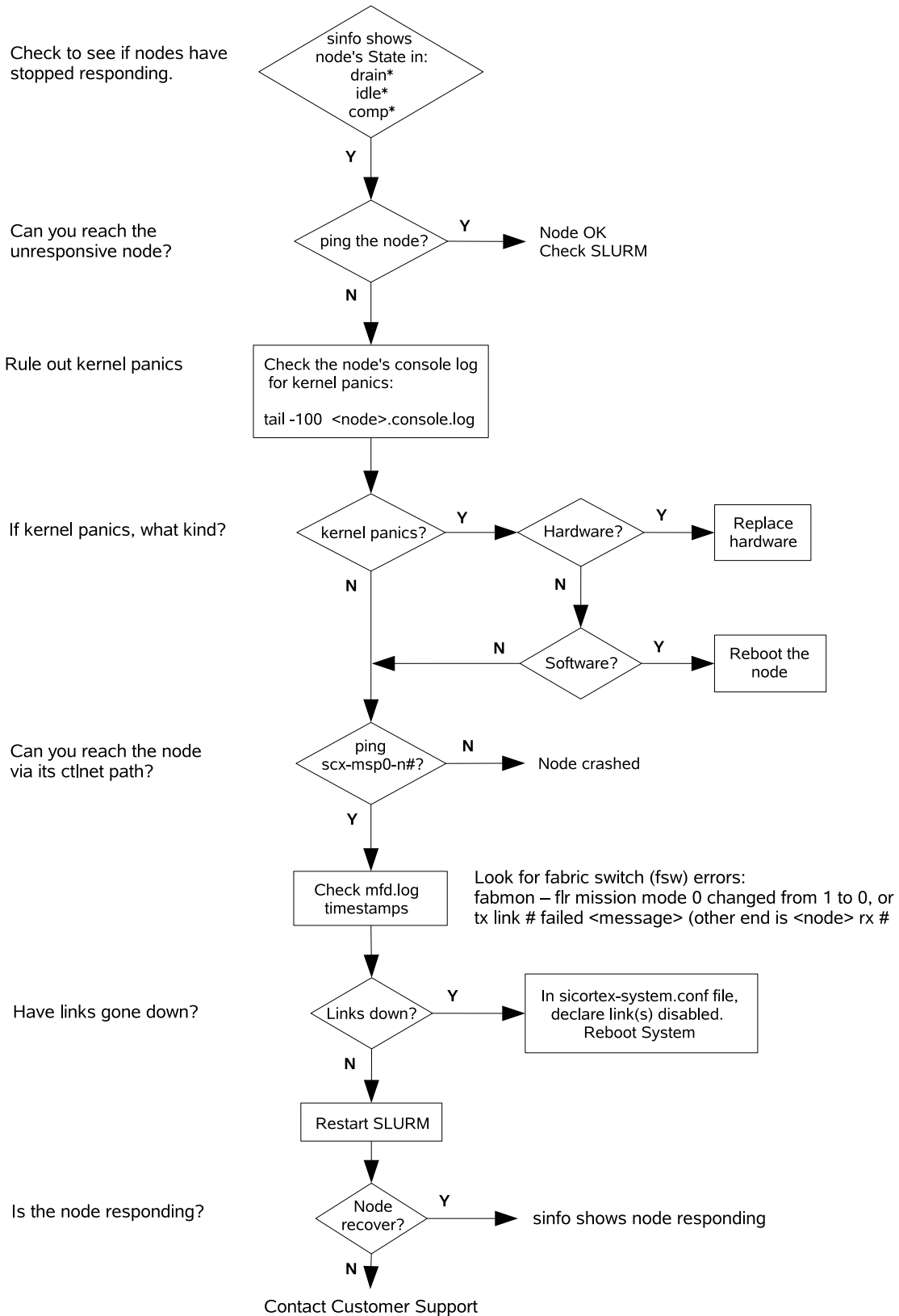


Figure 5. Flow chart for diagnosing disabled nodes and links

## Concept—Overview of Rebooting the System

To reboot the entire system:

1. Reboot the SSP.
2. Boot the nodes using the `scboot` utility.

To reboot just the nodes:

1. Execute the `scboot` command.



# Chapter 3    **Configuring and Booting the System**

This chapter describes the steps to configure, customize, and boot the system for the first time. Each step includes one or more procedures.

*The first time you set up your system, you must execute the steps in this chapter in the order shown.* You perform many of these steps only when you set up the system for the first time, or after you install a new release. Certain procedures, such as booting the nodes, you will probably perform again later.

For the concepts and background behind these procedures, see Chapter 2 - *System Software Concepts* on page 19.

## **Steps for Configuring and Booting the System**

- Step 1—Prepare to Boot the System
- Step 2—Boot the SSP
- Step 3—Change Root Password on SSP and Nodes
- Step 4—Back up the Node Root File System
- Step 5—Set Your Time Zone
- Step 6—Configure SSP Networking
- Step 7—Verify That Basic SSP Networking Works
- Step 8—Configure SSP Firewall
- Step 9—Configure System and Networking Settings
- Step 10—Edit Software Installation Settings
- Step 11—Boot the Nodes
- Step 12—Test the Head Node
- Step 13—Verify Networking Works on the Nodes
- Step 14—Add User Accounts
- Step 15—Reboot the SSP and Nodes and Test All Changes

### **Additional Boot Options**

- Variations on Booting the Nodes

## **Installing a New Release on an Existing System**

If you are installing a new SiCortex software release on an existing SiCortex system, see:

Chapter 4 - *Installing Software Releases* on page 95

## **Configuring and Booting a New System**

If you are installing, configuring, and booting your new SiCortex system for the first time, proceed with the steps in this chapter.

This chapter assumes you have completed all the steps in the *Hardware Installation Guide*.

### **Where to Find the Background Concepts**

This chapter explains the steps and procedures for configuring, customizing, and booting your SiCortex system.

For the concepts behind each step, see Chapter 2 - *System Software Concepts* on page 19.

## **Step 1—Prepare to Boot the System**

Complete these steps before you begin booting the system.

### **1a—Allocate Names and Addresses**

See *Concepts—IP Addressing on the SiCortex System* on page 23.

*This step can happen before the system is delivered.* Allocate unique names and IP addresses in your DNS and DHCP servers for the following:

- the SSP
- the head node
- all nodes to be used as router nodes
- all nodes to be used as storage I/O nodes, for example, nodes to be used as gateways to external NFS or Lustre servers

## 1b—Verify System Has Power

If a separate team has assembled the system, then before you start the sequence of booting the system for the first time:

1. Ensure that the system's main power cords are connected to power.

For more information, see the *Hardware Installation Guide*.

## 1c—Document Your Configuration Changes

It's strongly recommended to record all customizations you make to system files when configuring and booting your system. For the reasons, see *Concept—Document Configuration Changes* on page 21.

It's also important to back up your customized system files. For the reasons, see *Concepts—What to Back up, and Why* on page 22.

Appendix E—*Customized System Files— Checklist* on page 277 lists the files you are likely to modify as part of configuring, booting, and customizing your system.

# Step 2—Boot the SSP

## 2a—Connect a Keyboard and Monitor to the SSP on a New System

The first time you boot the system, it's important to connect a monitor and keyboard directly to the SSP. This allows you to log into the SSP's console and see what's going on, before the SSP is connected to your network.

1. Plug a 640 x 480 analog VGA monitor into the video jack on the back of the SSP.
2. Plug a PS/2 keyboard into the round PS/2 keyboard jack on the back of the SSP. (You can also plug a USB keyboard into the USB port on the front of the SSP.)

## 2b—Boot the SSP

See *Concept—Choosing Netblock in GRUB Screen* on page 23.

1. To boot the SSP, press the ON switch at the upper right of the SSP front panel.



The SSP ON switch lights up green as it powers up. The SSP boots itself automatically on power-up.

2. When the SSP boots, it pauses at a GRUB boot prompt, and presents a set of options similar to those shown, including three possible netblocks:

```
>> Help for boot options <<
Boot sda5, 3.0.0_rc11-r52 (R1), internal netblock 172.31.x.x/16
Boot sda5, 3.0.0_rc11-r52 (R1), internal netblock 10.137.x.x/16
Boot sda5, 3.0.0_rc11-r52 (R1), internal netblock 192.168.x.x/16
Placeholder for partition 6, netblock 172.31.x.x/16
Placeholder for partition 6, netblock 10.137.x.x/16
Placeholder for partition 6, netblock 192.168.x.x/16
Placeholder for partition 7, netblock 172.31.x.x/16
Placeholder for partition 7, netblock 10.137.x.x/16
Placeholder for partition 7, netblock 192.168.x.x/16
```

3. Use the cursor or arrow keys to move the highlight to choose an internal IP netblock *that cannot overlap with anything on your site LAN*, to be used for IP networks that are inside the SiCortex system.

**⚠ To avoid conflicts, select an IP address range in the GRUB boot menu that is not already in use on your LAN and is not part of a network already accessible on your LAN.**

For example, if your site LAN is 10.A.B.C, choose 172.31.x.x.

4. Press the Return key to select the highlighted netblock.



When you press Return, the SSP boots. Wait for the boot to finish. A number of boot status messages print to the console—the monitor you plugged into the SSP—during the process.

## Step 3—Change Root Password on SSP and Nodes

For security reasons, it is strongly recommended that you change the root password for the SSP and for the nodes immediately, from the published default to a secure password.

See *Concept—Root Passwords on the SSP and Nodes* on page 21.

### 3a—Change the SSP Root Password

To change the root password for the SSP itself, you must edit the root file system **for** the SSP, **on** the SSP. Use the regular `passwd` command:

1. Log in to the SSP as `root`:
  - Press RETURN on the keyboard until you get a LOGIN prompt on the monitor.
  - At the prompt, type `root` and the default root password:

```
LOGIN: root
PASSWORD: sicortex
```
2. Type `passwd` at the prompt.

The `passwd` command edits the SSP's root password, which resides in the copy of the `passwd` file in the SSP's root file system.

3. Follow the prompts to change the SSP's root password.

### 3b—Change the Node Root Password

To reset the root password for the nodes, use the `scpasswd` command:

1. Since you just reset the SSP root password, you should already be logged into the SSP as `root`.
2. Type `scpasswd` at the prompt.

The `scpasswd` command is SiCortex-specific. It has the same syntax as the `passwd` command, but `scpasswd` edits the copy of the

## Step 4—Back up the Node Root File System

`passwd` file in the *node* root file system on the SSP. This is an image of the root file system that will be served to the nodes. For more information, see *Concepts—Root File System* on page 28.

3. Follow the directions to change the nodes' root password.

The new node root password will be propagated to the nodes later, when you issue the `scboot` command in *Step 11—Boot the Nodes* on page 85.

**scboot command** The command to boot the nodes is `scboot`. This command boots the kernel on all nodes in the system. Many of the configuration steps in this chapter become effective the next time you issue a `scboot` command to boot the nodes.

## Step 4—Back up the Node Root File System

### Back up the Node Rootfs

See *Concept—Back up the Node Root File System* on page 23.

At this point in the process, it is advised that you back up the root file system, using a backup method of your choosing.

One way to do this is to use `tar(1)` as follows:

1. As `root` on the SSP, change to the directory that contains the root file system:

```
cd /opt/sicortex/rootfs
```

2. Enter this command to create a tarball of the root file system in the same directory as above:

```
tar czf build.<date>.tgz build/
```

The above commands create a tarball of the node root file system that can later be unpacked to restore the root file system.

## To Unpack the Backup Rootfs Later

If you need to unpack the tarball at a later date to restore your root file system, you would use the following commands. **Do not execute these commands now:**

```
cd /opt/sicortex/rootfs
mv build build.corrupt
tar xzf build.<date>.tgz
```

## Step 5—Set Your Time Zone

Before you continue, you need to set the time zone on the SSP and the nodes, and also on the MSPs. Doing so at this point ensures that all messages in log files will have accurate timestamps from this point on.

See *Concept—Time Zones on the System* on page 24.

### EST Is the Default Time Zone

Eastern Standard Time (EST and EST5EDT) is the default time zone on the SSP, the nodes, and the MSPs. You do not need to set your time zone if you run on Eastern Standard Time.

### 5a—Checking Time Zone Files

1. SSH as root to the SSP, or be on the SSP's console.
2. To show the time zone on the SSP, type these commands:

```
readlink -f /etc/localtime
```

This command shows the correct time zone file, in `/usr/share/zoneinfo....`

3. To show the time zone on the nodes, from the SSP, type these commands:

```
readlink -f /opt/sicortex/rootfs/default/etc/localtime
```

This value will also be `/usr/share/zoneinfo`, not `/opt/sicortex/rootfs/default/usr/share/zoneinfo`.

4. You cannot actually check the MSP time zone until after the first `scboot`.

## 5b—Change the SSP Time Zone

On the SSP, the instructions to set the time zone are the same as in the Gentoo Linux handbook (see the link below).

[http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=1&chap=7#doc\\_chap1](http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=1&chap=7#doc_chap1)

1. You first need to select your time zone so that your system knows where it is located. Give the following command to list the time zones in `/usr/share/zoneinfo`.

```
# ls /usr/share/zoneinfo
```

2. Then make `/etc/localtime` into a link to your time zone file:

```
ln -sf /usr/share/zoneinfo/<timezone> /etc/localtime
```

For example, to use Pacific Standard Time:

```
# ln -sf /usr/share/zoneinfo/PST8PDT /etc/localtime
```



Avoid using the `/usr/share/zoneinfo/Etc/GMT*` timezones as their names do not indicate the expected zones. For instance, GMT-8 is in fact GMT+8.

## 5c—Prevent Clock Inconsistency on the SSP

The file `/etc/conf.d/clock`, as shipped with your SiCortex system, saves the time to the hardware clock in terms of UTC.

If you modify the file so that your hardware clock is not using UTC, then you need to add `CLOCK="local"` to the `/etc/conf.d/clock` file. Otherwise you will notice clock inconsistency.

1. The following URL displays Chapter 8 in the Gentoo Linux handbook. Section 3.4.

<http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=1&chap=8>

2. Redefine the same time zone that you previously linked to `/etc/localtime`, so that further upgrades of the `sys-libs/time-zone-data` package can update `/etc/localtime` automatically.

3. Edit `/etc/conf.d/clock`, using this command:

```
# nano -w /etc/conf.d/clock
```

For instance, if you used the GMT timezone, you would add

```
TIMEZONE="GMT"
to /etc/conf.d/clock
```

When you're finished configuring `/etc/conf.d/clock`, save the file and exit.

## 5d—Change the Nodes Time Zone

Setting the time zone on the nodes is basically the same process as setting the SSP time zone.

In this example, the time zone on the nodes is set to EST5EDT:

```
# ln -sf /usr/share/zoneinfo/EST5EDT
/opt/sicortex/rootfs/default/etc/localtime
```

## 5E—Change the MSPs Time Zone

You also need to explicitly set the time zone for the MSPs, if your time zone is anything other than the default of Eastern Standard Time.

**⚠ NOTE: The MSPs' TZ file format differs from other time zone files.**

If the file `/var/state/msp/etc/TZ` exists on the SSP, it overrides the `/etc/TZ` file on the MSP.

1. Type the following command to create the `/msp/etc/` subdirectory:

```
mkdir -p /var/state/msp/etc/
```

2. Using a text editor, create a TZ file in the `/var/state/msp/etc/` directory with contents for your time zone, similar to the following sample TZ files.

For example, to change the time zone on the MSPs to Central Standard Time with Daylight Savings, you would use:

```
CST6CDT,M3.2.0/02:00:00,M11.1.0/02:00:00
```

instead of the default:

```
EST5EDT,M3.2.0/02:00:00,M11.1.0/02:00:00
```

For details on the contents of the time zone file, see:

[http://www.opengroup.org/onlinepubs/009695399/basedefs/xbd\\_chap08.html](http://www.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap08.html)

## Guidelines—Editing Configuration Files

Many of the rest of the steps below include instructions for editing system configuration files. In many cases, these files include commented lines that are provided as samples you may edit. The following instructions apply to editing any system configuraton file:

1. Remove the # character to uncomment the line.
2. Remove any leading spaces, so the first character is in column 1.
3. If the line includes any reference to the system boot partition, change it to the boot partition appropriate to your model, where boot partition is one of the following:

`scx` for an SC5832

`sci` for an SC1458

`sc1` for an SC648

For example, to edit this line for an SC5832...

```
# sc1.cluster.io-nodes = sc1-m2n1, sc1-m3n1
```

...you would change all instances of `sc1` to `scx`, and remove any leading spaces, so the first character is in column 1, like this:

```
scx.cluster.io-nodes = scx-m2n1, scx-m3n1
```

You would also edit the list of nodes to show the nodes you are actually configuring as router nodes on your system.

## Step 6—Configure SSP Networking

You need to edit the file `/etc/conf.d/net` to control whether the SSP will use dynamic addresses and DHCP (the default) or static addresses for its networking. See *Concept—SSP Networking* on page 25.

The `/etc/conf.d/net` file, as shipped, is shown below:

**⚠ These procedures apply only to the SC5832, SC1458, and SC648. They do *not* apply to the SC072.**

```
# /etc/conf.d/net:
# $Id: net 56234 2008-06-20 17:45:27Z brooks $

# Pull in system configuration and network profile
source /opt/sicortex/config/cthlib \
    SCv_ssp_site_hostname \
    SCv_system_profile \
    SCv_internal_domain \
    || exit 1

# The "site" interface uses DHCP by default
config_site=("dhcp"); dhcpcd_site="-H -N -h $SCv_ssp_site_hostname"

# BEGIN static site configuration
#
# Get these names and numbers from your system administrator:
#
# IP address:          ip1.ip2.ip3.ip4
# Broadcast address:  bc1.bc2.bc3.bc4
# Netmask:             nm1.nm2.nm3.nm4
# Gateway address:    gw1.gw2.gw3.gw4
# DNS server(s):      DNSa1.DNSa2.DNSa3.DNSa4 DNSb1.DNSb2.DNSb3.DNSb4
# Domain search path: mydomain.name other.mydomain.name another.my.domain.name
# NTP server(s):      some.ntp.server another.ntp.server
# Domain name         ssp.domain.name
#
# Uncomment these lines and fill in the names and numbers above.
#
# config_site="ip1.ip2.ip3.ip4 broadcast bc1.bc2.bc3.bc4 netmask nm1.nm2.nm3.nm4"
# routes_site=("default via gw1.gw2.gw3.gw4")
# dns_servers_site="DNSa1.DNSa2.DNSa3.DNSa4 DNSb1.DNSb2.DNSb3.DNSb4"
# dns_search_site="mydomain.name other.mydomain.name another.my.domain.name"
# dns_domain_site="my.domain.name"
#
# Do NOT configure ntp_servers_site in this file.
# The resulting /etc/ntp.conf file won't serve the nodes.
# Edit /etc/ntp.conf and add your NTP server names there
#
# ... see /etc/conf.d/net.example for further options
#
# END static site configuration

# Source configuration for ctl and mgt* interfaces
source /etc/conf.d/net_${SCv_system_profile}

dns_search_lo=( "$SCv_internal_domain" )
dns_servers_lo=( "127.0.0.1" )
```

## 6a—To Use DHCP

See *Concept—Choosing DHCP (Default) or Static Addresses* on page 26.

1. Ensure you are logged into the SSP as `root`.
2. Examine the `/etc/conf.d/net` file to see if you need to edit it.
3. Edit the `/etc/conf.d/net` file to tailor it to your environment.

You probably won't need to change the file much. It is set up to use DHCP by default.

4. **If you edited the `/etc/conf.d/net` configuration file, reboot the SSP** to make your configuration changes effective, before continuing to the next step.

Note: If you edited only the site network settings, you don't need to reboot the SSP to implement your setting changes. It's sufficient to enter the following command at the physical console (the keyboard and monitor you plugged directly into the SSP) before continuing to the next step:

```
/etc/init.d/net.site restart
```

## 6b—To Use Static Addresses


To use static addresses, edit the `/etc/conf.d/net` file as follows:

1. Ensure you are logged into the SSP as `root`.
2. Comment out the following “dhcp” line in the file, to deactivate it, as shown below:

```
# The "site" interface uses DHCP by default  
# config_site=("dhcp"); dhcpcd_site="-H -N -h $SCv_ssp_site_hostname"
```

3. Un-comment and edit the Static Site Configuration section of the file. Specify the correct static addresses for your system.
4. If there are no valid DNS servers available on your local network, or if the SiCortex system will be set up standalone (not connected to a network), then ensure that the DNS lines in `/etc/conf.d/net` are commented out.



 The `/etc/conf.d/net` file sources the `cthl1b` file to pull in system information. The `cthl1b` library aggregates information about many parts of the system in one place. Many scripts reference this file for specific pieces of system information at run time. **Do not modify the `cthl1b` file. Changes to this file are likely to have wide-ranging and unpredictable consequences.**

5. If you edited the `/etc/conf.d/net` configuration file, reboot the SSP to make your configuration changes effective, before continuing to the next step.

Note: If you edited only the site network settings, you don't need to reboot the SSP to implement your setting changes. It's sufficient to enter the following command at the physical console (the keyboard and monitor you plugged directly into the SSP) before continuing to the next step:

```
/etc/init.d/net.site restart
```

If you changed any DNS/domain information, you must also restart `dnsmasq`:

```
/etc/init.d/dnsmasq restart
```

## Step 7—Verify That Basic SSP Networking Works

1. Verify that IP addresses are working:
  - From the SSP, `ping` the main router on your network.
  - From the SSP, `ping` another machine on your network.
2. Verify that DNS service is working:
  - From the SSP, `ping` one or more entities by name (`www.google.com`, for example, or an NFS server, if your network has no Internet access).
3. Check that NTP is working, using the `ntpq` command, which displays status for NTP:

```
ntpq -c peers
```

## Step 7—Verify That Basic SSP Networking Works

It will take a few minutes for the SSP to synchronize to its NTP server. The response should indicate that the system is syncing to the same NTP server that you specified when you edited the `/etc/conf.d/net` file, above.

4. To test the SSP's connectivity, use the `tracpath` command:

```
tracpath <address-of-external-entity-on-same-LAN>  
tracpath <name-of-external-entity-on-same-LAN>
```

The responses should indicate that the head node connects directly to and returns directly from each of these entities with no intervening connections. This indicates the head node is visible on your network and has two-way connectivity.

At this point, the SSP is up and you have verified that basic networking works on the SSP.

## Step 8—Configure SSP Firewall

When you boot the SSP, the access to it is initially wide open, with no system security. The boot process preconfigures the SSP to NAT/forward everything that comes from the SiCortex system's internal networks, to the site LAN. See *Concepts—Security on the SiCortex System* on page 20.

**⚠ By default, no SSP firewall security is configured, and no access restrictions are provided.**

- The SiCortex system supports `iptables`. To configure the SSP firewall to restrict access to your SSP, use `iptables`.
- Do not block out access to the SSP from the netblock you configured for the interior nodes on the GRUB screen in *Step 2—Boot the SSP* on page 53.
- For information on configuring your firewall with `iptables`, see the following link or the `iptables` reference of your choice:
  - [http://gentoo-wiki.com/HOWTO\\_Iptables\\_for\\_newbies](http://gentoo-wiki.com/HOWTO_Iptables_for_newbies)

**Example iptables script** The following script is provided as an example only. Your actual `iptables` configuration will be different and more complex.

```
#!/bin/bash

# Simplified example of how to use iptables. This is meant only to serve as
# an example. Do not use this script for any real-world security purpose.
# Once you have a set of rules you like for your system, you can save them
# with "/etc/init.d/iptables save"

# Set up some basic parameters
table=INPUT
# DBG=-d
ANYWHERE="0.0.0.0/0"

# Allow all traffic from the nodes. This netblock is the same one specified
# in grub at boot time.
iptables $DBG -A $table -s 172.31.0.0/16 -j ACCEPT

# Allow incoming mail, www, dns, ssh, dns connections. NB that this is just an example.
# The services you allow on your system should be determined by your security needs.
# You can add more or fewer service descriptions to this list.
iptables $DBG -A $table -p tcp -s $ANYWHERE --dport ssh --syn -j ACCEPT
iptables $DBG -A $table -p tcp -s $ANYWHERE --dport smtp --syn -j ACCEPT
iptables $DBG -A $table -p tcp -s $ANYWHERE --dport www --syn -j ACCEPT
iptables $DBG -A $table -p tcp -s $ANYWHERE --dport https --syn -j ACCEPT
iptables $DBG -A $table -p tcp -s $ANYWHERE --dport http --syn -j ACCEPT
iptables $DBG -A $table -p tcp -s $ANYWHERE --dport domain --syn -j ACCEPT
iptables $DBG -A $table -p udp -s $ANYWHERE --dport domain --syn -j ACCEPT

# All other traffic is denied and logged
iptables $DBG -A $table -j LDROP
```

## Step 8—Configure SSP Firewall

```
#!/bin/bash

# Simplified example of how to use iptables. This is meant only to serve as
# an example. Do not use this script for any real-world security purpose.
# Once you have a set of rules you like for your system, you can save them
# with "/etc/init.d/iptables save"

# Set up some basic parameters
table=INPUT
# DBG=-d
ANYWHERE="0.0.0.0/0"

# Allow all traffic from the nodes. This netblock is the same one specified
# in grub at boot time.
iptables $DBG -A $table -s 172.31.0.0/16 -j ACCEPT

# Allow incoming mail, www, dns, ssh, dns connections. NB that this is just an example.
# The services you allow on your system should be determined by your security needs.
# You can add more or fewer service descriptions to this list.
iptables $DBG -A $table -p tcp -s $ANYWHERE --dport ssh --syn -j ACCEPT
iptables $DBG -A $table -p tcp -s $ANYWHERE --dport smtp --syn -j ACCEPT
iptables $DBG -A $table -p tcp -s $ANYWHERE --dport www --syn -j ACCEPT
iptables $DBG -A $table -p tcp -s $ANYWHERE --dport https --syn -j ACCEPT
iptables $DBG -A $table -p tcp -s $ANYWHERE --dport http --syn -j ACCEPT
iptables $DBG -A $table -p tcp -s $ANYWHERE --dport domain --syn -j ACCEPT
iptables $DBG -A $table -p udp -s $ANYWHERE --dport domain --syn -j ACCEPT

# All other traffic is denied and logged
iptables $DBG -A $table -j LDROP
```

## Step 9—Configure System and Networking Settings

In the next series of procedures, you configure the main settings that control the hardware configuration of your system.

On a running system, these settings are in the file:

```
/etc/sicortex-system.conf
```

The boot program, `scboot`, reads this file and uses the values it specifies to configure many parts of the system.

This section explains how to modify the settings in this file to configure various parts of your system. For the concepts and background information, see:

*Concepts—Customizing and Configuring Your System* on page 26

*Concepts—Nodes as Network Devices* on page 35

### 9a—Create Your Own `sicortex-system.conf`

The sample file, `/etc/sicortex-system.conf.example`, documents the current options you can specify in `/etc/sicortex-system.conf`.

The system ships with a `sicortex-system.conf.example` file. This sample file shows you how to build your own working `sicortex-system.conf` file.

For the complete text of `sicortex-system.conf.example`, see Appendix F - *sicortex-system.conf.example* on page 281.

1. Type the following command to ensure the target file exists:

```
touch /etc/sicortex-system.conf
```

2. Make a copy of (don't just rename) the sample file from:

```
/etc/sicortex-system.conf.example
```

to

```
/etc/sicortex-system.conf
```

3. Following the directions in the rest of Step 9—Configure System and Networking Settings, edit your new version of `/etc/sicortex-system.conf` to tailor the configuration for your site.

## 9b—Edit the File Header

The beginning of the `sicortex-system.conf.example` file is shown below:

```
# This is the main configuration file for the SiCortex system.
#   - Contains options specific to the SiCortex software release
#   - For options that begin with a partition, you must scoot the
#     partition for changes to take affect.
#   - /etc/sicortex-system.conf should be a symlink to
#     /var/state/etc/sicortex-system.conf. This allows the system
#     configuration to be persistent across different software release
#     installs on the same SSP.

# If you uncomment any of the example lines in this file,
# check the system prefix and change it to the correct one for your system,
# as necessary:
#
# sca = SC072   (Catapult)
# scl = SC648   (snow)
# sci = SC1458  (hail)
# scx = SC5832  (blizzard)

# This header is necessary for the config parser module. Do not alter.
[DEFAULT]

#
# Warning! When editing this file, use comments only at the beginning
# of a line! Trying to do something like
#
# keyword = value    # comment
#
# will not do what you want!
#
#-----
```

1. Add any comments you wish to your customized version of the `sicortex-system.conf` file header, to document your changes.
2. Your customized version *must* include the `[DEFAULT]` line above.
3. The rest of the comments shown are provided as guides. You may keep the rest of the comments, or omit them.


## 9c—Serving the Root File System to the Nodes

The `Basic cluster and boot parameters` section of `sicortex-system.conf.example` starts with comments that show the default method for serving the root file system to the nodes, for each system model. Smaller systems use NFS, while larger systems use NFSNBD.

For the concepts on how the system software serves the root file system to the nodes using NFS, versus NFSNBD, see:

*Concepts—Root File System* on page 28

```
#-----
# Basic cluster and boot parameters
#
# The way we serve the rootfs to the nodes.
#
# Default mode for serving rootfs on Catapult
# sca.boot.rootfs-mode = nfs
#
# Default mode for serving rootfs on SC648
# sc1.boot.rootfs-mode = nfs
#
# Default mode for serving rootfs on SC1458
# sci.boot.rootfs-mode = nfsnbd
#
# Default mode for serving rootfs on SC5832
# scx.boot.rootfs-mode = nfsnbd
```

 The default method of serving the root file system provides the best combination of efficiency, speed, ease, and scalability for each model size. **It is strongly recommended that you accept the default method of serving the root file system for your model.**

## 9d—Set the Kernel Log Level for Booting

OPTIONAL—You can set the node kernel log level that `scboot` will use while booting the system, according to your preferences. This line shows the default level:

1. To change the node kernel log level, uncomment and edit this line:

```
sc1.boot.log-level = 8
```

For more information, see the `syslog` man page.

Also see `/usr/src/linux/Documentation/...`

## 9e—Set Additional Node Kernel Arguments

OPTIONAL—You can specify additional arguments to append to the node kernel command line.

To specify additional kernel arguments:

1. Un-comment and edit this line in `/etc/sicortex-system.conf`, as described in *Guidelines—Editing Configuration Files* on page 60:

```
sc1.boot.append-kargs = initcall_debug
```

2. You may add any additional kernel arguments to the list, separated by spaces. For example:

```
sc1.boot.append-kargs = initcall_debug sched_debug
```


## 9f—About Configuring Networking

For the background, see *Concepts—Nodes as Network Devices* on page 35.

The following sections explain how to edit `sicortex-system.conf` to configure the following network devices:

- Head node
- Router nodes
- Default router
- Routes to specific external networks

### About Configuring Storage

 *Do not* use `sicortex-system.conf` to configure nodes to talk to storage devices. For specifics on configuring I/O for storage devices, see:

Chapter 9 - *NFS File Systems* on page 173

Chapter 10 - *Direct-Attached File Systems* on page 177

Chapter 11 - *Lustre File Systems* on page 183



## 9g—Configure a Head Node

The `sc1.cluster.head-node` keyword identifies the head node for the system. This keyword defaults to `sc1-m3n6` if you copy the `scicortex-system.conf.example` file as shipped.

For concepts and background information, see:

*Concept—Configuring Nodes for Networking* on page 35

*Concept—Head Node* on page 38

*I/O-Ready Nodes* on page 120

*Designating a Head Node* on page 124.

```
# -----
# Configuration for specialized nodes, ie head nodes etc

#
# Which node is the head node, ie has external ethernet interfaces, users are
# expected to log in to it etc. If not set, there is no designated head node.
#
# IMPORTANT! When specifying a head node, you must also configure their
# network interfaces, in the network configuration section, below.

# sc1.cluster.head-node = sc1-m3n6
```

There are several steps to creating a viable head node:

1. Select an appropriate IO-ready node as the head node.

The default is `m3n6`.

2. Ensure that the appropriate port from the selected node has been connected to a switch on your LAN.

SC5832, SC1458, SC648—If you chose Node 6, connect one of the two Ethernet ports on the front of the module to your site LAN as follows:

- Ethernet port 1 connects to the eth1 interface on `mxn6`
- Ethernet port 2 connects to the eth0 interface on `mxn6`

If you chose Node 0, 1, or 3, connect the PCIe Express Module port to a switch on your LAN.

The physical connection makes the module and its nodes externally visible on your LAN.

3. Configure the head node in the `sc1cortex-system.conf` file using the `sc1.cluster.head-node` setting.
4. Configure the external interface for the head node, as explained in

*9j—Configure Network Interfaces for Exterior Nodes* on page 75

For information about wiring your system at installation time to enable the creation of a head node, see “Cabling the Processor Modules” in the *Hardware Installation Guide*.

### Renaming Your Head Node

OPTIONAL—You may want to name your head node something other than its native SiCortex designation, for example, you might want to rename it `HeadNode`, instead of `scx-m3n6`.

Several steps are required to rename your head node so that all entities in the SiCortex system and on your network know it by the new name.

See *Renaming Your Head Node* on page 125.

## 9g—Configure Router Nodes

STRONGLY ADVISED—For the concepts and rules for configuring router nodes, and the consequences of omitting them, see *Concept—Router Nodes* on page 39.

The sample file specifies `m2n1` and `m3n1` as the router nodes, as shown:

```
#
# Which nodes are router nodes. The key name is an historical artifact; this
# list should include only *network* I/O nodes, not storage I/O nodes.
#
# IMPORTANT! When specifying IO node(s), you must also configure
# their network interfaces, in the network configuration section, below.
#
# sc1.cluster.io-nodes = sc1-m2n1, sc1-m3n1
#-----
```

1. Un-comment and edit this line. For details, see *Guidelines—Editing Configuration Files* on page 60:

```
sc1.cluster.io-nodes = sc1-m2n1, sc1-m3n1
```

- Change the node list to specify all the nodes you will actually be using as your router nodes. Note that the list must be delimited by “, “ (comma followed by a space).

## 9h—Configure a Default Router

You must specify a default router here, even if you usually get that information from DHCP.

For the rules and reasons, see *Concept—Configuring a Default Router* on page 39.

```
#-----
# Network configuration

# Specify the default gateway for the cluster
# If "ssp" is specified, the SSP is used as the default gateway. Otherwise,
# traffic from internal nodes is distributed across the router nodes.
#
# sc1.cluster.default-router = gw.example.com
```

1. In the section of `sicortex-system.conf` shown above, un-comment the `<partition>.cluster.default-router` line, and edit according to *Guidelines—Editing Configuration Files* on page 60.
2. Configure the `sc1.cluster.default-router` keyword to point to the host name or IP address of an external router on your local network. Choose an external router already connected to your site LAN. The router you specify will serve as the default router for all traffic coming to and from the router nodes on the SiCortex system.

If you specify a default router (other than the SSP), traffic from interior nodes will be distributed across the configured router nodes, which will then forward to the designated routes.

## 9i—Configuring Extra Routes

You may, optionally, define additional routes for your system to use.

```
# Specify additional routes (optional). This is a space-separated list of CIDR
# network numbers (a.b.c.d/e). Routes to each of these will be added through
# the per-interface gateway addresses (below) - not the default router address
# (above) - and traffic from internal nodes will be distributed across the
# router nodes.
# sc1.cluster.external-network = 10.0.5.0/24 10.6.0.0/16
```

1. In the section of `scicortex-system.conf` above, un-comment the `<partition>.cluster.external-network` line, according to *Guidelines—Editing Configuration Files* on page 60.

For example:

```
scx.cluster.external-network = 10.0.5.0/24 10.6.0.0/16
```

2. Although the key name is singular, you can specify a space-separated list of networks to add multiple extra routes, if you wish.


On the head node and router nodes, the system will add these routes using the interface-specific gateway addresses, so if you also specified this line in `scicortex-system.conf`:

```
sc1.node.sc1-m1n6.eth1.gateway = 10.0.0.1
```

Then the effect would be similar to executing the following commands on `m1n6`:

```
route add -net 10.0.5.0/24 gw 10.0.0.1
route add -net 10.6.0.0/16 gw 10.0.0.1
```

On the interior nodes, these routes are added using the router nodes you previously specified as gateways. This distributes the traffic evenly among the routers, as is done for the default route.

-  Note that the interface-specific gateways used for these specific routes might not be the same as the default router defined in the previous section. This provides maximum flexibility, but using multiple external routers like this increases routing complexity and should be done with caution.

## 9j—Configure Network Interfaces for Exterior Nodes

This section describes editing the `Network configuration` section of the `sicortex-system.conf` file to configure network interfaces. For details on eligible nodes and their interfaces, see:

*Concepts—Nodes as Network Devices* on page 35

**⚠ You must configure the network interfaces in this section for:**

- **The head node** you defined above.
- **Each of the router nodes** you defined to perform IP networking.

**⚠ Do not** configure any network interfaces for storage I/O nodes or FibreChannel nodes you have connected directly to disks or disk arrays.

The section of `sicortex-system.conf` below shows the sample head node (`m3n6`) and two sample router nodes (`m2n1` and `m3n1`) that are used elsewhere in this sample file.

```
# Network configuration for exterior nodes
#
# sc1.node.sc1-m3n6.interfaces      = eth1
# sc1.node.sc1-m2n1.interfaces     = eth1
# sc1.node.sc1-m3n1.interfaces     = eth1
```

1. In the section above, enter a separate, un-commented line to define which interface you are using on each special-purpose node:
  - for the head node you chose
  - for each router node you chose

## 9k—Specify Routing Mode: *yes* or *nat*

In this section of the file, you specify the routing mode for the head node and each router node you configured above.

```
# For each router node, what kind of routing should it do? Legal values are:
#  yes - simple routing, without NAT
#  nat - routing with NAT
#  no  - do not route

# Specifying different values for the router nodes, or specifying "no" for
# any, will probably result in a system that cannot route properly or
# consistently and is strongly discouraged.

# sc1.node.sc1-m2n1.router      = nat
# sc1.node.sc1-m3n1.router      = nat
```

1. Enter an uncommented line in the `For each router node` section as shown above:

- for the head node
- for each router node you have chosen

The `<partition>.node.<node>.router` keyword enables “routing mode” in the kernel for the specified node, so that it can do routing to external entities, on behalf of the interior, non-router nodes.

2. In each line, specify *yes* or *nat* to identify how that node acts as a router. **Do not specify *no*.**

yes	Internal netblock for SiCortex system is a unique subnet mask on your LAN, and every other entity on your LAN knows how to route to that subnet mask. Route the packet straight through, without address translation.
nat	Apply Network Address Translation (NAT) to the packet, then route it. Using NAT means that addresses behind the router (ie. internal nodes) are invisible on your LAN. Other entities route packets to this router node, and the router does all the work of matching packets to source and target nodes.
no	This node does not act as a router. <i>This option is syntactically legal but will result in a system that cannot route properly.</i>

## 9I—Configure Network Interfaces

In the section of your edited `scicortex-system.conf` shown below, you configure either static addresses or DHCP to connect to the network.

You must specify `address/netmask/gateway`, or `dhcp`, for each exterior node you have defined:

- the head node
- each router node

```
# If you are using static addresses, you need to specify
# address/netmask/gateway for each interface on each node.
# The per-interface gateway address is *not* used for
# default routes, but *is* used for external-network routes.

# sc1.node.sc1-m3n6.eth1.address      = 10.4.2.27
# sc1.node.sc1-m3n6.eth1.netmask     = 255.255.0.0
# sc1.node.sc1-m3n6.eth1.gateway     = 10.4.0.1
# sc1.node.sc1-m2n1.eth1.address     = 10.4.2.28
# sc1.node.sc1-m2n1.eth1.netmask     = 255.255.0.0
# sc1.node.sc1-m2n1.eth1.gateway     = 10.4.0.1
# sc1.node.sc1-m3n1.eth1.address     = 10.4.2.29
# sc1.node.sc1-m3n1.eth1.netmask     = 255.255.0.0
# sc1.node.sc1-m3n1.eth1.gateway     = 10.4.0.1

# If you are using dhcp, you need to specify it once per interface
# on each node.

# sc1.node.sc1-m0n1.eth1.address     = dhcp
# sc1.node.sc1-m2n1.eth1.address     = dhcp
# sc1.node.sc1-m3n1.eth1.address     = dhcp

# -----
```

### Using static addresses

1. If you are using static addresses, specify a group of lines like this:

```
<partition>.node.<node>.ethX.address = <address>
<partition>.node.<node>.ethX.netmask  = <netmask>
<partition>.node.<node>.ethX.gateway  = <gateway>
```

for **each** of these:

- the head node you chose
- for *each* router node you chose


### Using DHCP

1. If you are using DHCP, add one line of the following form:

```
<partition>.node.<node>.ethX.address = dhcp
```

- for the head node
- for *each* router node

## 9m—Declare Placeholder Cards

 **This section applies only to SiCortex systems that include placeholder cards in place of one or more processor modules.**

If your SiCortex system includes placeholder cards, you must declare them in your customized `si-cortex-system.conf` file.

For the concepts, see *Concept—Declaring Placeholder Cards* on page 44.



Use of placeholder cards is supported only in certain module slots. For information on the valid combinations of placeholder cards and modules, see *Placeholder Card Configurations* on page 6.

```
# -----
# Placeholder module configuration
#
# Specify system slots that contain placeholder modules
# sc1.cluster.placeholder-modules = 1, 3
#
# Specify links disabled by the placeholder configuration
# sc1.cluster.placeholder-disabled-links = m0n3-rx2

# -----
# for 12 slot blizzard
# scx.cluster.placeholder-modules =
0,2,3,4,5,8,9,12,13,14,15,16,17,18,19,21,23,25,26,29,31,32,33,35
# scx.cluster.placeholder-disabled-links = m1n20-rx1,m10n18-rx0,m28n14-rx2,m22n13-rx0,m1n24-
rx2,m10n19-rx0,m1n13-rx0,m1n26-rx1,m10n15-rx0,m1n23-rx0,m30n18-rx0,m27n18-rx0,m20n17-
rx0,m27n15-rx0,m24n19-rx0,m1n23-rx2,m27n16-rx0,m1n23-rx2,m6n24-rx0,m1n22-rx0,m22n23-
rx0,m6n22-rx0,m24n13-rx0,m6n15-rx0,m10n22-rx0,m27n19-rx0,m34n17-rx0,m24n24-rx0,m10n16-
rx0,m11n15-rx0,m11n16-rx0,m34n23-rx0,m24n18-rx0,m34n24-rx0,m24n22-rx0,m6n14-rx2,m1n24-
rx0,m22n20-rx1,m1n21-rx2,m7n23-rx2,m28n22-rx0,m20n21-rx1,m34n22-rx0,m6n16-rx0,m30n19-
rx0,m1n18-rx0,m7n13-rx0,m11n18-rx0,m11n13-rx0,m28n17-rx0

# -----
# for 20 slot blizzard
# scx.cluster.placeholder-modules = 2,3,4,8,9,16,18,19,21,22,24,25,26,28,29,34
# scx.cluster.placeholder-disabled-links = m0n18-rx0,m12n6-rx0,m20n17-rx0,m5n25-rx0,m27n20-
rx0,m0n4-rx0,m0n3-rx0,m30n20-rx0,m5n13-rx2,m11n26-rx1,m17n13-rx2,m13n18-rx0,m13n19-rx0,m17n2-
rx0,m0n19-rx0,m31n16-rx0,m31n15-rx0,m31n22-rx0,m0n24-rx0,m32n3-rx0,m15n20-rx1,m13n3-
rx0,m20n21-rx1,m35n13-rx0,m33n22-rx0,m14n6-rx0,m5n24-rx0,m13n4-rx0,m12n17-rx0,m17n14-
rx0,m35n18-rx0,m23n22-rx0,m33n20-rx0,m32n24-rx1

# -----
# for 28 slot blizzard
# scx.cluster.placeholder-modules = 0,3,9,13,14,19,26,31
# scx.cluster.placeholder-disabled-links = m18n17-rx0,m5n25-rx0,m11n26-rx1,m30n20-rx0,m11n26-
rx1,m11n17-rx1,m11n17-rx1

# -----
# for 3 slot hail
# sci.cluster.placeholder-modules = 0,2,3,5,7,8
# no disabled links in this configuration

# -----
# for 6 slot hail
# sci.cluster.placeholder-modules = 1,4,7
# no disabled links in this configuration
# -----
```

1. In the placeholder modules section, add an uncommented line that specifies any placeholder modules on your system, according to their module position numbers, as shown in the example above.

Note: All the placeholder card configurations in the sample file are commented out. The lines appear in this document to be uncommented because they are very long, and no line-wrap syntax is used in the file.

To declare your placeholder cards, find the declaration that matches your system model and the number of placeholder cards you have. Un-comment the beginning of that declaration.

You may leave the non-applicable placeholder declarations in the file but commented out, or you may remove them from your `scicortex-system.conf` file.

2. Later, when you replace a placeholder card with a processor module, remember to update your active `scicortex-system.conf` file to remove that slot from the `scX.cluster.placeholder-modules` list.
3. When you have replaced all placeholder cards with processor modules, `scboot` automatically regenerates the `/var/state/route_info.<partition>` file, to ensure that your new processor modules will be included in the next boot of the system.

## Example—SC5832, 20 Processor Modules

The following excerpt shows how the placeholder module section would be modified for an SC5832 with 20 processor modules and 16 placeholder cards.

```
# -----
# Placeholder module configuration

# Specify system slots that contain placeholder modules
# for 20 slot blizzard
scx.cluster.placeholder-modules = 2,3,4,8,9,16,18,19,21,22,24,25,26,28,29,34

# Specify links disabled by the placeholder configuration
# for 20 slot blizzard
scx.cluster.placeholder-disabled-links = m0n18-rx0,m12n6-rx0,m20n17-rx0,m5n25-rx0,\
m27n20-rx0,m0n4-rx0,m0n3-rx0,m30n20-rx0,m5n13-rx2,m11n26-rx1,m17n13-rx2,m13n18-rx0,\
m13n19-rx0,m17n2-rx0,m0n19-rx0,m31n16-rx0,m31n15-rx0,m31n22-rx0,m0n24-rx0,m32n3-rx0,\
m15n20-rx1,m13n3-rx0,m20n21-rx1,m35n13-rx0,m33n22-rx0,m14n6-rx0,m5n24-rx0,m13n4-rx0,\
m12n17-rx0,m17n14-rx0,m35n18-rx0,m23n22-rx0,m33n20-rx0,m32n24-rx1

# -----
```

## 9n—Declare Disabled Nodes, Modules, Links

From time to time, you may have a node, module, or link that has problems, and either malfunctions or does not boot.

You must use the “Other cluster-wide configuration” section of the `sicortex-system.conf.example` file to declare to `scboot` any modules, nodes, or links that are disabled. Declaring them as “disabled” excludes them from the boot process until they can be investigated and repaired. See:

*Concept—Mapping the Interconnect Fabric* on page 42


The “Other cluster-wide configuration” section in the sample file is reproduced below. Check the version on your system for the latest sample file.

```
# -----
# Other cluster-wide configuration

# If some nodes are broken or misbehaving, they can be masked
# out of the system configuration by adding them to this list.
#
# Disabled nodes, modules and links
# sc1.cluster.disabled-nodes = m0n21, m2n23, m3n2
# sc1.cluster.disabled-modules = 1,2,3
# sc1.cluster.disabled-links = m1n20-rx0, m3n12-tx2

# -----
```

The section of `sicortex-system.conf.example` shown above is provided so that you can declare your disabled nodes, modules, and links to `scboot`.

 You may experience problems if you declare as disabled any of the designated router nodes or the nodes that serve the root file system.

### Declaring disabled nodes

1. To declare one or more disabled nodes, enter a command of this form in the `sicortex-system.conf` file:

```
<boot-partition>.cluster.disabled-nodes = mxny[, mxny]...
```

Note—Excluding a “bad” node from the boot process does not alert SLURM that the node has problems. You must also tell SLURM not to attempt to allocate the node for jobs.

### Declaring disabled modules

2. To declare one or more disabled modules, enter a command of this form:

```
<boot-partition>.cluster.disabled-modules = x[, y]...
```

Note—You may experience problems if you exclude modules that include designated router nodes or the nodes that serve the root file system.

### Declaring disabled links

3. To declare one or more disabled links, enter a command of this form:

```
<boot-partition>.cluster.disabled-links = x[, y]...
```

In `sicortex-system.conf`, you must specify the links in this format:

`mXnY-rxZ` (receiving link), or `mXnY-txZ` (transmitting link)

where X is the module number, Y is the node number, and `rxZ/txZ` specifies one of the node's six tx/rx links. For example:

```
sc1.cluster.disabled-links = m1n20-rx0, m3n12-tx2
```

In this example, `m1n20-rx0` indicates a receiving link, and `m3n12-tx2` indicates a transmitting link.

## 9o—Restoring Disabled Components

### Restoring some components

When you repair or replace a node, module, or link, you must remove it from the list of disabled nodes, modules, or links in the `sicortex-system.conf` file. Every time you reboot the system, `scboot` automatically regenerates the `route_info` file, so that only components listed as disabled will be excluded from the boot.

### Restoring all components

If you have repaired or replaced **all** disabled components, delete all the `<partition>.cluster.disabled...` commands from the `sicortex-system.conf` file.



It is no longer necessary to hand-delete the previously generated `/var/state/route_info.<partition>` file. This is because `scboot` now automatically regenerates this file on every boot. So if

you remove all the `<partition>.cluster.disabled...` commands from `sicortex-system.conf`, when the file is automatically regenerated, all nodes, modules, and links (except any still cited as “disabled”) will be included in the next boot.

## Step 10—Edit Software Installation Settings

The `scboot` utility reads the following file to retrieve the system installation settings. Editing this file is optional. Booting with an empty file in this location is the default state and is a valid boot method.

The `/etc/sicortex-install.conf` settings are specific to the current software release that is running.

You might want to edit this file if you have customized the software on the system and now want to make your customizations the default. In most cases, you should just use the default settings. Note that the settings in this file provide default values to `scboot`. You can still override the default settings with `scboot` command line arguments.

The active copy of the file is at this location:

```
/etc/sicortex-install.conf
```

The example which includes documentation of each option is in:

```
/etc/sicortex-install.conf.example
```

### 10a—Do Not Edit the File Header

```
#
# sicortex-install.conf
#
# This is the main configuration file for a SiCortex software release:
#   - Contains options specific to the SiCortex software release
#   - For options that begin with a partition, you must reboot the
#     partition for these options to take affect.

# This header is necessary for the config parser module. Do not alter.
[DEFAULT]

#
# Warning! When editing this file, use comments only at the beginning
# of a line! Trying to do something like
#
# keyword = value    # comment
#
# will not do what you want!
```

## 10b—Using a Non-Default Root File System Image


The next section of the `scicortex-install.conf.example` file specifies the root file system image:

```
#-----
# Basic cluster and boot parameters
#
# Which rootfs image we're using
#
# sc1.boot.rootfs = /opt/scicortex/rootfs/default
```

**On NBD systems** The SC1458 and the SC5832 use NBD to boot the root file system. On these systems, if you want to boot an alternate root file system image:

1. Un-comment and modify the `sc1.boot.rootfs` keyword in your copy of `scicortex-install.conf.example`, as shown above, and set it to a different rootfs image. For example:

```
# SiCortex install configuration
# See /etc/scicortex-install.conf.example
#
scx.boot.rootfs = /opt/scicortex/rootfs/my-edited-rootfs
```

**On NFS systems**  Do not modify the `sc*.boot.rootfs` variable on systems that use NFS to serve the root file system. On NFS systems, `scboot` will not be able to find the alternate root file system. On NFS systems (SC072, SC648) do the following:

1. To use a root file system image other than the default image, run `scboot` with the `-R nfs` option. Also specify the `-r` option with the path of the alternate root file system. For example:

```
ssp016 rootfs # scboot -R nfs -r /opt/scicortex/rootfs/my-edited-rootfs/
```

## 10c—Setting a Different Default Kernel

The next section of `scicortex-install.conf.example` lets you boot a non-default kernel image if you wish. Here's how it looks before you edit:

```
# Which kernel to boot.
#
# Turns into an arg to scboot
#
# sc1.boot.kernel = /opt/scicortex/kernel/linux/default
# -----
```

1. To set an alternate default kernel, add an uncommented line similar to the line above, and specify a location other than the location of the default kernel. For example:


```
sc1.boot.kernel = /opt/sicortex/kernel/linux/my-linux-kernel
```

You would want to set an alternate default kernel if you have compiled your own kernel to add hardware drivers, etc.

## Step 11—Boot the Nodes

The node root file system on the SSP contains all the software that is used to boot the nodes. After the SSP has successfully booted, log into the SSP as **root** to boot the nodes. The steps are explained in detail below.

**scboot command** The command to boot the nodes is **scboot**. This command boots the kernel on all nodes in the system.

**Must log in as root**  You must be logged in as **root** to run **scboot**.

The **scboot** command is an executable script that resides on the SSP. **scboot** performs the following functions:

- Automatically tracks the boot process using **scbootmon**
- Gathers all the boot software to be loaded onto the nodes
- Initializes the Module Service Processors (MSPs)
- Updates the node rootfs image (NBD boot mode only)
- Halts the nodes
- Initializes various SSP services needed for the boot
- Loads the Linux kernel onto the nodes
- Loads the initial ramdisk image onto the nodes
- Starts the nodes booting
- Coordinates the configuration of the node root file system
- Waits for the nodes to fully boot

## Booting the Nodes

1. Enter the `script` command to journal your console session to a file:

```
root@ssp:~> script /tmp/my-first-sboot-session.txt
```

2. As `root`, enter the following `sboot` command to test your configuration edits:

```
root@ssp:~> sboot --show_settings
```

**sboot ...  
--show\_settings as a  
pre-boot test**

The `sboot` command with the `--show_settings` option pretends to perform all its normal actions, sending the output to the console, without actually performing the boot.

**Exit from script**

3. When the output stops scrolling, type `exit` to close the console script file and stop scripting the console.

4. Check your `/tmp/my-first-sboot-session.txt` file.

Look for any errors that may indicate mistakes or omissions in your edited versions of these files:

- `sicortex-system.conf`
- `sicortex-install.conf`

5. Correct any errors in the above files.

6. Rerun your test command:

```
sboot --show_settings
```

7. Repeat the process until the output in of your `sboot` test looks good. See below for successful `sboot` output.

**Actually boot the  
nodes**

8. When you're satisfied that your configuration is set up properly, and the way you want it, rerun the same `sboot` command without the `--show_settings` option. This will actually boot the nodes:

```
root@ssp ~ $ sboot
```

Note that since the test boot does not actually boot the nodes, it tests your software configuration only. When you actually boot the nodes, you may find hardware issues.



**Successful sbboot output**

sbboot automatically tracks the boot process using the sbbootmon option. The output of a successful boot on an SC648 will look much like this:

```

root@ssp020 ~ $ sbboot
/var/state/route_info.sc1 checks out OK!

Checking Module Service Processors

Creating boot configuration

Halting all nodes

Setting up node rootfs image
Restarting evld
* Stopping evld
...
[ ok ]
* Starting evld
...
[ ok ]

Checking Master Fabric Daemon
Launching Master Clock Agent

Loading and booting linux
  bamf: Loading vmlinux
  bamf: Loading bootk
Finished loading linux (kernel boot initiated)

Waiting for global clock completion
-----sbboot-monitor-----
  secs  kernel fabric initfs slurm
    238   108   108   108   107
global clock sync complete
    250   108   108   108   108
Node boot complete
sysadmin@ssp021 ~

```

The output above shows that on this SC648, 108 nodes booted successfully, which is the full set of nodes on the system.

It is also possible that one or more nodes will have trouble booting. For example, if some nodes failed to boot due to fabric initialization problems, the last part of the output above might look like this:

```

-----sbboot-monitor-----
  secs  kernel fabric initfs slurm
    97   108   107   0   0
err: <error message reported up from fabric software>
    127   108   107   0   0

```

This output shows that the fabric initialized successfully for only 107 nodes, not 108.

`scbootmon` prints messages to `stdout`, and reports `mfd` (master fabric daemon node setup) and `fabricd` (node link setup) errors as they occur.

9. After `scboot` finishes, you can type the SLURM `sinfo` command to check which nodes have booted successfully:

While `scboot` is executing, nodes will still be going through the boot process. During this time the nodes are shown as having the states `down`, `down*` and `idle*` until they become available.

When `scboot` finishes, you can use the SLURM `sinfo` command to check if all of the nodes are finished booting. The fully booted state, where all nodes are ready to run jobs via SLURM, looks like this:

```
sysadmin@ssp ~ $ sinfo -p sc1
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
sc1        up      infinite   108   idle  sc1-m0n[0-26],sc1-m1n[0-26],sc1-m2n[0-26],sc1-m3n[0-26]
```

If you see a state like this, showing `down`, `down*` or `idle*` instead of `idle` the nodes are not currently in contact with the main SLURM controller, `slurmctld`, most likely because they are still booting:

```
sysadmin@ssp ~ $ sinfo -p sc1
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
sc1        up      infinite   108   idle* sc1-m0n[0-26],sc1-m1n[0-26],sc1-m2n[0-26],sc1-m3n[0-26]

sysadmin@ssp ~ $ sinfo -p sc1
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
sc1        up      infinite   108   idle* sc1-m0n[0-26],sc1-m1n[0-26],sc1-m2n[0-26],sc1-m3n[0-26]
```

If you run this command repeatedly (or use the `sinfo -i` flag, causing it to run continuously), you will see nodes assume the `idle` state over time. If a small set of nodes fails to reach the `idle` state significantly after the rest, these nodes may be experiencing problems.

```
sysadmin@ssp ~ $ sinfo -p sc1
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
sc1        up      infinite    1   down  sc1-m3n7
sc1        up      infinite    2   down* sc1-m0n26,sc1-m2n1
sc1        up      infinite   105   idle  sc1-m0n[0-25],sc1-m1n[0-26],sc1-m2n[0,2-26],sc1-m3n[0-6,8-26]
```

**Wait for scboot to finish**

scboot provides feedback on how many nodes have reached each stage of the boot process. It exits when the nodes are ready to be used for running jobs.

**Check the time zone on the nodes**

10. After the first time you boot the nodes, you can actually check the MSP time zone.

To show the time zone on the MSP, first telnet to the MSP on your system.

```
telnet <name-of-MSP>
```

where name-of-MSP is one of the following:

SC5832	scx-msp0
SC1458	sci-msp0
SC648	sc1-msp0

Then type this command to display the MSP's timezone file:

```
cat /etc/TZ
```

## Step 12—Test the Head Node

1. Log into the new head node, using the address you chose when you edited the `sicortex-system.conf` file:

```
# ssh <head-node-name>
```

For example:


```
# ssh jrandom.mydomain.tld
```

2. To test the head node's connectivity, use the `tracpath` command:

```
tracpath <address-of-external-entity-on-same-LAN>
```

```
tracpath <name-of-external-entity-on-same-LAN>
```

The responses should indicate that the head node connects directly to and returns directly from each of these entities with no intervening connections. This indicates the head node is visible on your network and has two-way connectivity.

 You could use the more familiar `traceroute` command above. However, the `tracpath` command provides additional useful “path MTU” information that is not provided by `traceroute`.

## Step 13—Verify Networking Works on the Nodes

In this step, you repeat on the nodes essentially the same networking checks you performed on the SSP.

1. Log into the head node from the SSP. For example, on an SC648:

```
root@ssp027# ssh <head-node-name>
```

For example:

```
# ssh jrandom.mydomain.tld
```

2. From the head node, ping several entities by IP address, and by name. You should get the same responses you received on the SSP.
3. Test that NTP is working on the node. Use the same command as on the SSP:

```
ntpq -c peers
```

However, the node should respond that it is syncing to the SSP (rather than to the external NTP server.)

## Step 14—Add User Accounts

This is a good time to add all the user accounts that are currently needed. You can add, modify, or delete user accounts at any point later, as well.

For details about configuring user accounts, see the chapter *Chapter 7 - Managing User Accounts* on page 137.

## Step 15—Reboot the SSP and Nodes and Test All Changes

At this point, you have completed all persistent configuration changes to the files that boot the SSP and the nodes.

Before turning the system over for production work, you should verify that:

- Your changes to the user configuration and boot scripts are effective and will be recreated each time the system reboots.
- The changes you implemented result in a system on which all aspects of configuration and networking work correctly and as you intended.

To re-execute and test the complete boot and configuration sequence:

1. Reboot the SSP.
2. Run all tests of configuration and connectivity on the SSP as described in previous sections.
3. Run `scboot` to reboot the nodes. `scboot` will:
  - Apply any changes from the `/etc/sicortex-system.conf` and `/etc/sicortex-install.conf` files to the booted state of the nodes
  - Implement your new, secure root password for the nodes
4. Run all tests of networking and configuration on the nodes, to verify that all aspects of the system configuration are now customized for your environment, and that networking and all services are up and running.

## Variations on Booting the Nodes

This section describes some of the additional options for how to boot the nodes on a SiCortex system.

You can use the `-h` option to display all the options for `scboot`.

For complete information about `scboot`, type `man scboot` on the SSP.

## Default Kernel Directory

When you issue the `scboot` command *without* the `-k` option, `scboot` boots the default kernel on the nodes.

Note that the `/etc/sicortex-install.conf` file can override the system default kernel setting.

In determining what kernel to boot, the `scboot -k` option

overrides the

`/etc/sicortex-install.conf` setting

which in turn overrides the

`/opt/sicortex/kernel/linux/default` symlink

## Booting the Kernel from a Non-default Directory

To boot a kernel that is not the default kernel, use the `scboot` command with the `-k` option. For example:

```
root@ssp:~> scboot -k /opt/sicortex/kernel/linux/mykernel
```

Command line arguments allow you to override the default directories used by `scboot`.

<b>scboot Command Line Option</b>	<b>Default Directory It Overrides</b>
<code>-k xxx SCBOOT_KERNEL_FILES</code>	override kernel files directory (vmlinux and System.map)
<code>-K xxx SCBOOT_KERNEL_MODULES</code>	override kernel module directory
<code>-L xxx SCBOOT_LUSTRE_MODULES</code>	override Lustre modules tree

## Booting a Non-Default File System

To boot a root file system that is not the default, use the `scboot` command with the `-r` option. For example:

```
root@ssp:~> scboot -r /opt/sicortex/rootfs/<my-root-fs>
```

The default root file system is in `/opt/sicortex/rootfs/default`, unless that default has been overridden by a setting in `/etc/sicortex-install.conf`.

## Checking Release and Component Versions Used by scboot

This section describes tools you can use to check on which release and components `scboot` is using:

**SSP release** To query which release is installed on the SSP, run either of these commands:

```
epm -q sicortex-ssp-svn
cat /etc/sicortex-release
```

## Rebooting a Single Node

From time to time, you may have a node that fails, or that does not boot properly. To fix this problem, you can reboot a single node while the other nodes are still running.

The `--nodes` option lets you reboot a single node. You may specify a single node only:

```
scboot --nodes scboot --nodes=mXnY
```

## Executing Custom Boot Scripts on the Nodes

The SiCortex system provides a special directory where you may optionally put your custom boot scripts.

This directory is available both on the SSP and on the nodes, but to execute scripts during boot, you must add them to this directory on the SSP:

```
/opt/sicortex/config/local.d
```

This is where any non-routing configuration should be done. In particular, this is the recommended place to mount additional external file systems (e.g. NFS or Lustre). At the appropriate point in the boot process, `scboot` scans this directory and executes any scripts it finds there, in alphanumeric order by filename.



# Chapter 4 Installing Software Releases

Periodically, SiCortex issues a full software release to provide new features and system updates. This chapter describes how to install a new software release. In this chapter:

- Checking Which Release Is Currently Installed
- Rolling Your Custom Configuration Data Forward
- Getting Access to a New Release
- Installing a New Release
- Configuring and Booting with a New Release
- Summary of install-ssp Options
- Sample install-ssp from Download Site
- Resolving Conflicts When Installing Updates
- About the n32 and n64 ABI Environments

## Checking Which Release Is Currently Installed

The file `/etc/sicortex-release` contains the release information. This file changes with every software install or update. You can look at this file to determine which version of the software is currently installed.

Example contents of `/etc/sicortex-release` file:

```
SiCortex Release 4.1.0_p4-r68 (V2.2 (R1B) Pass 4)
```

If you run the update script, be sure to run `etc-update` or `dispatch-conf`, otherwise the `/etc/sicortex-release` file will still make it appear that the old version is installed instead of the new one.

## Rolling Your Custom Configuration Data Forward

Before you install a new release, you should retrieve and save all the configuration changes and customizations you've made on your system, so that after the new release is installed, you can easily recreate them.

When you install a new SiCortex software release there are very few custom configuration changes that are automatically rolled forward to the new install.

The only configuration that is maintained automatically is the configuration contained in `/etc/sicortex-system.conf`. This file is a symlink to `/var/state/etc/sicortex-system.conf`. The contents of `/var/state` are preserved across installs of new releases (unless you repartition the install disk), and are shared between all installed software releases.

## Frequently Customized Files

See Appendix E—*Customized System Files— Checklist* on page 277 for a list of the system files you are likely to have modified in configuring your system. You may also have used it to record the location of a backup copy of each file you modified.

The process of rolling forward custom configuration changes can happen before rebooting or after rebooting or some combination of both. However, before rebooting, it is important to at least roll forward SSP networking configuration changes (`/etc/conf.d/net`) because otherwise the system may reboot into a state that is not accessible on the network.

To roll forward other custom configuration changes, it is helpful if you have kept a record of all configuration changes that have been made to the system. It is also possible to query portage (the Gentoo package manager) to determine if certain configuration files have changed after the initial installation. The `epm` and `equery` commands can be used to query the portage database. See *Managing Installed Software* on page 227 for more information.

Here are some examples of commands that can be run against an existing installation to determine what files have changed:

- List all packages that own files in `/etc` on the SSP. There are about 90 packages that own files in `/etc`.

```
root@ssp ~ # epm -qf /etc
openssl-0.9.8g
cyrus-sasl-2.1.22-r2
...
```

- List all packages that own files in `/opt/sicortex/rootfs/default/etc` (the node `/etc`). There are about 60 packages that own files in the node's `/etc`.

```
root@ssp ~ # scepms -qf /etc
```

```
openssl-0.9.8g
syslog-ng-1.6.12-r1
...
```

- For each of the packages above, determine if any of the files that are owned by that package have changed since being installed:

```
eprn -V <PKG_NAME>
```

```
sceprn -V <PKG_NAME>
```

- For example, `sicortex-configfiles`, one of the packages that installs, specifies a number of the configuration files that are critical to system operation.

```
# eprn -V sicortex-configfiles
..5....T /etc/pam.d/system-auth
.....T /etc/init.d/netctl
..5....T /etc/openldap/slapd.conf
```

The '5' means that the `md5sum` (checksum) of the file is different from when the package was installed. This means it is likely that the system administrator has modified it. The T just means the time-stamp on the file is different (T by itself is not very important).

Since you install the new release in a new boot disk partition, after you install the new release you can mount the disk partition for the previous release, and copy from it any customized files that are not in `/var/state`.

So in the above example, `/etc/pam.d/system-auth` and `/etc/openldap/slapd.conf` are different from when they were installed.

```
# sceprn -V sicortex-configfiles
..5....T /etc/fstab
.....T /etc/runlevels/boot/clock
```

In the above example `/etc/fstab` is the only file listed as changed. The actual location of this file from the SSP's perspective is:

```
/opt/sicortex/rootfs/default/etc/fstab
```

## Getting Access to a New Release

This section describes how to get the new release and what its component parts are. See the *Release Notes* for a description of the release features and issues.

## Delivery Methods for New Software Releases

SiCortex provides each new software release in both of the following ways:

- **Web**—Via download from this location on the web:

`http://downloads.sicortex.com/release/<VER>`

<VER> is the release version. To use the most recent release you can specify `latest` as the version.

- **DVD**—On a Release DVD

The SiCortex release files on the DVD are located at:

`/mnt/cdrom/sicortex`

# Assuming the DVD is mounted to `/mnt/cdrom`

## Layout of Release Software Files

The layout of release files whether on the DVD or the download site is as follows:

<code>ssp-&lt;REL_VER&gt;.tgz</code>	The SSP release tarball
<code>ssp.tgz</code>	Symlink to SSP release tarball
<code>portage-&lt;REL_VER&gt;.tgz</code>	Portage tree seed tarball
<code>portage.tgz</code>	Symlink to portage tree seed tarball
<code>install-ssp</code>	SSP Install script
<code>doc/</code>	All shipped documentation

The release DVD also includes a basic Gentoo Linux rescue/install image and is fully bootable, so it can be used to recover a system whose software has been corrupted or compromised.

## Disk Partitions for Installing Releases

The SiCortex SSP comes pre-configured with three 40GB disk partitions that are available for installation of separate SiCortex software releases.

The following table lists the current SiCortex partition layout:

Partition	Size(GB)	SC072, SC648	SC5832
<code>/boot</code>	1	<code>/dev/sda1</code>	<code>/dev/cciss/c0d0p1</code>
<code>swap</code>	1	<code>/dev/sda2</code>	<code>/dev/cciss/c0d0p2</code>
<code>/ (1)</code>	40	<code>/dev/sda5</code>	<code>/dev/cciss/c0d0p5</code>
<code>/ (2)</code>	40	<code>/dev/sda6</code>	<code>/dev/cciss/c0d0p6</code>

```

/ (3)      40      /dev/sda7      /dev/cciss/c0d0p7
/var/log   30/60   /dev/sda8 (30) /dev/cciss/c0d0p8 (60)

```

This illustration shows graphically how the SSP's disk is partitioned:

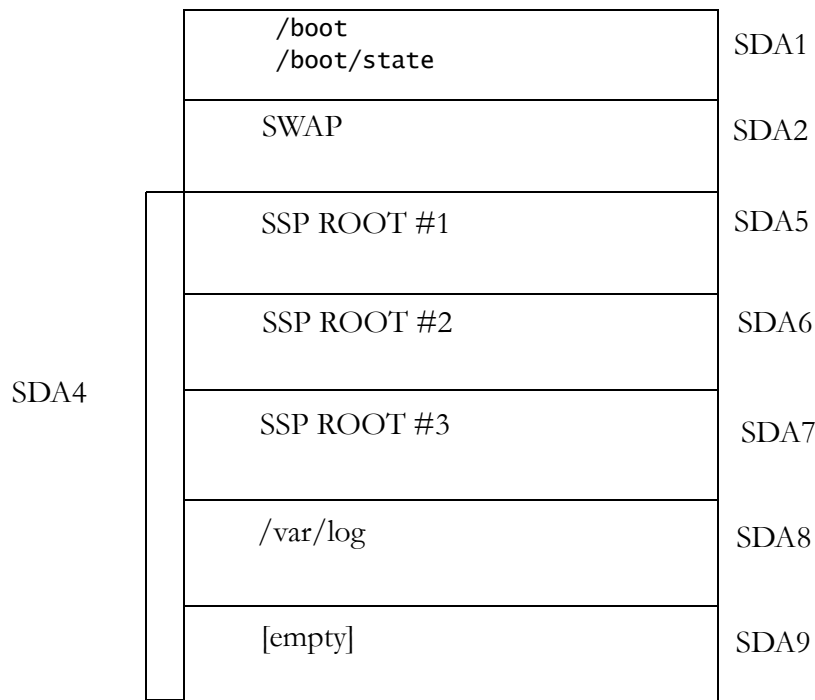


Figure 1. SSP disk partitions

## SSP Ships with Software on Partition 5

A new SiCortex system generally ships with the latest release of the software installed on disk partition 5 (`sda5`) on the SSP.

By default when you install a new release, the `install-ssp` script leaves the current release in its boot partition, and installs the new release in the next boot partition. It also increments the default boot partition to the partition where you just installed the new release.

To install in the next available boot partition and then boot from that partition, skip to the section *Installing a New Release* on page 101.

## Resetting the Default Boot Partition

Before you install a new release, or at any time, you can optionally reset the default boot partition, using the `ssp-grub-set-default` command.

1. At the SSP prompt, type:

```
ssp-grub-set-default
```

This command places you in a screen that displays all 9 combinations of boot partition and internal netblock that you can normally select in the GRUB screen when you boot the SSP. For example:

```
-----  
1: Boot sda5, 4.1.0_rc7-r63 (V2.2 (R1B) RC7), internal netblock 172.31.x.x/16  
2: Boot sda5, 4.1.0_rc7-r63 (V2.2 (R1B) RC7), internal netblock 10.137.x.x/16  
3: Boot sda5, 4.1.0_rc7-r63 (V2.2 (R1B) RC7), internal netblock 192.168.x.x/16  
4: Boot sda6, 3.1.0_p3-r62 (V2.1a (Pass 3)), internal netblock 172.31.x.x/16  
5: Boot sda6, 3.1.0_p3-r62 (V2.1a (Pass 3)), internal netblock 10.137.x.x/16  
6: Boot sda6, 3.1.0_p3-r62 (V2.1a (Pass 3)), internal netblock 192.168.x.x/16  
7: Boot sda7, 4.1.0_rc8-r64 (V2.2 (R1B) RC8), internal netblock 172.31.x.x/16  
8: Boot sda7, 4.1.0_rc8-r64 (V2.2 (R1B) RC8), internal netblock 10.137.x.x/16  
*# 9: Boot sda7, 4.1.0_rc8-r64 (V2.2 (R1B) RC8), internal netblock 192.168.x.x/16  
  
* - currently booted  
# - currently selected for next boot
```

```
Select a grub boot option (q to cancel)
```

```
-----  
The asterisk (*) indicates the combination with which the system was most recently booted. The pound sign (#) indicates the default disk partition from which scboot will boot the system, the next time you boot the nodes.
```

When the screen displays, the line with the default choice for the next boot is highlighted. `install-ssp` selects the appropriate choice automatically.

2. Notice which disk partition is a placeholder partition, or contains the oldest release, and write it down.

This is the partition you should probably choose, when you specify a partition in which to install the new release in a later step.

3. In most cases, you can just press Q to quit and exit at this point.

This is because under most circumstances, you do not need to modify that choice.

4. In the unlikely event that you needed to modify the default choice, you could use the arrow keys to move the highlight up or down, to select the line with the boot partition and netblock you want to use the next time the system is booted, but you don't need to.

When you have moved the highlight to the desired line, press ENTER to select that option and exit the screen.

## Installing a New Release

This section describes the steps for installing a new release.

### install-ssp Script

The `install-ssp` script makes installing a new software release simple. You can use this script from an SSP that you already booted normally, or you can boot the SSP directly from the release DVD.

You can use the `install-ssp` script to install a software release that you access via the file system (such as on the release DVD), or a software release that you download from the SiCortex release web site.

The following sections provide examples of using `install-ssp`.

### Methods for Installing a New Release

This section describes the various ways you can install a new release on your existing SiCortex system.

<b>If...</b>	<b>Then Use This Method to Install the New Release</b>
Your SSP and your system are healthy, and you have an install DVD from which you want to install the new release.	<i>Installing on a Running SSP from the DVD</i> on page 102. This is the fastest method, as you install directly from the media.
Your SSP and your system are healthy, and you want to download the new release from the web.	<i>Installing on a Running SSP from the Web</i> on page 103. This method is slower, because you're downloading the new release.
Your system is healthy, but: <ul style="list-style-type: none"> <li>• Your SSP has become corrupted or has crashed</li> <li>• You prefer to both boot the SSP and install from the DVD</li> <li>• You are told to use this method by SiCortex Customer Support.</li> </ul>	<i>Installing by Using the DVD to Boot First</i> on page 104

## Installing on a Running SSP from the DVD

This method is recommended because it's the fastest method for installing a new release. The system is booted from the software release that is installed on partition 5 of `/dev/sda`.

Somebody who has physical access to the system must insert the DVD disk into the SSP drive. After that, the system administrator can execute the rest of the procedure below from any workstation that is networked to the SSP, even a remote workstation.

1. Insert the SiCortex release DVD into the DVD drive on the SSP.

2. Mount the DVD:

```
mount /dev/cdrom
```

3. Use the `install-ssp` script to install the software release on the DVD to partition 6 of the disk.

To install to partition 6 of `/dev/sda` on an SC072 or SC648:

```
/mnt/cdrom/sicortex/install-ssp -d /dev/sda -r 6 \  
-b /mnt/cdrom/sicortex
```

To install to partition 6 of `/dev/cciss/c0d0` on an SC5832:

```
/mnt/cdrom/sicortex/install-ssp -d /dev/cciss/c0d0 -r 6 \  
-b /mnt/cdrom/sicortex
```

4. Copy over your customized configuration files at this point. See *Configuring and Booting with a New Release* on page 105 for details.
5. From the SSP console or workstation you're using, unmount the DVD:

```
umount /dev/cdrom
```

6. Reboot the SSP:

```
reboot
```

7. The next time you reboot the SSP and then boot the nodes, the new release will be booted automatically.

For instructions on booting the nodes, see *Step 11—Boot the Nodes* on page 85.



## Installing on a Running SSP from the Web

This example explains the most common method of installing a new release. This method assumes that your SSP and system are healthy.

You can use any workstation that is networked to your SSP to install with this method, even a workstation that is remote from the system. With the SSP still running, you download the new release from the web, and install it on your system.

In this example, the system is booted from the software release that is installed on partition 5 of `/dev/sda`. This example installs on an SC648.

1. Download the `install-ssp` script:

```
wget http://downloads.sicortex.com/release/latest/install-ssp
chmod +x install-ssp
```

2. Assuming the previous release was installed on disk partition 5:

To choose a release, go to this location with a browser to see the list of releases that are available.

```
http://downloads.sicortex.com/release
```

3. Then pick the desired release and install it on partition 6 on an SC648 with the `install-ssp` command:

```
./install-ssp -d /dev/sda -r 6 \
-b http://downloads.sicortex.com/release
```

To install the latest release on partition 6 on an SC5832:

```
./install-ssp -d /dev/cciss/c0d0 -r 6 \
-b http://downloads.sicortex.com/release
```


4. The next time you reboot the SSP and then boot the nodes, the new release will be booted automatically.

For instructions on booting the nodes, see *Step 11—Boot the Nodes* on page 85.

## Installing by Using the DVD to Boot First

### Connect Directly to the SSP

When you boot the SSP from the install DVD, the DVD does not configure your network. Therefore, you cannot use this method from a workstation.

 You must connect a monitor and keyboard directly to the SSP, as explained in:

*2a—Connect a Keyboard and Monitor to the SSP on a New System on page 53*

### Power on the SSP and Install from DVD

You must power on the SSP (lamp green instead of orange) before you can insert the DVD. The SSP is designed to boot from a DVD disk if one is present.

Follow these steps to boot the SSP from the DVD:

1. Press the power button to power on the SSP. There is a window of about 10 seconds while the SSP performs its self-test.
2. Press the DVD drive button to open the drive.
3. As soon as the drawer opens, snap in the DVD and gently press the door closed.
4. Wait for the DVD to boot with the live CD. It asks questions, but you can just wait for the questions to time out.
5. When the SSP is ready, it displays the "#" prompt on the console.
6. At the SSP prompt, log into the SSP as `root`.
7. Use the `install-ssp` script to install to partition 6 of `/dev/sda` using the release on the DVD. The `-p` option is required when booting from the DVD because the `install-ssp` script is not able to determine the system profile. Parameters for the `-p` option are:

SC072	sca
SC648	sc
SC1458	sci
SC5832	scx

So the command to install to partition 6 on an SC648 would be:

```
/mnt/cdrom/sicortex/install-ssp -d /dev/sda -r 6 \  
-p sc -b /mnt/cdrom/sicortex
```

8. It is also possible to boot from the DVD and then install from the download site. You may need to configure the system's networking first.

```
cd /tmp  
wget http://downloads.sicortex/release/latest/install-ssp  
./install-ssp -d /dev/sda -r 6 \  
-p scx -b http://downloads.sicortex/release/latest
```

9. You must reboot the SSP. To do so, type:

```
reboot
```

Be sure to remove the SiCortex release DVD during the SSP's self-test phase, or the system will boot into the install image again.

10. Now reboot the nodes, and the new release will be booted automatically.

For instructions on booting the nodes, see *Step 11—Boot the Nodes* on page 85.

## Configuring and Booting with a New Release

### First, Mount the Old Rootfs Partition

After you install a new release, the first step is to mount the disk partition that contains the previous software release. This disk partition contains the entire root file system for the previous release, including all your customized files. Therefore, it's a very useful resource when you are reestablishing your configuration after installing a new software release.

### Next, Re-Execute These Configuration and Booting Steps

Then you must redo some but not all of the steps in Chapter 3 - *Configuring and Booting the System* on page 51. Reexecute these steps:

**Step 3—Change Root Password on SSP and Nodes on page 55.**

All passwords reside in the root file system and are not shared between installed software versions. This includes the root passwords for the SSP and the nodes, and also all passwords for LDAP, `/etc/passwd`, and NIS, as well.

**Step 4—Back up the Node Root File System on page 56.**

**Step 5—Set Your Time Zone on page 57.**

After installing a new software version, you must redo configuration of `/etc/localtime` and `/opt/sicortex/rootfs/default/etc/localtime`. But the MSPs time zone configuration in `/var/state/msp/etc/TZ` is shared between installed software versions, so you do not need to redo it.

**Step 6—Configure SSP Networking on page 60.**

**Step 7—Verify That Basic SSP Networking Works on page 63.**

**Step 8—Configure SSP Firewall on page 65.**

**Step 9—Configure System and Networking Settings on page 67.**

As the first step after installing the new release, above, you will have mounted the disk partition containing the previous release. This gives you access to the complete root file system for the previous release, including your customized copies of `sicortex-system.conf` and `sicortex-install.conf`.

Reestablishing your network setup may be as simple as copying over your customized `sicortex-system.conf` file into the rootfs for the new release.

However, before you simply port your customized `sicortex-system.conf` file directly into the root file system for the new release, you should run the following check to find out if the default file has changed:

1. Run a file difference program (for example, `diff`) of the following two files:

`sicortex-system.conf.example` (unedited) from the previous release

`sicortex-system.conf.example` (unedited) from the new release

The output will tell you whether any settings have been added to or deleted from this important system file, for the new release.

2. If there have been changes, edit your customized `sicortex-system.conf` file so that it includes any new required settings or changes.

This ensures that you are configuring everything that is required for the new release, and that your networking still works the way you set it up to work.

***Step 10—Edit Software Installation Settings on page 83***

If you customized the `sicortex-install.conf` file for the previous release, run a diff of the old and new example versions of the file, similar to Step 9 above, to test whether the system default version of the file has changed. If it has, edit your customized version to include any required system changes.

***Step 11—Boot the Nodes on page 85***

***Step 12—Test the Head Node on page 89***

***Step 13—Verify Networking Works on the Nodes on page 90***





***Step 14—Add User Accounts on page 90***

After you install a new release, you must recreate all your user accounts. Whether you use LDAP, passwd files, or NIS to manage user accounts, all user account information is stored in a part of the root file system for the current release that is not shared automatically with the previous or next software release.

Note also that SLURM configuration information resides in the SSP rootfs. You must recreate any customizations you have made to the system's SLURM configuration after you install a new software release. This includes any changes or additions you have made to partition definitions and partition settings.

***Step 15—Reboot the SSP and Nodes and Test All Changes on page 91***

## Summary of install-ssp Options

Option	Action
-p <partition>	Specify the root partition on this system ( <i>sca</i> , <i>sc1</i> , <i>sci</i> , or <i>scx</i> ).
-d /dev/cciss/c0d0	Specify the boot disk on an SC5832.
-r <partition>	Select the disk partition in which to install the release.
-s <path-to-SSP-code-tarball>	
-b -b /mnt/cdrom/sicortex -b http://downloads.sicortex/release/latest	<p>Specifies the location of a set of release files:</p> <pre> /mnt/cdrom/sicortex      # install from the DVD http://&lt;download-site&gt; # install from the SiCortex                         download site </pre> <p>If you omit the -b option, the install-ssp command checks to see if the /mnt/cdrom/sicortex/ssp.tgz file is present. If so, it prompts to ask if you want to install from /mnt/cdrom/sicortex.</p> <p>The diagnostics are stored in a separate file in the same base directory, for example, /mnt/cdrom/sicortex/diags.tgz. If install-ssp finds this diagnostics file in the base directory, it unpacks and installs the diagnostics.</p>
--verbose -v	Show verbose output.
--pretend	Display the actions that would be performed without the --pretend option, but do not perform them.
--noninteractive	<p>Automatically answers yes to all interactive prompts.</p> <p> Be careful when using this because many of the prompts are warnings.</p>
--repartition	<p>Repartition and reformat the entire disk. <b>WARNINGS:</b></p> <p> <b>All data and installed software releases will be lost.</b></p> <p> <b>Use the --repartition option only in cases where the SSP has become corrupted, or has been compromised.</b></p> <p> <b>Note that none of the configuration files are saved, not even those in /var/state, if you use the --repartition option of the install-ssp command, which reformats and repartitions the entire disk.</b></p>
--keep-boot	Keep the current grub boot option/partition. The default action is to switch the default boot partition to the one that was just installed.

## Sample install-ssp from Download Site

The following example shows an `install-ssp` command that downloads version 2.2 from the SiCortex download site, and installs it to disk partition 7. The command launches an interactive process, to which you provide answers, as shown below.

```

ssp # ./install-ssp -r 7 -b http://downloads.sicortex.com/release/V2.2
===>install-ssp> Continue using installed profile 'sc' (y/n)? y
===>install-ssp> Detected 2 disks: /dev/sda /dev/sdb
===>install-ssp> Continue using DISKDEV '/dev/sda' (y/n)? y
===>install-ssp> No --hostkeys, reuse installed ssh host keys (y/n)? y

===>install-ssp> Formatting /dev/sda7
===>install-ssp> Creating directory structure in /mnt/gentoo (/dev/sda7)
===>install-ssp> Downloading (wget) http://downloads.sicortex.com/release/V2.2/ssp.tgz
14:56:33 URL:http://downloads.sicortex.com/release/V2.2/ssp.tgz [2056366369/2056366369] ->
"/mnt/gentoo/ssp.tgz" [1]
===>install-ssp> Downloading (wget) http://downloads.sicortex.com/release/V2.2/portage.tgz
14:56:35 URL:http://downloads.sicortex.com/release/V2.2/portage.tgz [51973718/51973718] ->
"/mnt/gentoo/portage.tgz" [1]
===>install-ssp> Copying /tmp/hostkeys.23995.tgz
===>install-ssp> Unpacking software release to /mnt/gentoo
.....
===>install-ssp> Unpacking portage tree to /mnt/gentoo/usr/portage
..
===>install-ssp> System specific setup (fstab, profile, ethers, etc)
===>install-ssp> Setting up grub.conf
===>install-ssp> Doing miscellaneous file cleanup
===>install-ssp> Installing grub
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(hd0) /dev/sda
===>install-ssp> Generating ssh DSA key for root
Generating public/private dsa key pair.
Your identification has been saved in /mnt/gentoo/root/.ssh/id_dsa.
Your public key has been saved in /mnt/gentoo/root/.ssh/id_dsa.pub.
The key fingerprint is:
f7:f4:7a:1b:cc:93:c7:79:26:21:54:69:59:87:5e:12 root@ssp020
===>install-ssp> Creating /etc/ssh/ssh_known_hosts with node RSA keys
===>install-ssp> Logging to /mnt/gentoo/var/log/install-ssp.log
===>install-ssp> Unmounting filesystems
Install complete

```

## Resolving Conflicts When Installing Updates



This section applies only to installing an update release, not to installing a complete new release in a separate boot partition.

 This section applies only to SSPs running Gentoo Linux. It does not apply to SSPs running Red Hat Linux.

When you run a SiCortex release update script to install an update release, you may encounter messages that say you have conflicts between existing files and new files, that need to be resolved. If there are files that have changed and need to be merged by hand, the update script asks you this question:

```
Interactively resolve now with etc-update (y/n)?
```

If you enter Y for Yes, then the system runs `etc-update` for you. If you enter N for No, then you see this help message:

```
Please resolve configuration updates before rebooting using:
CONFIG_PROTECT="-* /etc /opt/sicortex/rootfs/default" etc-update
or:
CONFIG_PROTECT="-* /etc /opt/sicortex/rootfs/default" dispatch-conf
```

There are several utilities you can run to resolve these conflicts. The system messages offer you `etc-update`, which is the default conflict resolution tool. Alternatively, you could use `dispatch-conf`. (You could also use `cfg-update`, but it is not installed by default). You can run one of these tools manually, and it allows you to see what configuration files are being changed and what is changing in each file.

If you upgrade a Gentoo package and there are configuration file updates, then Gentoo's portage system will put the new configuration file (provided by the new version of the package) in a temporary location.

The `etc-update` command is an interactive command line tool to list and resolve these configuration updates. You can run `etc-update` on the nodes by invoking it directly as `root`:

```
~ # etc-update
```

If you are running on the SSP then you can resolve updates for both the SSP and the nodes with this command:

```
~ # CONFIG_PROTECT="-* /etc /opt/sicortex/rootfs/default" etc-update
```



## About the n32 and n64 ABI Environments

By default, the compilers and all other tools on the system use the n64 Application Binary Interface (ABI). You can optionally install the n32 ABI on the nodes, in addition to n64.

For a description of the n32 and n64 ABIs, and the procedure for installing the n32 ABI on the nodes in addition to the n64 environment, see Chapter 16 - *Managing Installed Software* on page 227.

For a description of the n32 and n64 Application Binary Interfaces, see Chapter 10 - Understanding the Application Binary Interfaces, in the *Programming Guide*.



## Chapter 5 System Services

This chapter describes the system services that are started by default on the SSP, by the SSP boot process. It also provides other information about working with system services. In this chapter:

- Services Started by Default on the SSP
- What Services Are and Are Not Running?
- Removing a Default System Service
- Notes on DNS Server

To see a list of the services that are loaded on the SSP by default when it boots:

```
sysadmin@ssp # rc-update -s default
```

### Services Started by Default on the SSP

The services that are currently started on the SSP by default are explained in the following table.

Service	Purpose	Status
apache2	Will be required in a future release, for monitoring and web-based administration.	Not required for this release. May be turned off.
autofs	Supports LDAP autofs maps.	Not required. May be helpful for mounting site file systems.
conserver	Implements the console command, and makes it possible to talk to the console on any node.	Required
dnsmasq	Provides DHCP, DNS, and tftp service.	Required
envmond	Monitors system environmental sensors.	Required

Services Started by Default on the SSP

Service	Purpose	Status
ev1d	An event-coordination facility, used to coordinate the setup of the NBD mounts. It should be started on the SSP when it boots. If this service does not start, start it by hand.	Required
fabricd	Monitors system logs and reports fabric problems.	Required
kernmond	Monitors system logs and reports kernel panics and memory ECC errors	Disabled in R1
local	SSP-specific startup.	Required
net.ctl	Brings up CTLnet.	Required
net.mgt[0 - 3]	Brings up MGTnet. One process, on an SC648: <ul style="list-style-type: none"> <li>• net.mgt0 for sc0 in left bay</li> <li>• net.mgt1 for sc1 in right bay</li> </ul> Four processes, on an SC5832: <ul style="list-style-type: none"> <li>• net.mgt0</li> <li>• net.mgt1</li> <li>• net.mgt2</li> <li>• net.mgt3</li> </ul>	Required
net.site	Brings up SITEnet.	Required
netmount	Mounts any network mounts defined in fstab. Mounts NFS mountpoints as client.	Required
nfs	Required by /opt/sicortex/config and also by /opt/sicortex/rootfs.	Required
nscd	Name Service Caching daemon.	Required
ntp-client	Network Time Protocol Client	Required
ntpd	Network Time Protocol Daemon	Required
policyd	Monitors system hardware and software status, and sends alerts to Nagios.	Required
portmap	Necessary for mounting on the SSP (autofs, fstab defined, or manual, etc.) Used by NFS/netmount.	Required
postfix	Supports email alerts.	Required for email alerts. Otherwise optional.

Service	Purpose	Status
scconserver	Program that serves as an intermediary to conserver. Provides one connection to each MSP by multiplexing 27 node consoles over one channel.	Required
slapd	LDAP server.	Required
slurmctld	SLURM control daemon.	Required
sshd	Secure Shell daemon. Supports remote access.	Required
syslog-ng	Logs messages to syslog.	Required
vixie-cron	Runs crontab jobs.	Optional
watchdogd	Checks that policyd and envmond are running.	Required
xinetd	Daemon that provides time for the MSPs.	Required

## What Services Are and Are Not Running?

If a service on the SSP does not appear to be working correctly, you can use the `runlevel` and `rc-status` commands to see what services are in the current runlevel, and which of those services are stopped:

1. First give the `runlevel` command to verify the current runlevel:

```
ssp ~ # runlevel
N 3
```

2. Check to see if there were any warnings or services that were not started. This will display some information and any services that are not up will have the word “stopped” next to them. For example, `conserver` is stopped in the output below:

```
ssp ~ # rc-status
Runlevel: default
apache2      [ started ]
autofs       [ started ]
conserver    [ stopped ]
dnsmasq      [ started ]
local        [ started ]
net.ct1      [ started ]
net.mgt0     [ started ]
net.site     [ started ]
netmount     [ started ]
nfs          [ started ]
nscd         [ started ]
ntp-client   [ started ]
ntpd         [ started ]
```

## Services Started by Default on the Nodes

```
portmap      [ started ]
postfix      [ started ]
sconserver   [ started ]
slapd        [ started ]
slurmctld    [ started ]
sshd         [ started ]
sshd.admin   [ started ]
syslog-ng    [ started ]
vixie-cron   [ started ]
xinetd       [ started ]
```

3. Alternatively, to see a list only of those services that are stopped:

```
ssp ~ # rc-status | grep stop
```

4. To restart services, go to `/etc/init.d/`.

5. To restart `sconserver` and `conserver`, follow this example:

```
# /etc/init.d/sconserver start
```

Note that starting `sconserver` also starts `conserver`.

For more information, see the following man pages:

- `runlevel`
- `rc-status`
- `rc-update`

## Services Started by Default on the Nodes

The services that are currently started on the nodes by default are explained in the following table.

Service Name	Purpose	Status
<code>ntpd</code>	Time synchronization	Required by many applications
<code>portmap</code>	NFS port mapping	Required for NFS mounting
<code>rpc.statd</code>	NFS locking	Required for NFS locking
<code>slurmd</code>	compute node daemon	Required for use of SLURM and <code>srun</code>
<code>sshd</code>	ssh daemon	Required for network logins (i.e. on the head node)
<code>syslog-ng</code>	Node logging	Required for node logging

## Removing a Default System Service

The Gentoo Linux method for turning off a default system service is to issue the following command:

```
# rc-update del <SVC> default
```

The purpose of this command is to remove the <SVC> `rc service` script from the default run level.

## Notes on DNS Server

The SSP runs `dnsmasq` to provide DNS information to the system. The DNS server on the SSP provides the address lookup for those nodes, and reverse lookup for their IP addresses, and forwards DNS queries from nodes to other DNS servers as appropriate.

[Not yet implemented] The master configuration information for the DNS server is stored in LDAP, and DNS configuration files are generated as needed.

## Running Services Under `xinetd`

If you have services that need to run under `xinetd` on the nodes, you first need to start `xinetd` on the nodes. There are two options:

- Start `xinetd` by modifying the `rootfs` and rebooting
- Start `xinetd` one time using `srun`

### **Persistent `xinetd` on nodes**

The first method requires altering the `rootfs`. This method is persistent across reboots of the nodes, but is not persistent across the installation of a new software version:

1. Add `xinetd` to the default run level:

```
ROOT=/opt/sicortex/rootfs/default rc-update add xinetd default
```

2. Run `scboot` to reboot the nodes again.

The next time the nodes `sboot`, the `xinetd` service will be started on the nodes automatically.

**Non-persistent xinetd on nodes**

Alternatively, to start `xinetd` immediately on a running system without rebooting the nodes, use `srun`:

Use `srun` to start `xinetd` immediately on the nodes:

```
srun -p <partition> -N <nodes> /etc/init.d/xinetd start
```

For example, to run the job on all nodes on an SC648:

```
srun -p sc1 -N 648 /etc/init.d/xinetd start
```

This method is not persistent across either a node reboot or the installation of a new software version.

The services that are provided by `xinetd` are configured in individual files at `/opt/sicortex/rootfs/default/etc/xinetd.d/`. By default, most `xinetd` services start off disabled. To activate the services you must change the "disabled" flag in the file to "no". For example, in `.../xinetd.d/time-stream` you should have something like this to activate it:

```
-----  
...  
service time  
{  
    disable          = no  
...  
}
```

In order to make this change take effect you must restart the `xinetd` service itself either by rebooting the nodes or by issuing the following `srun` command:

```
srun -p <partition> -N <nodes> /etc/init.d/xinetd restart
```



# Chapter 6 Providing Access to System Resources

The SiCortex system provides an abundance of resources that users can employ to run applications, process data, and get results. In this chapter:

- Primary System Resources
- Using Nodes for Computing and I/O
- Separating I/O Nodes from Compute Nodes
- Designating a Head Node
- Creating and Using Partitions
- Defining Partitions

## Primary System Resources

This chapter explains how to configure these primary system resources:

- **Nodes** - For computing and/or I/O
- **Partitions** - Groups of nodes for computing
- **I/O channels** - Available on nodes 0, 1, 3, and 6

System resources also include storage, both internal and external. For information about configuring and providing access to system storage, see these chapters:

- **Storage Overview**—*Managing Storage—Overview* on page 169
- **NFS**—*NFS File Systems* on page 173
- **Direct-Attached**—*Direct-Attached File Systems* on page 177
- **Lustre**—*Lustre File Systems* on page 183.
- **FabriCache**—Internal Lustre storage, configured entirely on SiCortex nodes for the MDS, the OSSes, and the clients. See *FabriCache File Systems* on page 197.

# Using Nodes for Computing and I/O

This section describes how to configure and use nodes as compute nodes or as network devices, and how to designate and use head nodes.

## Compute Nodes

Each processor module (board) in a SiCortex system has 27 nodes, each node with 6 processors. Every node is designed to be used for computing.

To define a node as a compute node, include it in the definition of a partition, then specify the partition when you submit the job with the `srun` command. This section describes the default partitions, further below.

**⚠ As a general practice, you should exclude head nodes, router nodes, and node root file system server nodes from compute partitions.** Less memory and fewer cycles are available on special purpose nodes for computing; therefore if you use them for compute jobs they will either slow the job down or be overloaded.

## I/O-Ready Nodes

On each module, only specific nodes have the hardware that allows them to be connected and used as either the head node, a router node, or a storage I/O node.

### *SC648, SCI458, and SC5832*

The large SiCortex system models all use the same 27-node processor module. Of the 27 nodes on each processor module, only the four nodes listed below can be wired and configured for I/O:

**Node 6**—Has an on-board, hard-wired, dual gigabit Ethernet port, on node `sc*-m*n6`:

Ethernet Port on Module Faceplate	Maps to Ethernet Interface
1	<code>eth1</code> on node 6
2	<code>eth0</code> on node 6

**Nodes 1, 3, 0**—Are connected to PCI Express® ports:

PCIExpress Port Number	Port Position on Module Front Panel	Connected to Node Number
PCIExpress 1	Top	sc*-m*n1
PCIExpress 2	Middle	sc*-m*n3
PCIExpress 3	Bottom	sc*-m*n0

**Default nodes for I/O** On all larger models, node 6 on each module has built-in dual gigabit Ethernet network interface circuitry, and therefore node 6 on each module is designated as the default node for I/O.

**SC648:** The default nodes for I/O on an SC648 are:

- sc\*-m0n6 - Serves the root file system to the nodes
- sc\*-m1n6 - Has built-in dual gigabit Ethernet connection
- sc\*-m2n6 - Has built-in dual gigabit Ethernet connection
- sc\*-m3n6 - Has built-in dual gigabit Ethernet connection

**SC1458:** The default nodes for I/O on an SC1458 are:

- sc\*-m0n6 - Serves the root file system to the nodes
- sc\*-m1n6 - Has built-in dual gigabit Ethernet connection
- sc\*-m2n6 - Has built-in dual gigabit Ethernet connection
- sc\*-m3n6 - Has built-in dual gigabit Ethernet connection
- sc\*-m4n6 - Has built-in dual gigabit Ethernet connection
- sc\*-m5n6 - Has built-in dual gigabit Ethernet connection
- sc\*-m6n6 - Has built-in dual gigabit Ethernet connection
- sc\*-m7n6 - Has built-in dual gigabit Ethernet connection
- sc\*-m8n6 - Has built-in dual gigabit Ethernet connection

**SC5832:** The default nodes for I/O on an SC5832 are:

- scx-m0n6 - Serves the root file system to the nodes
- scx-m2n6 - Serves the root file system to the nodes
- scx-m4n6 - Serves the root file system to the nodes
- scx-m6n6 - Serves the root file system to the nodes

- **scx-m32n6** - Lustre gateway and head node
- **scx-m34n6** - Lustre gateway

In practice, the nodes chosen for special roles will depend on how the cables are plugged in.

The "gateway" node will be whatever node is connected to the cable from the SSP mgt0 interface.

The root file system servers will be whichever nodes are connected to the cables from mgt[0 | 1 | 2 | 3].

Other special-purpose nodes, such as the head node, must be configured by the system administrator on site.

After the system comes up, you should ensure that the **sc\*-comp** partition *excludes* the head node, the router nodes, and the root file system nodes that you have configured on your system.

### **Connecting nodes for I/O**

To use Node 6 on any module for I/O, connect Port 1 (eth1) or Port 2(eth0) on the faceplate of a Module *x* to your site LAN. Node 6 on every module is hard-wired to provide dual gigabit Ethernet I/O. To perform I/O, the node also needs to be properly configured.

You have to connect Nodes 0, 1, and 3 to a PCI ExpressModule™ before you can use them for I/O:

1. Connect a PCI ExpressModule to the PCI Express port for the node.
2. Connect a network or storage device to the PCI ExpressModule.

For details, see the *Hardware Installation Guide*.

### **PCI ExpressModules supported**

Each PCI Express I/O port accepts one PCI ExpressModule™. The following vendors' PCI ExpressModules are supported:

- Marvell SK-9I22 dual-port copper gigabit Ethernet NIC
- Myricom Myri-10G-PCIE-8AE-R+E Ethernet NIC
- QLogic QEM2462 dual-port 4gbps FibreChannel HBA
- Mellanox MSEA28-2 dual-port 10-gbps Infiniband HCA

- PCI Express I/O channels can be connected to standard disk arrays and other I/O systems, and offer enough capacity to accommodate I/O systems of enormous scale.

Model	Processor Modules	PCI Express Ports per Module	TOTAL PCI Express Channels
SC648	4 (27 nodes)	3	12
SC1458	9 (27 nodes)	3	27
SC5832	36 (27 nodes)	3	108

## Separating I/O Nodes from Compute Nodes

In most cases, it's recommended to separate I/O nodes from compute nodes.

An I/O node is a node that is wired for and is used for I/O.

- It has an I/O port: either a PCIe bus or a GigE port.
- There is a device connected to that port.
- It actually does I/O to that device.

A compute node is any node used for computing by an application.

To segregate I/O nodes from compute nodes, define SLURM compute partitions that exclude the I/O nodes. For details, see *Creating and Using Partitions* on page 126.

### Why Segregate I/O Nodes?

An example illustrates why it's helpful to segregate the I/O nodes, and to exclude them from the compute partitions.

On an SC648 there are 108 nodes. Suppose you distribute your application across all 108 nodes. This means that 104 of them are busy crunching your numbers, but four of them are also doing I/O. Those four nodes will run more slowly. Suppose that you lose half the CPU cycles on those four nodes to I/O. By the numbers, you've lost 2% of your total CPU cycles to I/O.

For loosely coupled applications, that 2% is just the cost of your I/O, and the results of the application are unaffected. But there are tightly-coupled applications that require close communication and coordination between instances of the application running on different nodes. In such an application, every node has to wait until all the nodes are done with a given step. In such a case, if four of the nodes take twice as long because they are also handling the I/O, then the entire application takes twice as long to complete, and that 2% performance hit becomes a 50% hit.

It is therefore recommended that you designate certain nodes to do I/O, and define compute partitions that exclude those nodes. It is considered best practice to run compute jobs on compute partitions that exclude the I/O nodes.

## Designating a Head Node

The *head node* (sometimes called a *login node*) is the node in a partition that users log into, to run their jobs.

Each partition on a SiCortex system has a node which serves as the head node. You choose and designate the head node when you configure and boot the system.

The head node supports the following activities that are primarily executed by application developers:

- Interactively running an application
- Scheduling via commandline (if allowed)
- Checking job queue or job progress
- Compiling
- Debugging
- Interacting with the execution environment

The above tasks are not limited to the head node, but the head node is a natural and accessible node for doing such tasks. SLURM partitions should be configured to exclude the head node, to ensure that interactive use of the head nodes will not affect compute jobs.

On SiCortex systems, the head node for each partition will have an IP alias appearing on an I/O node which has direct connectivity to the customer site LAN.

## Configuring a Head Node

Any of the I/O nodes can be designated to be the head node.

To designate a head node:

1. Select a node that has the following properties:
  - has an I/O interface (node 0, 1, 3, or 6)
  - is not a member of any SLURM compute partition
  - is configured to be visible on your local-area network (LAN)
2. Perform the necessary commands to make the chosen I/O node externally visible on the system's LAN.

For detailed instructions on configuring the head node, see *9g—Configure a Head Node* on page 71.

## One Head Node Is Sufficient

It is possible to configure multiple head nodes. However, it's not required to ensure that users have convenient access to the system. One node has ample capacity to handle all user logins to the SiCortex system.

You can define a single head node for the entire system, and direct all users to log in to the same head node. Users can then use the `srun` command to submit jobs to run on compute nodes.

The `srun` command lets you specify how many nodes you want SLURM to use to run your job. `srun` determines which nodes to use.

When multiple users are logged in to a single head node, and all of them use `srun` to submit their jobs, `srun` takes care of allocating nodes to each user's job, and returning `stdout` and `stderr` to each user.

## Renaming Your Head Node

Like all other nodes on a SiCortex system, the system refers to the head node using the `scx-mynz` format.

To rename your head node, add the two scripts shown below to the `/opt/sicortex/config/local.d/` directory on the SSP. Scripts that `scboot` finds in `.../local.d/` are executed on every node at boot time.

1. Give these scripts exactly the names shown below, to ensure they execute in the order shown. `scboot` executes the scripts in `.../local.d/` according to the alphanumeric order of their names.
2. Modify the italic elements of each script to reflect your domain names and head node.

For example, if your head node is `scx-m1n6` and you want to rename it `myheadnode`, add these scripts to `/local.d/`:

`050_dns_search.sh`

```
#!/bin/bash
if [ -e /proc/scinfo ]; then
    echo "search my.domain.com my-other.domain.com my-
third.domain.com" \
        >> /etc/resolv.conf
fi
```

The `050_dns_search.sh` script assumes that `dnsmasq` (namely, the `/etc/dnsmasq-resolv.conf` file generated by the network configuration) does know about upstream DNS servers which can resolve each of the search domains. This is conventional, however, and is reasonable to expect with a proper setup.

`999_headnode_rename.sh`

```
#!/bin/bash
myname=$(hostname -s)
[ "$myname" = "scx-m1n6" ] && hostname myheadnode.my.domain.com
```

The `999_headnode_rename.sh` script should be run last (to avoid confusing any other `.../local.d/` scripts which rely on hostname checks and to be sure to have the search path as part of `/etc/resolv.conf`).

You can expand this script to rename multiple headnodes, if you wish.

## Creating and Using Partitions

The SiCortex system uses SLURM to manage resources and schedule jobs. SLURM allows you to define partitions to allocate resources to jobs. See *SLURM for job management* on page 14.

A *partition* is a group of nodes.



**SiCortex partitions not physically separate**

On many cluster systems, a partition is a physically separate set of nodes. This is not true on a SiCortex system. All the nodes in a SiCortex system are connected through a degree-3 directed Kautz network, or a similar de Bruijn graph. The nodes communicate with one another through DMA over the Interconnect Fabric, a fast network used for internode IP networking and direct user-mode communications.

The SiCortex system uses SLURM, and SLURM uses the concept of a partition to allocate nodes to jobs. Therefore, the SiCortex system implements partitions, to facilitate resource allocation using SLURM.

To run a job on a group of nodes, you specify a SLURM partition in the `srun` command. You can either use one of the default partitions that ship with the system, or create a partition that includes some other group of nodes.

**Default SLURM Partitions as Shipped**

The following table describes the SLURM partitions that are automatically defined, when you boot a SiCortex system.

System Model	Default Partition	What It Contains
SC072	sca	Root and compute partition. Contains nodes 0 - 11, ie. all nodes on an SC072.
SC648	sc1	Root partition used to boot the system. Includes all nodes on an SC648 built in the right side of the cabinet.
	sc1-comp	Includes all nodes except m0n6 (the rootfs node) and m3n6.
	sc1-comp1	Includes all nodes except m0n6 (the rootfs node) and m*n1 (Node 1 on each module). Use for compute jobs if Node 1's are I/O nodes.
	sc1-comp3	Includes all nodes except m0n6 (the rootfs node) and m*n3 (Node 3 on each module). Use for compute jobs if Node 3's are I/O nodes.
SC1458	sci	Root partition used to boot the system. Includes all nodes on an SC1458.
	sci-comp	Includes all nodes except m0n6 (the rootfs node), m1n6, and m8n6. Use for compute jobs.

System Model	Default Partition	What It Contains
SC5832	scx	Root partition used to boot the system. Includes all nodes on an SC5832.
	scx-comp	Includes all nodes except m0n6, m2n6, m4n6, m6n6, m32n6, m34n6, on the assumption that those nodes are I/O nodes and are used to serve the root file system to the other nodes.

**SLURM partition config file**

The default SLURM partitions are defined in this file on every model:

```
/opt/sicortex/config/slurm-profile-model.conf
```

where *model* is one of:

Abbreviation	Model Number
sca	SC072
sc	SC648
sci	SC1458
scx	SC5832

**Root Partition sc\*, for Administrative Use**

The root partition is the partition that contains all the nodes on the system. You specify this partition when you issue the `scboot` command to reboot the nodes.

The root partition is designed for administrative use only, and is not intended for use by users running jobs. To run jobs, users should always specify one of the comp partitions that excludes the head node, I/O gateway nodes, and rootfs server nodes.

**SC072** On an SC072, the root partition is `sca`.


**SC648** On an SC648, the root partition is `sc1`. The `sc1` partition contains all the nodes in the system, and is for administrative use. For example, the command to boot the nodes is:

```
root@ssp:~> scboot -p sc1
```

This partition is not hidden, and users can use it.

**SC1458** On an SC1458, the root partition is `sci`.

**SC5832** On an SC5832, the root partition is `scx`.

 No one should delete the root partition [ `sca|sc1|sci|scx` ].

`scboot` assumes the existence of the root partition, so you may not be able to boot the nodes if you have deleted it. As the root partition is the only partition that includes all the nodes, if you delete it, SLURM administration may produce unpredictable results.

If you delete the root partition accidentally, you must recreate it.

## Computing Partitions Should Exclude I/O Nodes

The partitions you use for computing should be configured to exclude all nodes that perform I/O, including the head node and all gateway nodes.

Each module has three nodes that are equipped with a PCI bus and can therefore be used for I/O: Nodes 0, 1, and 3.

### *SC648 Compute Partitions*

**sc1-comp1 partition** `sc1-comp1` excludes `m0n6`, the rootfs node, and Node 1 on each module to reserve those five nodes for I/O. It includes all other nodes for use in computation.

**sc1-comp3 partition** `sc1-comp3` excludes `m0n6`, the rootfs node, and Node 3 on each module to reserve those nodes for I/O. It includes all other nodes for use in computation.

SiCortex expects users to do most of their work on `sc1-comp1` or `sc1-comp3`.

SiCortex expects users to connect their storage systems to the four Node 1 PCIExpress I/O interfaces, and then run their compute jobs on `sc1-comp1` (which excludes `m0n6` and the four Node 1 nodes).

If for some reason users cannot use the Node 1 nodes (e.g. one of them is broken), they can connect their storage to the four Node 3 nodes, and then run their compute jobs on `sc1-comp3` (which excludes `m0n6` and the four Node 3 nodes).

### *SC5832 Compute Partitions*

**scx-comp** On an SC5832, the default compute partition is `scx-comp`. It includes all nodes **except** these nodes: `m0n6`, `m2n6`, `m4n6`, `m6n6`, `m32n6`, `m34n6`

**Excluding node 6** Each module also has an on-board dual gigabit Ethernet interface connected to Node 6. If you are using one or more of the Node 6's for I/O or as a head node, you will probably want to create a compute partition that excludes those nodes as well.

**Excluding head nodes** The compute partitions should also exclude all head nodes.

## **Specifying a Default Partition**

Optionally, you can designate a default partition that SLURM will use for all user jobs that do not explicitly specify a partition.

You might want to choose the partition that excludes the special purpose and I/O nodes; for example, the `scx-comp` partition on an SC5832.

The file `/etc/slurm.conf` defines the overall SLURM configuration. It includes several other files. On an SC5832, the included file `/opt/sicortex/config/slurm-profile-scx.conf` actually defines the as-shipped partitions on an SC5832, including `scx-comp`.

To designate `scx-comp` as the default SLURM partition:

1. Edit the file `/etc/slurm.conf`.
2. Add a line to the file as shown below:

```
Include /opt/sicortex/config/slurm-ControlMachine.conf
Include /opt/sicortex/config/slurm-core.conf
Include /opt/sicortex/config/slurm-profile-scx.conf

PartitionName=scx-comp default=yes
```

The `default=yes` parameter above designates `scx-comp` as the default partition to use for all SLURM jobs that do not explicitly specify a partition, even though `scx-comp` is defined in the included file `/opt/sicortex/config/slurm-profile-scx.conf`.

You could also apply the `default=yes` parameter directly to the partition definition in `/opt/sicortex/config/slurm-profile-scx.conf`.

## Viewing Existing Partitions

Use the `scontrol` command to see partitions, as shown in this example:

```
jrandom@sc1-m0n0:~>scontrol show partitions
PartitionName=sc1-4nodes TotalNodes=4 TotalCPUs=24 RootOnly=NO
  Default=NO Shared=FORCE State=UP MaxTime=UNLIMITED Hidden=NO
  MinNodes=1 MaxNodes=UNLIMITED AllowGroups=ALL
  Nodes=sc1-m0n[0-3] NodeIndices=8-11

PartitionName=Module0-all TotalNodes=4 TotalCPUs=24
RootOnly=NO
  Default=YES Shared=NO State=UP MaxTime=UNLIMITED Hidden=NO
  MinNodes=1 MaxNodes=UNLIMITED AllowGroups=ALL
  Nodes=sc1-m0n[0-3] NodeIndices=8-11

PartitionName=Module0-TwoNodes TotalNodes=2 TotalCPUs=12
RootOnly=NO
  Default=NO Shared=NO State=UP MaxTime=UNLIMITED Hidden=NO
  MinNodes=1 MaxNodes=UNLIMITED AllowGroups=ALL
  Nodes=sc1-m0n[0-1] NodeIndices=8-9

PartitionName=quick TotalNodes=4 TotalCPUs=24 RootOnly=NO
  Default=NO Shared=NO State=UP MaxTime=3 Hidden=NO
  MinNodes=1 MaxNodes=UNLIMITED AllowGroups=ALL
  Nodes=sc1-m0n[0-3] NodeIndices=8-11

jrandom@sc1-m0n0:~>
```

For more information, see the following man pages: `slurm`, `scontrol`.

## Defining Partitions

A *partition* is a set of nodes.

Nodes may appear in more than one partition. You can define as many partitions as you like, as shown below:

```
PartitionName=all   Nodes=sc1-m0n[0-3] Default=yes
PartitionName=half  Nodes=sc1-m0n[0-1]
PartitionName=quick Nodes=sc1-m0n[0-3] MaxTime=3
```

## SLURM File Defines Partitions

The system uses a hierarchy of files to configure the SLURM definitions. At the top is the configuration file `/etc/slurm.conf`. This file then includes several subfiles that reside in `/opt/sicortex/config`.

The third included file contains the persistent SLURM partition definitions. This third file's name and contents vary by SiCortex system model. The file's name is:

```
/opt/sicortex/config/slurm-profile-model.conf
```

To edit the default partition definitions, edit the file referenced by the `/etc/slurm.conf` file.

Each partition is defined by a single line in this file that:

- gives the partition name
- lists the nodes in the partition, and omits the nodes that are not included
- may provide other information about the partition

For example, on an SC648, the system partition used to boot the nodes is:

```
PartitionName=sc1 Nodes=sc1-m0n0,sc1-m0n1,sc1-m0n2,sc1-m0n3
```

## Persistent Partition Changes in Conf File

To permanently create, modify, or delete a partition, make your changes in `/etc/slurm.conf`.


### *After Changing File, Restart slurmctld*

After you edit the SLURM file that defines the default partitions, you must restart `slurmctld`. This causes the partition changes to take effect on the system.

```
# /etc/init.d/slurmctld restart
```

## Temporary Partition Changes with scontrol

To temporarily create, modify, or delete a partition, you can use the `scontrol` command as shown below.

 You must be root to issue the `scontrol` command.

**Creating a partition** To create a partition, use the `scontrol` command. For example:


```
root@ssp020 # scontrol update PartitionName=small Nodes=sc1-m0n[1-4]
```

**Modifying a partition** You can modify a partition with a command of this form:

```
root@ssp020 # scontrol update PartitionName=small Shared=yes
```

**Deleting a partition** You can delete a partition with a command of the form shown below:

```
root@ssp020 # scontrol delete PartitionName=small
```

 Changes you make with `scontrol update` will be lost when the SSP is rebooted. To make persistent changes to the SLURM configuration, edit the `/etc/slurm.conf` file on the SSP.

## Node Addressing

Each node can be specified and addressed individually. The following table describes the addressing scheme for the nodes.

System	Node Address Format	Meaning
SC5832	<b>scx-mjnz</b>	Where: <b>scx</b> indicates the SC5832 <b>m</b> stands for module <i>y</i> is an integer from 0 to 35 <b>n</b> stands for node <i>z</i> is an integer from 0 to 26
SC1458	<b>sci-mjnz</b>	Where: <b>sci</b> indicates the SC1458 <b>m</b> stands for module <i>y</i> is an integer from 0 to 8 <b>n</b> stands for node <i>z</i> is an integer from 0 to 26

System	Node Address Format	Meaning
SC648	<b>sc1-m<math>y</math>n<math>z</math></b>	Where: <b>sc1</b> indicates the SC648 mounted in the right side of the cabinet <b>m</b> stands for module $y$ is an integer from 0 to 3 <b>n</b> stands for node $z$ is an integer from 0 to 26
SC072	<b>sca-m<math>y</math>n<math>z</math></b>	Where: <b>sca</b> indicates the SC072 <b>m</b> stands for module $y$ is 0 <b>n</b> stands for node $z$ is an integer from 0 to 11

## Node Naming Notation

The nodes in a SiCortex system use a consistent naming scheme. Each node is named by appending the module number and the node number to the system name.

**SC072 node names** An SC072 has one CPU module with 12 nodes, total. The nodes are:

`sca-m0n0, sca-m0n1, ... sca-m0n11`

**SC648 node names** An SC648 has 4 CPU modules with 27 nodes each, for a total of 108 nodes. For the SC648 named `sc0`, the nodes are:

`sc1-m0n0, sc1-m0n1, ... sc1-m0n26`  
`sc1-m1n0, sc1-m1n1, ... sc1-m1n26`  
`sc1-m2n0, sc1-m2n1, ... sc1-m2n26`  
`sc1-m3n0, sc1-m3n1, ... sc1-m3n26`

**SC1458 node names** An SC1458 has 9 CPU modules with 27 nodes each, for a total of 243 nodes. On an SC1458 named `sci`, the nodes are:

`sci-m0n0, sci-m0n1, ... sci-m0n26`  
`sci-m1n0, sci-m1n1, ... sci-m1n26`  
`sci-m2n0, sci-m2n1, ... sci-m2n26`  
`sci-m3n0, sci-m3n1, ... sci-m3n26`  
`...`  
`sci-m8n0, sci-m8n1, ... sci-m8n26`



**SC5832 node names** An SC5832 has 36 CPU modules with 27 nodes each, for a total of 972 nodes. For the SC5832 named scx, the nodes are

```
scx-m0n0,  scx-m0n1,  ...  scx-m0n26
scx-m1n0,  scx-m1n1,  ...  scx-m1n26
scx-m2n0,  scx-m2n1,  ...  scx-m2n26
...
scx-m35n0, scx-m35n1, ...  scx-m35n26
```

**Network node names** In all cases, the name of a node is also its name on the network.

```
jrandom@ssp022:~>ssh sc1-m0n0
Last login: Wed May 16 10:39:52 2007 from mgt0-ssp0.scsystem
jrandom@sc1-m0n0:~>hostname --short
sc1-m0n0
jrandom@sc1-m0n0:~>
```

**Abbreviating a series of nodes** SLURM supports a range notation for abbreviating series of nodes. The following partition:

```
PartitionName=sc1 Nodes=sc1-m0n0,sc1-m0n1,sc1-m0n2,sc1-m0n3
```

could also be defined as:


```
PartitionName=test Nodes=sc1-m0n[0-3]
```



# Chapter 7 Managing User Accounts

This chapter provides information about administering user accounts. In this chapter:

- System Access and User Privileges
- Preparing to Configure User Accounts and Passwords
- Method 1 — Use an External LDAP Database
- Method 2 — Configure LDAP Stand-Alone
- Method 3 — Use passwd Files for User Accounts
- Method 4 — Use NIS on the SSP and Nodes
- Logging into the System
- Changing User Accounts—Command Summary
- Always Use ssh Instead of telnet

 **The procedures in this chapter and book do not apply to the SC072-PDS. They apply only to the SC5832, SC1458, and SC648.**

For all procedures for the SC072-PDS, see the SC072-PDS documentation.

## System Access and User Privileges

This section explains how you log in as system administrator, and how to set up access to the system for application users.

### SSP for System Administrator Only


The SiCortex system is designed to give users easy access to the nodes to run jobs.

The SSP is designed to be used as the management appliance, by the system administrator only, for administering the system.

## Users Should Log in to Head Node

Users other than the system administrator should always log into the head node to access the system.

It is strongly recommended that users should **not** be allowed to log into the SSP under any circumstances. The SSP is a management appliance. It is not intended for general usage or to be used as a file server on large SiCortex systems (SC648, SC1458, SC5832).

 The head node is normally excluded from compute partitions, the partitions that users specify when using `srun` to run jobs. Including the head node in compute partitions could result in greatly increased running times.

For more information about the default partitions and creating partitions, see *Creating and Using Partitions* on page 126.

For more information about designating head nodes, see *Designating a Head Node* on page 124.

## Login Policies

These login policies are strongly recommended:

Class of user	Allowed to log into...	Purpose	Restrictions
<b>System Administrator</b>	SSP Head node	Administer system	<b>NONE</b>
<b>Users</b>	Head node	Run applications	<b>SHOULD NOT BE ALLOWED TO LOG INTO SSP</b>

## SSP User Has Total System Access

The SSP user has total access to all parts of the system. Therefore, the system administrator should be the only user with access to the SSP.

# Preparing to Configure User Accounts and Passwords

This section describes how to configure the system so that you will be able to add user accounts and passwords, and so that those users will be recognized on all nodes. Although users only log into the head node, they need to be recognized on all nodes so that SLURM jobs can run under their names on all nodes.

## Brief Introduction to LDAP

Lightweight Directory Access Protocol, or LDAP, is commonly used for access control on Linux systems.

To configure LDAP for your site, you need to customize several different LDAP files for your site's requirements.

---

<code>/etc/openldap/ldap.conf</code>	Tells the LDAP client running on the SSP how to find the LDAP server you're using (localhost or external). Affects only the OpenLDAP utilities, such as those used for administration and debugging.
<code>/etc/ldap.conf</code>	Tells the operating system how to find user authentication information in an LDAP database.
<code>/etc/pam.d/system-auth</code>	Tells the operating system what authentication scheme(s) to use to authenticate users.
<code>/etc/nsswitch.conf</code>	Configures which services are used to find user account information such as hostnames, password files, and group files. Used by other services in addition to LDAP.
<code>/etc/openldap/slapd.conf</code>	Syncs user account data either to the local standalone LDAP database on the SSP, or to a specified external LDAP database.

---


**Advantages of LDAP** The advantages of using LDAP for user access control instead of using local passwd files are:

- LDAP provides a single point of administration for all user accounts:
- The existing site-wide LDAP server for the external LDAP method.

- The LDAP server on the SSP for the standalone method.
- Either of the LDAP methods allows you to add, modify, or delete user authentication information without rebooting the nodes. With the passwd file method, anytime you add or change any user account information, you must reboot the nodes for those changes to take effect on the system.

### System Uses LDAP on the SSP

The SiCortex system is designed to use Lightweight Directory Access Protocol (LDAP) on the SSP to set up and maintain user accounts. When the SSP boots, the LDAP server process, `slapd`, is started by default. The nodes, in turn, are configured to query the LDAP server on the SSP for user account data. This makes it possible for a single userid and password to work across the SSP and the nodes.

 Note that only the system administrator should have access to both the SSP and the nodes. The benefit for other users is that they will automatically have access to all nodes for running SLURM jobs.


If you opt to avoid LDAP and use passwd files instead to manage user accounts, see *Method 3 — Use passwd Files for User Accounts* on page 161.

### Changing the LDAP Root Password

The `/etc/openldap/slapd.conf` file holds the LDAP root password on the SSP.

As shipped, the LDAP password is **secret**.

The system administrator at the customer site should change this to a secure password.

 For security, the `slapd.conf` file must have an owner:group of `root:ldap` and permissions of `0640`.

### Four Methods to Manage User Accounts

The SSP can get the user account data from one of these places:

- Method 1—Externally, from another LDAP database at the site

- Method 2—Standalone, from the database of the LDAP server on the SSP
- Method 3—From passwd files on the SSP
- Method 4—From NIS on the SSP and Nodes Instead of LDAP

Before you can add user accounts, you must configure the system to run LDAP on the SSP, and to manage user accounts in one of the ways listed above.

The rest of this chapter describes how to configure your system to use each of the methods listed above for user authentication.

## Method 1 — Use an External LDAP Database

You can configure the LDAP server on the SSP to obtain account data from another LDAP server located elsewhere on your site. To do this, you use the `syncrpl` mechanism. This method allows you to integrate user account management on the SiCortex system with an existing LDAP server at your site.

The examples in this section use the following terms:

<i>my-ldap.my-company.com</i>	An external LDAP server that already has customer account data.
<i>sync-user</i> , with password <i>sync-user-password</i>	The <i>my-ldap</i> server must have a user account that has read access to all account data that is to be sent to the LDAP server on the SSP.

To configure the LDAP server on the SSP for `syncrpl` operation, you must edit several files on the SSP.

### Configuring the LDAP Client, Part One

**/etc/openldap/  
ldap.conf**

As the first step to configure the LDAP client, you need to edit the `/etc/openldap/ldap.conf` file. This file tells the LDAP client running on the SSP how to find the LDAP server.

**as shipped**

On the SSP, this file looks like this, as shipped:

```
# LDAP Defaults
#
# See ldap.conf(5) for details
```

```
# This file should be world readable but not world writable.
#BASE    dc=example, dc=com
#URI     ldap://ldap.example.com ldap://ldap-master.example.com:666
#SIZELIMIT 12
#TIMELIMIT 15
#DEREF   never
```

**as edited** To customize the `/etc/openldap/ldap.conf` file to your site, you must insert the the lines shown below, above the `#SIZELIMIT` line:

```
HOST my-ldap.my-company.com
BASE dc=my-company,dc=com
TLS_REQCERT allow
```

Tailor the above lines according to your site requirements:

- The `HOST` line tells the LDAP client the identity of the LDAP server.
- The `BASE` line tells the LDAP client the root of the LDAP server tree.
- The `TLS_REQCERT` (Transport Layer Security) line controls how LDAP servers and clients encrypt their traffic on the network.
- In this syntax, `dc` stands for Domain Component. In all cases where you see `dc=com`, the `com` element can be any top-level domain (TLD) appropriate to your site, such as:

```
[ com | edu | net | gov | mil ]
```

## Configuring the LDAP Client, Part Two

**/etc/ldap.conf** As the next step in configuring the LDAP client, you must edit the `/etc/ldap.conf` file. Contrary to its name, this file does not configure an aspect of LDAP. Rather, it tells the operating system on the SSP where and how to find the user authentication information that can now be found in various places, but that historically would have existed directly in the `/etc/passwd` file.

As shipped, `/etc/ldap.conf` contains over 300 lines of code. The lines you will actually edit in this file appear as shown below. (Intervening lines are omitted):

```
as shipped host 127.0.0.1
...
base dc=padl,dc=com
...
```



```
#ldap_version 3
...
#pam_filter objectclass=account
...
#pam_password crypt
...
#nss_base_passwd ou=People,dc=pad1,dc=com?one
#nss_base_shadow ou=People,dc=pad1,dc=com?one
#nss_base_group ou=Group,dc=pad1,dc=com?one
```


**as edited** Uncomment and edit these lines as follows:

```
host my-ldap.my-company.com
base dc=my-company,dc=com
ldap_version 3
pam_filter objectclass=posixAccount
pam_password crypt
nss_base_passwd ou=People,dc=my-company,dc=com
nss_base_shadow ou=People,dc=my-company,dc=com
nss_base_group ou=Group,dc=my-company,dc=com
```

### customizing your edits

Tailor the above lines to your individual site requirements.

- `pam` stands for “pluggable authentication module”
- `ou` stands for “organization unit”
- `pam_filter` specifies the class of LDAP object that contains authentication information. It may be set to either of the following depending on your site LDAP configuration:
  - `objectclass=account`
  - `objectclass=posixAccount`
- `nss_base_passwd` identifies the subtree in the LDAP database where user information other than passwords can be found.
- `nss_base_shadow` identifies the subtree in the LDAP database where user passwords can be found.
- `nss_base_group` identifies the subtree in the LDAP database where user groups can be found.
- The `ou=People` element in the above variable definitions may be absent, present, or present but set to a custom variable rather than “People”, depending on whether `ou=People` is present on your LDAP server.

 There may be other variables you need to add to the `/etc/ldap.conf` file, to ensure it is consistent with your LDAP database configuration. For example, you probably need to add:

- `nss_map_attribute uniqueMember member`

## Configuring the LDAP Client, Part 3

The last step in configuring the LDAP client requires that you edit two more files:

- `/etc/pam.d/system-auth`
- `/etc/nsswitch.conf`

### `/etc/pam.d/system-auth`

Next, you need to edit `/etc/pam.d/system-auth`. This file sets up the system so that when someone logs on, the system knows how and when to use LDAP to authenticate the user.

### `/etc/pam.d/system-auth, as shipped`

```

#%PAM-1.0
# $Id: system-auth 46497 2007-10-25 19:31:03Z joelm $

auth      required      pam_env.so
auth      sufficient   pam_unix.so likeauth nullok
auth      required      pam_deny.so

account   required      pam_unix.so

password  required      pam_cracklib.so difok=2 minlen=8 dcredit=2 ocredit=2 retry=3
password  sufficient   pam_unix.so nullok shadow use_auth tok md5
password  required      pam_deny.so

session   required      pam_limits.so
session   required      pam_unix.so
    
```

### `system-auth, as edited`

Edit the file shown above so that it reads as shown below. Edits are shown in **boldface**:

```

auth      required      pam_env.so
auth      sufficient   pam_unix.so likeauth nullok
auth     sufficient   pam_ldap.so use_first_pass
auth      required      pam_deny.so

account  sufficient   pam_ldap.so
account  sufficient   pam_ldap.so
account  required     pam_deny.so

password  required      pam_cracklib.so difok=2 minlen=8 dcredit=2 ocredit=2 retry=3
    
```

```

password sufficient pam_unix.so nullok md5 shadow use_auth tok
password sufficient pam_ldap.so use_auth tok
password required pam_deny.so

session required pam_limits.so
session required pam_unix.so
session optional pam_ldap.so

```

**/etc/nsswitch.conf** You also need to edit the `/etc/nsswitch.conf` file. As shipped, the relevant lines in the file look like this:

```

as shipped # /etc/nsswitch.conf:
# $Id: nsswitch.conf 33782 2007-03-14 13:52:53Z brooks $
...
passwd:      compat
shadow:     compat
group:      compat
...

```

**as edited** You should edit the above lines in the file, so they read as shown below:

```

passwd:      files ldap
shadow:     files ldap
group:      files ldap

```

## Configuring the LDAP Server

**/etc/openldap/slapd.conf** To configure the LDAP server, you must edit the `/etc/openldap/slapd.conf` file on the SSP.

**as shipped** This is the as-shipped version of the `/etc/openldap/slapd.conf` file. (Note that some comment lines have been rewrapped to fit this text column, for inclusion in this book.)

```

#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include      /etc/openldap/schema/core.schema
include      /etc/openldap/schema/cosine.schema
include      /etc/openldap/schema/inetorgperson.schema
include      /etc/openldap/schema/rfc2307bis.schema

# Define global ACLs to disable default read access.

# Do not enable referrals until AFTER you have a working directory

```

## Method 1 — Use an External LDAP Database

```
# service AND an understanding of referrals.
#referral      ldap://root.openldap.org

pidfile        /var/run/openldap/slapd.pid
argsfile       /var/run/openldap/slapd.args

# Load dynamic backend modules:
# modulepath   /usr/lib64/openldap/openldap
# moduleload   back_shell.so
# moduleload   back_relay.so
# moduleload   back_perl.so
# moduleload   back_passwd.so
# moduleload   back_null.so
# moduleload   back_monitor.so
# moduleload   back_meta.so
# moduleload   back_hdb.so
# moduleload   back_dnssrv.so

# Sample security restrictions
#   Require integrity protection (prevent hijacking)
#   Require 112-bit (3DES or better) encryption for updates
#   Require 63-bit encryption for simple bind
# security ssf=1 update_ssf=112 simple_bind=64

# Sample access control policy:
#   Root DSE: allow anyone to read it
#   Subschema (sub)entry DSE: allow anyone to read it
#   Other DSEs:
#       Allow self write access
#       Allow authenticated users read access
#       Allow anonymous users to authenticate
#   Directives needed to implement policy:
# access to dn.base="" by * read
# access to dn.base="cn=Subschema" by * read
# access to *
#     by self write
#     by users read
#     by anonymous auth
#
# if no access controls are present, the default policy
# allows anyone and everyone to read anything but restricts
# updates to rootdn. (e.g., "access to * by * read")
#
# rootdn can always read and write EVERYTHING!

#####
# Uncomment this to allow users to update their passwords
#####
#access to attrs=userPassword,userPKCS12
#     by self write
#     by * auth
#
#access to attrs=shadowLastChange
#     by self write
#     by * read
#
#access to *
#     by * read

#####
# BDB database definitions
#####

database       bdb
suffix         "dc=my-domain,dc=com"
```

```

checkpoint      32      30 # <kbyte> <min>
rootdn          "cn=Manager,dc=my-domain,dc=com"

# Cleartext passwords, especially for the rootdn, should
# be avoid. See slapasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw         secret

# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory      /var/lib/openldap-data

# Performance tuning
loglevel       0
# idletimeout   60 # configure for your site
cachesize      10000
concurrency    96
threads        128
conn_max_pending 2000
conn_max_pending_auth 2000

# Indices to maintain
index objectClass eq
index entryUUID eq
index uidNumber eq
index gidNumber eq
index member eq
index uid eq
index memberUid eq

#####
# Uncomment and modify the below to enable syncrepl synchronization
# with an upstream LDAP server
#####
#syncrepl rid=123
#   provider=ldap://ldap-1.example.com
#   type=refreshOnly
#   retry="30 10 120 +"
#   interval=00:01:00:00
#   searchbase="dc=example,dc=com"
#   schemachecking=off
#   bindmethod=simple
#   binddn="cn=ldapsync,ou=people,dc=example,dc=com"
#   credentials={private}
#
#updateref ldap://ldap-1.example.com

```

1. You should edit this file as follows:

suffix	"dc= <i>my-company</i> ,dc= <i>com</i> "
rootdn	"cn=Manager,dc= <i>my-company</i> ,dc= <i>com</i> "
rootpw	<i>my-strong-password</i>
provider	ldap:// <i>my-ldap.my-company.com</i>
searchbase	"dc= <i>my-company</i> ,dc= <i>com</i> "

binddn	"cn=ldapsync,ou=people,dc= <i>my-company</i> ,dc= <i>com</i> "
credentials	<i>ldapsync-password</i>

2. Uncomment and edit the following `syncrepl` section, to allow two-way synchronization with an upstream LDAP server:

For `dc=my-company`, specify the top-level root of your LDAP server.

```
#####
# Uncomment and modify the section below to enable
# syncrepl synchronization
# with an upstream LDAP server
#####
syncrepl rid=123
    provider=ldap://ldap-1.example.com
    type=refreshOnly
    retry="30 10 120 +"
    interval=00:01:00:00
    searchbase="dc=example,dc=com"
    schemachecking=off
    bindmethod=simple
    binddn="cn=ldapsync,ou=people,dc=example,dc=com"
    credentials={private}

updateref ldap://ldap-1.example.com
```

**For Openldap 2.3 only**



If your external LDAP server is using Openldap 2.3, you must also add the following lines to the `/etc/openldap/slapd.conf` file on the external LDAP server to allow `syncrepl` to work:

```
modulepath      /usr/lib64/openldap/openldap
moduleload      syncprov.so
overlay syncprov
syncprov-checkpoint 100 10
syncprov-sessionlog 100
```

The above lines are not needed if you are running Openldap 2.2.

The example shown above is for Suse Linux. The commands in this file may vary slightly for other versions of Linux.

Below is an example of a modified `/etc/openldap/slapd.conf` file.

```
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include          /etc/openldap/schema/core.schema
```

```

include          /etc/openldap/schema/cosine.schema
include          /etc/openldap/schema/inetorgperson.schema
include          /etc/openldap/schema/rfc2307bis.schema

# Define global ACLs to disable default read access.

# Do not enable referrals until AFTER you have a working
# directory service AND an understanding of referrals.
#referral        ldap://root.openldap.org

pidfile          /var/run/openldap/slapd.pid
argsfile         /var/run/openldap/slapd.args

# Load dynamic backend modules:
# modulepath     /usr/lib64/openldap/openldap
# moduleload     back_shell.so
# moduleload     back_relay.so
# moduleload     back_perl.so
# moduleload     back_passwd.so
# moduleload     back_null.so
# moduleload     back_monitor.so
# moduleload     back_meta.so
# moduleload     back_hdb.so
# moduleload     back_dnssrv.so

# Sample security restrictions
#   Require integrity protection (prevent hijacking)
#   Require 112-bit (3DES or better) encryption for updates
#   Require 63-bit encryption for simple bind
# security ssf=1 update_ssf=112 simple_bind=64

# Sample access control policy:
#   Root DSE: allow anyone to read it
#   Subschema (sub)entry DSE: allow anyone to read it
#   Other DSEs:
#       Allow self write access
#       Allow authenticated users read access
#       Allow anonymous users to authenticate
#   Directives needed to implement policy:
# access to dn.base="" by * read
# access to dn.base="cn=Subschema" by * read
# access to *
#     by self write
#     by users read
#     by anonymous auth
#
# if no access controls are present, the default policy
# allows anyone and everyone to read anything but restricts
# updates to rootdn. (e.g., "access to * by * read")
#
# rootdn can always read and write EVERYTHING!

#####
# Uncomment this to allow users to update their passwords
#####
#access to attrs=userPassword,userPKCS12
#     by self write
#     by * auth
#
#access to attrs=shadowLastChange
#     by self write
#     by * read
#
#access to *
#     by * read

#####
# BDB database definitions
#####

database         bdb
suffix           "dc=my-company,dc=com"
checkpoint       32      30 # <kbyte> <min>
rootdn           "cn=Manager,dc=my-company,dc=com"

# Cleartext passwords, especially for the rootdn, should
# be avoided. See slapd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw           my-strong-password

```

```

# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory          /var/lib/openldap-data


# Performance tuning
loglevel           0
# idletimeout      60 # configure for your site
cachesize          10000
concurrency        96
threads            128
conn_max_pending   2000
conn_max_pending_auth 2000

# Indices to maintain
indexobjectClass  eq
index entryUUID   eq
index uidNumber   eq
index gidNumber   eq
index member      eq
index uid         eq
index memberUid   eq

#####
# Uncomment and modify the below to enable syncrepl
# synchronization with an upstream LDAP server
#####
syncrepl rid=123
         provider=ldap://my-ldap.my-company.com
         type=refreshOnly
         retry="30 10 120 +"
         interval=00:01:00:00
         searchbase="dc=my-company,dc=com"
         schemachecking=off
         bindmethod=simple
         binddn="cn=ldapsync,ou=people,dc=my-company,dc=com"
         credentials=ldapsync-password

updateref ldap://my-ldap.my-company.com

```

 If you see messages in `/var/log/messages-date` complaining about doing unindexed searches, consider adding indexes to the above file for the fields LDAP is searching for.

## Restart the LDAP server on the SSP

After editing `/etc/openldap/slapd.conf`, restart the LDAP server on the SSP

```
ssp ~ # /etc/init.d/slapd restart
```

The LDAP server on the SSP should immediately begin importing account data from `my-ldap`. Account data for a few hundred users can typically be imported in less than 1 minute.

If you have a larger LDAP database, it's more time-efficient to import it as a separate procedure:

1. Export the LDAP database to a separate file.
2. Copy the file to the SSP.



3. Import the file into the SSP's LDAP database.
4. Restart the SSP's LDAP server, which will synchronize the imported LDAP data with any existing user account data it already contains.

## Check syncrepl operation

You can see the data that has been imported with the `ldapsearch` command.

```
sysadmin@ssp015:~>ldapsearch -x -h localhost
# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
[... some lines omitted ]
# search result
search: 2
result: 0 Success

# numResponses: 354
# numEntries: 353
```

If you don't see the data that you expect, `grep` the system log file `/var/log/messages-yyyymm` for `slapd` messages.

## Reboot the Nodes

1. Reboot the nodes, so that all your LDAP changes will take effect on the nodes.

```
ssp ~ # reboot
```

## Changing Account Data in an External LDAP Database

If you already have an LDAP server running in another location on your network, continue using the same procedures you already have in place for adding, modifying, and deleting user account data.

## Method 2 — Configure LDAP Stand-Alone

Every LDAP server has its own database. As system administrator, you can configure the LDAP database on the SSP to operate stand-alone, storing its own user account data in its own database. If you do not already have an LDAP server elsewhere on your system, this approach may require the least additional effort.

To configure LDAP standalone, you enter account data directly into the LDAP server on the SSP, and it then serves that data to the nodes.

You need to edit several files on the SSP, and then populate the LDAP server with account data.

1. Connect to the SSP:

```
root@ssp:~>
```

**/etc/ldap.conf** On the SSP, the first step to configuring the LDAP client is to edit the file `/etc/ldap.conf`.

**as shipped** As shipped, the relevant lines in the `/etc/ldap.conf` file look like this:

```
host 127.0.0.1
...
base dc=pad1,dc=com
...
#ldap_version 3
...
#pam_filter objectclass=account
...
#pam_password crypt
...
#nss_base_passwd ou=People,dc=pad1,dc=com?one
#nss_base_shadow ou=People,dc=pad1,dc=com?one
#nss_base_group ou=Group,dc=pad1,dc=com?one
```

Uncomment and edit the above lines as follows:

```
base dc=my-company,dc=com
ldap_version 3
pam_filter objectclass=posixAccount
pam_password crypt
nss_base_passwd ou=People,dc=my-company,dc=com
nss_base_shadow ou=People,dc=my-company,dc=com
nss_base_group ou=Group,dc=my-company,dc=com
```

**/etc/pam.d/system-auth** The second step is to edit the `/etc/pam.d/system-auth` file. This file sets up the system so that when someone logs on, the system knows how and when to use LDAP to authenticate the user.

**as shipped** This is how the `/etc/pam.d/system-auth` file appears as shipped and installed:

```

#%PAM-1.0
# $Id: system-auth 46497 2007-10-25 19:31:03Z joe1m $

auth      required      pam_env.so
auth      sufficient    pam_unix.so likeauth nullok
auth      required      pam_deny.so

account   required      pam_unix.so

password  required      pam_cracklib.so difok=2 minlen=8 dcredit=2 ocredit=2 retry=3
password  sufficient    pam_unix.so nullok shadow use_authtok md5
password  required      pam_deny.so

session   required      pam_limits.so
session   required      pam_unix.so

```

**system-auth, as edited** Edit the `/etc/pam.d/system-auth` file so that it reads as shown below. Edits are shown in **boldface**:

```

auth      required      pam_env.so
auth      sufficient    pam_unix.so likeauth nullok
auth      sufficient    pam_ldap.so use_first_pass
auth      required      pam_deny.so

account   sufficient    pam_unix.so
account   sufficient    pam_ldap.so
account   required      pam_deny.so

password  required      pam_cracklib.so difok=2 minlen=8 dcredit=2 ocredit=2 retry=3
password  sufficient    pam_unix.so nullok md5 shadow use_authtok
password  sufficient    pam_ldap.so use_authtok
password  required      pam_deny.so

session   required      pam_limits.so
session   required      pam_unix.so
session   optional     pam_ldap.so

```

**/etc/nsswitch.conf** You also need to edit the `/etc/nsswitch.conf` file. As shipped, the relevant lines in the file look like this:

**as shipped**

```

# /etc/nsswitch.conf:
# $Id: nsswitch.conf 33782 2007-03-14 13:52:53Z brooks $
...
passwd:      compat

```

```
shadow:    compat
group:     compat
...
```

**as edited** You should edit the above lines in the file, so they read as shown below:

```
passwd:    files ldap
shadow:    files ldap
group:     files ldap
```

**/etc/openldap/ldap.conf** The fourth step to configure the LDAP client requires that you edit the `/etc/openldap/ldap.conf` file.

**as shipped** On the SSP, this file looks like this:

```
#
# LDAP Defaults
#
# See ldap.conf(5) for details
# This file should be world readable but not world writable.
#BASE    dc=example, dc=com
#URI      ldap://ldap.example.com ldap://ldap-master.example.com:666
#SIZELIMIT    12
#TIMELIMIT    15
#DEREF       never
```

To customize the `/etc/openldap/ldap.conf` file to your site, add these three lines to the end of the file:

```
HOST localhost
BASE dc=my-company,dc=com
TLS_REQCERT allow
```

**/etc/openldap/slapd.conf** On the SSP, you must also configure the LDAP server. To do this, you edit the file `/etc/openldap/slapd.conf`.

**as shipped** As shipped, the `/etc/openldap/slapd.conf` file contains about 100 lines, most of them commented out.

1. Locate these lines:

```
...
suffix    "dc=my-domain,dc=com"
...
rootdn    "cn=Manager,dc=my-domain,dc=com"
...
rootpw    secret
```

...

2. Edit these lines to read as follows:

```
suffix          "dc=my-company,dc=com"
rootdn          "cn=Manager,dc=my-company,dc=com"
rootpw          my-strong-password
```

3. Uncomment the following section of the file, to allow users to update their own passwords:

```
#####
# Uncomment this to allow users to update their passwords
#####
access to attrs=userPassword,userPKCS12
        by self write
        by * auth

access to attrs=shadowLastChange
        by self write
        by * read

access to *
        by * read
```

## Restart the LDAP server on the SSP

After editing `/etc/ldap.conf` and `/etc/openldap/slapd.conf`, restart the LDAP server on the SSP:

```
ssp ~ # /etc/init.d/slapd restart
```

## Populate the LDAP Database on the SSP

At this point, you need to add users to the LDAP database on the SSP. You can do this in one of two ways, both of which are documented below. Do one of the following:

- Add hierarchical entries, then add users by hand.
- OR: Import the data from an existing LDAP database to the empty LDAP database on the SSP.

### **Create LDAP Hierarchical Entries and Add Users**

This step is required at this point, if you are creating your LDAP database from scratch.

Alternatively, you can create your standalone LDAP database by importing a copy of an existing LDAP database that resides on another system. To do that, skip this section and go to *Import a Copy of an Existing LDAP Database* on page 156, below.

Before you add accounts for individual users, you must create the basic LDAP hierarchical structure. This is a one-time change. Use the `ldapadd` command as shown below:

```
ldapadd -h localhost -x -D "cn=Manager,dc=my-company,dc=com" -w
my-strong-password << LDIF

#Create top level organization
dn: dc=my-company,dc=com
dc: my-company
o: my-company
objectClass: organization
objectClass: dcObject

#Create People organizational unit
dn: ou=people,dc=my-company,dc=com
objectClass: top
objectClass: organizationalUnit
ou: people

#Create Group organizational unit
dn: ou=group,dc=my-company,dc=com
objectClass: top
objectClass: organizationalUnit
ou: group

LDIF
```

To add new users, see *Changing User Account Data in a Standalone LDAP Database* on page 159.


### **Import a Copy of an Existing LDAP Database**

If you already have an LDAP database elsewhere, you can import its data into the new LDAP database on the SSP.

 This sub-section is intended for experienced LDAP users only.

If you have an existing LDAP database, you can use the following method to populate your LDAP database, instead of the method described above in *Create LDAP Hierarchical Entries and Add Users* on page 156.

If you would like to import data from an existing LDAP server without using Method 1 (syncrepl), then you can alternatively export the data from your existing LDAP server and import it into the SSP.

-  If you use this method, all the edits you make to the following LDAP files on this system must match the corresponding values and variables in the source database from which you import your data. If the values in these files do not match the data you import, then LDAP inquiries will fail due to non-matching entries.

Example of the procedure (adapt as needed for your site):

1. Ensure you are root on your existing LDAP server:

```
root@existing_ldap_server# slapcat -l contents.ldif
```

2. Copy contents.ldif to the SSP.

3. Import your existing LDAP data to the SSP, for example:

```
ssp ~ # /etc/init.d/slaped stop
ssp ~ # slapadd -l contents.ldif
ssp ~ # /etc/init.d/slaped start
```

## Reboot the Nodes

Reboot the nodes, so that all your LDAP changes will take effect. For example:

1. Reboot the nodes.

```
ssp ~ # sbboot -p sc1
```

## Testing Your LDAP Configuration

**Try logging in** The main test to verify you have properly configured LDAP is to attempt to log in. If you succeed, you can assume that you configured LDAP correctly.

However, if your attempt to log in fails, you'll see errors in the log files about unsuccessful login attempts. The error messages say only that attempt was made and it failed; they don't say what information was wrong or missing.

**ldapsearch** You can do an ldapsearch to see what's actually in the LDAP database. Replace the *site-ldap-server* and *searchbase* parameters with site-appropriate values.

```
# ldapsearch -h site-ldap-server -x -b searchbase
```

The following example displays the groups in the LDAP database:

```
user99@ssp:~
$ ldapsearch -h ldap-A -x -b "dc=my-company,dc=com" | tail
description: Consultant for Staff (see CEO)
shadowLastChange: 13962

# search result
search: 2
result: 0 Success

# numResponses: 448
# numEntries: 447
user99@ssp:~
```

The following example shows that if `/etc/openldap/ldap.conf` is set up correctly then you don't need to specify the `-h` option:

```
$ ldapsearch -x | tail
description: Consultant for Staff (see CEO)
shadowLastChange: 13962

# search result
search: 2
result: 0 Success

# numResponses: 448
# numEntries: 447
user99@ssp:~
$ exit
```

This example lists all the accounts in the LDAP database:

```
user99@ssp:~>ldapsearch -h localhost -x

[...]

# search result
search: 2
result: 0 Success

# numResponses: 398
# numEntries: 397
```

This example lists all the individual accounts and all the groups in the LDAP database:

```
user99@ssp:~>ldapsearch -h ldap-1 -x -b "dc=my-company,dc=com"
```



## Back up Your LDAP Database

Always keep a current backup of your LDAP database. This ensures that if it becomes corrupted for any reason, such as a device failure, you will be able to restore it easily.

## Changing User Account Data in a Standalone LDAP Database

You'll be adding, modifying, and deleting user accounts as time goes on, on an as-needed basis. This section details how to perform these tasks on a standalone LDAP server.

### Adding User Account Data

Now you're ready to add accounts for individual users.

Use the `ldapadd` command on the SSP to add account data to the LDAP server's database on the SSP. For example:

```
sysadmin@ssp:~>ldapadd -h localhost -x -D "cn=Manager,\
    dc=my-company,dc=com" -w my-strong-password <<LDIF
dn: uid=fred,ou=people,dc=my-company,dc=com
objectClass: top
objectClass: posixAccount
objectClass: shadowAccount
objectClass: inetOrgPerson
cn: Fred Flintstone
uid: fred
uidNumber: 1999
gidNumber: 100
homeDirectory: /var/tmp
sn: Flintstone
givenName: Fred
loginShell: /bin/bash
mail: fred.flintstone@my-company.com
userPassword: fred
LDIF
adding new entry "uid=fred,ou=people,dc=my-company,dc=com"
```

The `ldapadd` command expects to receive account data in LDAP Data Interchange Format (LDIF). You may write scripts to obtain account data from existing data stores, format it in LDIF, and pass it into the `ldapadd` command. SiCortex can provide advice and assistance with this, if desired.

### Modifying an Existing User Account


To update an existing user account in the standalone LDAP server database on the SSP, use the `ldapmodify` command, as shown here, with the appropriate substitutions for the items in *italics*:

The following example shows how you would modify an existing LDAP user account.

```
root@ssp:~>ldapmodify -h localhost -x -D "cn=Manager,dc=my-company,dc=com" -w my-strong-pw
<<LDIF
dn: uid=fred,ou=people,dc=my-company,dc=com
homeDirectory: /var/tmp/fred
LDIF
modifying entry "uid=fred,dc=my-company,dc=com"
root@ssp:~>
```

**Deleting a User Account** To delete an existing user account in the LDAP database, use the `ldapdelete` command, as shown in the following example, with the appropriate substitutions for the items in italics:

```
root@ssp:~>ldapdelete -h localhost -x -D "cn=Manager,dc=my-company,dc=com" -w secret <<EOF
uid=fred,ou=people,dc=my-company,dc=com
EOF
root@ssp:~>
```

 When *deleting* an account, do *not* specify `dn:` at the start of the line (as you would when modifying an account).

## Changing Group Data in a Standalone LDAP Database

This section describes how to add and delete user groups, and how to add and remove users from a group.

You must log into the SSP as root to perform all the procedures in this section that use `ldapxxx` commands.


**Adding a group** Use the `ldapadd` command on the SSP to add group data to the LDAP server's database on the SSP. For example:

```
root@ssp:~>ldapadd -h localhost -x -D "cn=Manager, \
dc=my-company,dc=com" -w my-strong-password << LDIF
dn: cn=newgroup,ou=group,dc=my-company,dc=com
objectClass: top
objectClass: posixGroup
objectClass: groupOfNames
cn: newgroup
gidNumber: 5001
member: uid=user1,ou=people,dc=my-company,dc=com
member: uid=user2,ou=people,dc=my-company,dc=com
LDIF
```

```
root@ssp: ~>
```

**Deleting a group** To delete an existing group from the LDAP database, use the `ldapdelete` command. For example:

```
root@ssp:~>ldapdelete -h localhost -x -D "cn=Manager, \
    dc=my-company,dc=com" -w my-strong-password << EOF
cn=newgroup,ou=group,dc=my-company,dc=com
EOF
root@ssp:~>
```

 When deleting a group, do not specify `dn:` at the start of the line (as you would when modifying an account).

**Adding a member to a group** To add a user to an existing group in LDAP, use the `ldapmodify` command. For example:


```
root@ssp:~>ldapmodify -h localhost -x -D "cn=Manager, \
    dc=my-company,dc=com" -w my-strong-password << LDIF
dn: cn=newgroup,ou=group,dc=my-company,dc=com
add: member
member: uid=user3,ou=people,dc=my-company,dc=com
LDIF
root@ssp:~>
```

**Removing a member from a group** To remove a user from an existing group in LDAP, use the `ldapmodify` command. For example:

```
root@ssp:~>ldapmodify -h localhost -x -D "cn=Manager, \
    dc=my-company,dc=com" -w my-strong-password << LDIF
dn: cn=newgroup,ou=group,dc=my-company,dc=com
delete: member
member: uid=user3,ou=people,dc=my-company,dc=com
LDIF
root@ssp:~>
```

## Method 3 — Use passwd Files for User Accounts

If you prefer, you can use `passwd` files instead of an LDAP server to manage user accounts on the SSP and to serve the user account information to the nodes.

-  On larger SiCortex systems when you use NBD to serve the root file system to the nodes, the nodes have a read-only root file system. One implication of this is that if you use passwd files instead of LDAP to define user accounts, those passwd files reside in the root file system. Therefore, users will require the assistance of the system administrator to change their passwords.

## Creating User Accounts in /etc/passwd

These steps describe how to create user accounts in /etc/passwd:

1. Create user accounts on the SSP, for example, with the `useradd` command:

```
ssp ~ # useradd --comment 'Fred Flintstone' --home-dir /tmp/fred --create-home fred
ssp ~ # passwd fred
New UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

2. Look at the resulting user information:


```
ssp ~ # grep fred /etc/passwd
fred:x:1136:1136:Fred Flintstone:/tmp/fred:/bin/bash
ssp sysadmin # grep fred /etc/shadow
fred:$1$Xpab43eE$i3I/1k6ULn90KoB3bBFBg/:13845:0:99999:7:::
```

3. Insert the user account line from the /etc/passwd file into the /etc/passwd file in the node root file system, above the last line in the file, which reads "+::::::" :

```
ssp ~ # vi /opt/sicortex/rootfs/default/etc/passwd
```

4. Append the user account line from the /etc/shadow file to the /etc/shadow file in the node root file system:

```
ssp ~ # vi /opt/sicortex/rootfs/default/etc/shadow
```

-  Note that user and group names, IDs and encrypted passwords must be the same on the nodes and the SSP.

5. If you are on a larger system and are therefore serving the rootfs using NBD (SC1458, SC5832) then you must reboot the nodes to propagate the change to the nodes.

```
ssp ~ # sboot
```

## 6. Create (or mount) home directories on the nodes, if desired:

```
ssp ~ # srun -N 4 /bin/mkdir /tmp/fred
ssp ~ # srun -N 4 chown fred /tmp/fred
```

## 7. Now you can log in to the SSP and the nodes with the same UID and password:

```
sysadmin@gs103:~>ssh fred@ssp
Password:
fred@ss ~ $ ssh sc1-m0n0
The authenticity of host 'sc1-m0n0 (172.31.201.200)' can't be established.
RSA key fingerprint is 8f:43:7c:e2:a4:db:6e:19:41:e0:a1:8d:d3:94:70:fa.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'sc1-m0n0,172.31.201.200' (RSA) to the list of known hosts.
Password:
fred@sc1-m0n0 ~ $ logout
Connection to sc1-m0n0 closed.
fred@ssp ~ $ ssh sc1-m0n0
Password:
fred@sc1-m0n0 ~ $ pwd
/tmp/fred
fred@sc1-m0n0 ~ $
```

## Back up Your Passwd Files

Always keep a current backup of your passwd files. This ensures that if they become corrupted for any reason, such as a device failure, you will be able to restore your user accounts easily.

## Changing User Passwords

If you are using passwd files to manage user accounts, then the method for changing a user password depends on whether you are on a smaller or larger SiCortex system.

The smaller SiCortex systems use NFS to serve the root file system to the nodes. This means the root file system is read/write, so adding users and changing user passwords is relatively simple.

The larger SiCortex systems use NBD to serve the root file system to the nodes. A side effect is that the rootfs is read-only. This means that a special procedure is required to change a user's password or to add user accounts.

There are two methods for changing user passwords on a system with an NBD rootfs. Both methods require that you reboot the nodes afterwards, to propagate the new password to the nodes.

## **chrootfs**

If you are using passwd files to manage user accounts, you must use the `chrootfs` procedure to create a read/write view of the rootfs. Then you can add a new user account or change a user's password in that read/write view. You must then reboot the nodes to propagate the change to the nodes. For the detailed procedure, see *Using chrootfs to Edit the Root File System* on page 270 in Appendix D - "Common Procedures".

If you are using `passwd` files, you can use the `scpasswd` command. See below.

## **scpasswd Command**

If you are using passwd files to manage user accounts, you can use the `scpasswd` command on the SSP to change a user's password.

1. Log into the SSP as root.
2. Use the `scpasswd` command the same way you would use the `passwd` command, to change the user's password.

The `scpasswd` command issued on the SSP edits the copy of the `passwd` file in the *node* root file system on the SSP.

Use the `man scpasswd` command to see the man page.

3. If NBD is being used to serve the root file system to the nodes, you must reboot the nodes to propagate the password change to the nodes.

If NFS is being used to serve the rootfs, the password change will be visible on all nodes immediately, without rebooting.

## **scuseradd Command**

If you are using passwd files to manage user accounts, you can use the `scuseradd` command on the SSP to add a new user.

1. Log into the SSP as root.

2. Use the `scuseradd` command the same way you would use the `useradd` command, to add a new user.

The `scuseradd` command edits the node root file system on the SSP, and adds the new user.

Use the `man scuseradd` command to see the man page.

3. If NBD is being used to serve the root file system to the nodes, you must reboot the nodes to propagate the user information to the nodes.

If NFS is being used to serve the rootfs, the new user information will be visible on all nodes immediately, without rebooting.

## Method 4 — Use NIS on the SSP and Nodes

You can use the Network Information System (NIS) instead of LDAP for managing user accounts and passwords for both the SSP and the nodes. In a typical NIS configuration, the NIS server (or servers) are available on the local site network.

Note that the nodes of a SiCortex system can add a significant load on the NIS servers, so you should ensure that they are capable of handling the added load.

To configure the system to use NIS, you must change the configuration separately for the SSP and for the SiCortex nodes.

### For the SSP

1. On the SSP, edit the file `/etc/nsswitch.conf` to include "nis" before "files" in the lines for `passwd`, `shadow`, and `group`. For example:

```
passwd:      nis files ldap
shadow:     nis files ldap
group:      nis files ldap
```

2. Edit the file `/etc/conf.d/net` to include these lines:

```
nis_domain="your-nis-domain"
nis_servers="nis-server-1 nis-server-2"
```

where *your-nis-domain* is your site NIS domain name (note: this is often different from the DNS domain name) and *nis-server-1* [and *nis-server-2*] are a list of one or more local NIS servers, separated by spaces.

3. Enable the NIS daemon, `ybind`, to start at boot time by executing this command:

```
rc-update add ybind default
```

### For the Nodes

1. On the SSP, repeat steps 1 and 2 above. But this time, make the edits to the corresponding configuration files in the node rootfs, in the following directory:

```
/opt/sicortex/rootfs/default/etc
```

2. Enable the nodes to start the NIS daemon at boot time by executing this command on the nodes, as `root`:

```
rc-update add ybind default
```

3. Reboot the nodes.

You could also start the NIS daemon by hand:

```
/etc/init.d/ybind start
```

Alternatively, you can get the effect of the `rc-update` by doing this:

```
# cd /opt/sicortex/rootfs/default/etc/runlevels/default  
# ln -s /etc/init.d/ybind .
```

## Logging into the System

The section about how to log into the system has moved to *Common Procedures* on page 269.



## Always Use ssh Instead of telnet

The system does not enable `telnet`, for security reasons. It is recommended that Secure Shell (`ssh`) be used exclusively to access the system.

## Changing User Accounts—Command Summary

On SiCortex systems, the commands and procedures you can use to add, modify, or delete user accounts depend on the following factors:

- Whether you are `root` or an application user
- Whether you are logged into the SSP or a node
- Whether you are using LDAP or `passwd` files to manage users
- Whether your system uses NFS or NBD to serve the rootfs to the nodes

This section summarizes the situations in which you can use various user account management commands, and details any required special procedures.

**Table 1. User Management Commands for LDAP**

Command	Role Required	Special Procedures and Conditions
Adding users, when using External LDAP server	root only	Use the standard LDAP methods.
Adding users, when using Standalone LDAP server	root only	Use the methods in <i>Changing User Account Data in a Standalone LDAP Database</i> on page 159.
<code>passwd</code>	root or user	Can execute on either SSP or nodes, affecting both. Effect is immediate, regardless of NBD/NFS boot mode.

**Table 2. User Management Commands for passwd Files**

Command	Role Required	Special Procedures and Conditions
<code>useradd</code>	root only	When issued on the SSP, affects the SSP itself, not the nodes. Should be used only when managing accounts with <code>passwd</code> files.

## Changing User Accounts—Command Summary

<b>Command</b>	<b>Role Required</b>	<b>Special Procedures and Conditions</b>
scuseradd scpasswd	root only	Must execute on the SSP. Affects node rootfs. If serving rootfs with NBD, must reboot nodes. Can be used only when using passwd files, not LDAP.
passwd	root or user	If executed on the SSP, affects only the SSP. If executed on a node, affects only the nodes. Effect is immediate on the SSP or on nodes booted using NFS. For nodes booted using NBD, a reboot is necessary.

# Chapter 8 Managing Storage—Overview

This chapter provides an overview of your options for managing your storage needs on a SiCortex system. In this chapter:

- NFS File Systems
- Direct-Attached File Systems
- Lustre File Systems
- FabriCache File Systems

## NFS File Systems

SiCortex systems support the use of network-attached storage (NAS) provided you use the industry-standard Network File System (NFS) protocol to communicate with clients.

Network-attached storage (NAS) systems package the file server and storage subsystem into a single device that serves one or more clients. NAS devices use Ethernet as the transport between client and server, and any of several industry-standard Network File System (NFS) protocols to communicate with clients. NAS is widely used throughout the industry.

SiCortex systems can interoperate with virtually any Network-Attached Storage (NAS) that is compliant with NFS V3 or later. All major NAS products, and Linux systems acting as file servers, meet this requirement. One exception is storage systems based on the Windows File Server kit from Microsoft, which probably do not interoperate with NAS and NFS.

You may already own a NAS device that the SiCortex system can share with other systems. If not, there are numerous vendors to choose from.

For in-depth information about mounting and managing NFS file systems, see Chapter 9 - "NFS File Systems".

## Direct-Attached File Systems

In Direct-attached storage (DAS), the storage subsystem acts as a block server. Typically, you would connect a high-end DAS system to a larger SiCortex system using FibreChannel (FC). SiCortex systems connect to

Fibre Channel via a PCIExpress port. The system currently supports the QLogic QEM2462 PCI ExpressModule.

You would connect a lower-end disk device to a smaller SiCortex system using SATA, SAS, or SCSI as the transport and protocol.

For detailed directions on using direct-attached storage with SiCortex systems, see Chapter 10 - "Direct-Attached File Systems".

## Lustre File Systems

Lustre is a distributed filesystem, often called a parallel filesystem. The main architectural difference between Lustre and more traditional file systems is in how the data is organized and accessed. Ordinary disk-based file systems or network file systems like NFS and CIFS keep their metadata and file data in the same backend storage area, and access them via a single access path (controller, network connection, etc). In contrast, Lustre separates the metadata from the file data, which allows for distributing the file data to a number of separate storage areas, which then permits parallel access to those storage areas.

This difference is key to a cluster-type computer system. With a non-parallel filesystem, the limited number of connections between hosts and storage becomes a bottleneck. For these types of systems, the emphasis has been on increasing the speed of a single connection by improving the controllers, and obtaining faster Ethernet speeds and higher performance memory systems.

On cluster systems, especially SiCortex systems, the requirements are different. A SiCortex system doesn't need to emphasize super high performance individual peripherals because it is not limited to a small number of connection points. On a SiCortex system, you can have a large number of ordinary storage devices, and achieve extraordinary performance by configuring them using Lustre, and connecting to a large number of them in parallel. Lustre is designed to take advantage of the strengths of a SiCortex system, namely, many parallel paths between nodes.

According to Cluster File Systems (CFS), the client software on SiCortex systems should be compatible with any Lustre-based server that runs Version 1.4.6 or later. For additional information, see the documentation pages on the CFS website.

For detailed information about configuring, mounting, and managing a Lustre file system, see Chapter 11 - "Lustre File Systems".

# FabriCache File Systems

SiCortex provides a feature called FabriCache™ that allows you to create a Lustre file system entirely on a subset of the nodes in your system, using the nodes' RAM for storage. You select a partition, and all the components of the Lustre file system are created on nodes. All data is stored on the memory of nodes, and other nodes are used as clients.

A FabriCache file system offers 2 GB of storage per node, paired with the speed of the fabric interconnect, to create a parallel file system that's even faster than a Lustre file system on external storage.

For complete information about configuring, mounting, and managing a FabriCache file system, see Chapter 12 - "FabriCache File Systems".



# Chapter 9 NFS File Systems

SiCortex systems support the use of network-attached storage (NAS) provided you use the industry-standard Network File System (NFS) protocol to communicate with clients.


Mounting file systems on a SiCortex system uses most of the commands and mechanisms you would use to mount disks on any Linux system. However, there are some features that are unique to the SiCortex system. This chapter explains the unique features in context with the commands you're used to. In this chapter:

- NFS File System on External Server

## NFS File System on External Server

You can configure your SiCortex system to connect to an NFS file system with an external server. Follow these steps:

1. Ensure network routing is complete before you begin configuring the NFS file system. See *Step 7—Verify That Basic SSP Networking Works* on page 63, and *Step 13—Verify Networking Works on the Nodes* on page 90.
2. For each node that will mount the NFS file system, configure a route to the external NFS server.

 You must use TCP as the transport mechanism. Do not use UDP.

3. **On the (external NFS) server**, export the NFS file system, with permissions.
  - **Edit `/etc/exports` file.** On the host machine where the disk or directory you wish to mount actually resides, edit the `/etc/exports` file, to export the disk as the entity you wish to mount it as, on the nodes (ie. `/myfilesystem`). The host machine may be an external file server, or it may be the SSP. It can be any NFS file server.
  - For mounting `/myfilesystem`, the `/etc/exports` entries would look like this:

```
<server>:/etc/exports
...
/myfilesys *.scsystem(rw,tcp,sync,insecure,no_root_squash,no_subtree_check)
...
```

Note that the additional options in `/etc/exports` (`rw`, `tcp`, `sync`, etc.) and `/etc/fstab` (`rw`, `tcp`, `rsize`, etc) are not generic. They are dependent on network configuration and policy, and other variable factors.

#### 4. On the client (the nodes):

- **Edit `/etc/fstab`** in the the nodes' root file system, to create the translation that links the host machine's path and name for the disk or directory, with the name you want to mount it as, on the nodes. Use `exportfs -a`.
- **NOTE**—The `fstab` approach requires read/write access to the root file system.
  - On machines that use NFS, the root file system is already read/write.
  - On machines that boot the root file system using NBD, the root file system is read-only. To edit it, you must use `chrootfs`.

To edit `/etc/fstab` using `chrootfs`, see *Using chrootfs to Edit the Root File System* on page 270.

```
<node>:/etc/fstab
...
server:/myfilesys /myfilesys nfs rw,tcp,rsize=1024,wsiz=1024,noatime,hard,intr 0 0
...
```

- 5. Add a mount script to `../local.d`:** To mount the NFS file system on some or all the nodes, all the time, add a script similar to the following example, into `/opt/sicortex/config/local.d/`. Scripts that `sboot` finds in `../local.d` are executed both on the SSP and on every node at boot time.

```
#!/bin/bash

if [ -e /proc/scinfo ]; then                # Only if we are a cluster node, execute the following:
    echo "NFS mounting /myfilesys"
    source /var/state/boot_args            # Get the module id variable
    sleep ${SCV_module_id}                 # Stagger mount commands to avoid contention on ext. server
    mount -o tcp ssp:/myfilesys /myfilesys # Mount from SSP (/etc/fstab previously defined)
fi
```



The sample script file shown below mounts `ssp:/myfilesys` on all the nodes.

- The conditionalization is required; it makes the script a no-op on the SSP.
- The sleep command is advised to prevent contention against the external storage server. It causes the nodes to wait a number of seconds that is a function of their module number, before attempting to execute the mount command against the external storage device.

In the example `local.d` script that is shown below, note these lines:

```
source /var/state/boot_args

sleep ${SCv_module_id}
```

In these lines, the `SCv_module_id` variable comes from `/var/state/boot_args`. That variable is the module number that this node is on (0-35 in an SC5832). The `sleep` command allows nodes on module 0 to immediately issue the `mount` command, while nodes on module 35 will delay 35 seconds before issuing the `mount` command.

For external storage, having 972 `mount` requests within 35 seconds may still be too fast for some storage systems to handle, so you might want to do the following:

```
sleep $(( 3 * ${SCv_module_id} ))
```

where 3 is a scaling factor on the delay time. In this case, 972 mounts will be spread across 105 seconds.

```
#!/bin/bash

if [ -e /proc/scinfo ]; then      # Only if we are a cluster node, execute the following:
    echo "NFS mounting /myfilesys"
    source /var/state/boot_args  # Get the module id variable
    sleep ${SCv_module_id}      # Stagger mount commands to avoid contention on ext. server
    mount /myfilesys            # Mount from SSP (/etc/fstab previously defined)
fi
```

**6. Boot the nodes** to complete the mounting of the NFS file system.

7. OR - to mount the NFS file system only on some nodes, and only when you want it mounted (not all the time), use the `srun` command. For example:

```
srun -p my_partition -N 30 mount -a
```

## Mounting an NFS File System from the SSP

This section uses the example of mounting `/lclhome` as a shared file system, which comes preconfigured for you in the `/etc/exports` file on the SSP.

Follow the same steps given in the general case above, with these differences:

1. Refer to `/lclhome` instead of `/myfilesystem`, everywhere in the above section.
2. Edit `/etc/exports` on the SSP, not on an external NFS server.
3. Edit `/etc/fstab` in the node root file system, BUT set `<server>` to “ssp” to point it to the SSP, not to an external NFS server.
4. Boot the nodes.

## Editing the Root File System

NBD provides a read-only root file system on the nodes. To edit the root file system, use the `chrootfs` command.


For more information, see *Using chrootfs to Edit the Root File System* on page 270.

# Chapter 10 Direct-Attached File Systems

You can configure a direct-attached file system to connect to a single IO node on your SiCortex system. In this chapter:

- Configuring a Direct-attached File System

## Configuring a Direct-attached File System

 The configuration in this chapter uses a single SiCortex node as the file system server.

1. Determine which IO node you will be using to connect your storage array to the SiCortex system. Choose Node 0, 1, or 3 on one or more modules. These nodes are connected to PCIExpress ports.
2. Install Fibre Channel PCI ExpressModules in the PCIExpress ports for the chosen IO node. The system currently supports the QLogic QEM2462 PCI ExpressModule.
3. Connect your storage array to the PCI ExpressModule you just installed, to provide the physical link between the storage array and the IO node that will be configured as the file system server.
4. Configure your storage array so that it exports n LUNs of the desired type and size. The two types of storage arrays that SiCortex has tested and qualified are:
  - Promise
  - Data Direct Networks (DDN)
5. You may want to partition your external file system into smaller virtual drives, by running `fdisk`, `cfdisk`, or another disk partitioning utility.
6. If you have changed the storage array configuration, reboot the nodes. This ensures that the QLA driver is reloaded, and reinitializes its image of the storage array and its configuration.
7. You can now create file systems on whatever disk partitions you created. The three kinds of file systems SiCortex has qualified for use in this case are:

- ext2
- ext3
- reiser

For example, you can use `mkfs` commands, such as:

```
mkfs -t<file-sys-type> <options> <block-device>
```

-or-

```
mke2fs -t<file-sys-type> <options> <block-device>
```

Where `<block-device>` is something like `/dev/sda`.

8. Create a mount point on the one attached node.
9. Mount the direct-attached file system on that mount point on that node.

To ensure that the file system remains mounted on that node all the time, put the file system device and mount point into `fstab`. This creates a local file system accessible only from one node.

10. To access the file system from other nodes, follow the appropriate steps shown in *NFS File Systems* on page 173.

However, note that for a direct-attached file system, you must reference the I/O node as the server for the file system you just created (instead of referencing an external server.)

## Using NFS Server on a SiCortex System

There are a number of scenarios in which it makes sense to use NFS server software on a SiCortex cluster.

- A cluster node has direct attached storage (DAS) that you wish to export to other cluster nodes using NFS
- A cluster node has direct attached storage that you wish to export outside the cluster, using NFS

A file system such as ext2, ext3, or reiser which is kept on direct attached storage will typically be available only on the node to which it is connected. One way to make a file system available to other nodes is to use

Lustre, described in Chapter 11 - "Lustre File Systems", but if the filesystem you have is not a lustre filesystem, or if you have reasons for not wanting to use lustre, then NFS can be an alternative.

To use NFS, you must export the filesystem from the node where it is directly attached. This is also called setting up an nfs server. You must also import the file system on other nodes, where it is needed. This is called setting up an nfs client.

The file `/opt/sicortex/script_examples/nfs-export.sh` is an example of how to do this.

This sample script can be modified and copied to `/opt/sicortex/config/local.d` to do the work in your cluster.

The outline of the procedure is as follows:

On the server node:

1. Mount the file system you wish to export in a convenient place.
2. Create an entry in `/etc/exports` that describes the file system you are exporting, and the nodes which are allowed to import
3. Tell the event daemon (`ev1d`) on the ssp that the file system export is ready.

On the client nodes,

1. Wait for the event daemon to report that the export is ready
2. Wait a short time period calculated from the node module ID in order to make sure that all the clients don't request the mount "at once"
3. Create an appropriate mount point
4. Mount the NFS file system with appropriate options

The sample script happens to use a direct attached storage system that is really a tmpfs kept in the RAM of the server node.

This is kind of like an NFS version of Fabricache, but without the performance. It is really only useful as an example.

In most real situations, you would mount a file system from a real hardware device and then export it.

The sample script must be modified in a few places to make sense for a new environment

- The node name of the server node should be changed to be correct, both in the shell case statement and in the nfs mount command
- The server node branch of the case statement should be changed to mount the actual DAS file system, rather than the tmpfs
- The entry created in /etc/exports should have the correct IP address reflecting the internal netblock in use on your system.

Another situation where NFS makes sense is if your SiCortex cluster has direct attached storage that you want to make available to hosts outside the cluster. This is more complicated, because a SiCortex node that has direct attached storage will generally NOT have direct external network connectivity as well.

The general procedure for this case is the same as export to internal nodes, but you must also arrange for network connectivity to work.

- External NFS clients must have routing table entries for the NFS server that create IP connectivity
- The NFS server node must not be connected to the clients through a NAT gateway, because NAT will not work for connections originated by the external NFS clients.
- The NFS server node must have routing table entries to route IP traffic from the server to the clients through the same gateway as the incoming traffic.

For example, you might assign an IO node for the purpose of routing NFS traffic, and set up routes on the NFS server, the gateway, and the clients that force the NFS IP traffic to follow the path that you want.

Here's the script: `/opt/sicortex/script_examples/nfs-export.sh`

```
#!/bin/bash

# Example script to set up NFS export from direct attached storage to other nodes
# L. Stewart 2008-08-01

# Standard boilerplate: grab boot args, preclude running on SSP.
if [ ! -L /var/state/boot_args ]; then
    # Don't run this on the SSP.
    exit 0
```

```

fi
. /var/state/boot_args

# Allow these to be overridden for debugging.
if [ -z "$me" ]; then
    me=$(hostname -s)
fi
if [ -z "$evlport" ]; then
    evlport="ssp 1234"
fi

# This is the prefix that all of our evld entries will use.
prefix="${SCv_name}_nfsdas1"
# sbboot will erase items in evld that begin with ${SCv_name} on each
# boot, which is why this works

# This is an example where node sf1-m0n0 exports and the others import
# naturally, your partition and node names may vary

case $me in
sf1-m0n0)
    # create a tmpfs blank file, create and loop mount it
    # if you have real DAS storage, use that, This is a hack to
create a ram fs
    # that can be exported using NFS
    # nb the loop device name must not conflict with one already in use
    mkdir -p /tmp/space
    mount -t tmpfs tmpfs /tmp/space
    dd if=/dev/zero of=/tmp/space/space1 bs=1M count=100
    losetup /dev/loop0 /tmp/space/space1
    # construct a filesystem
    mke2fs /dev/loop0

    # mount that
    mkdir -p /tmp/das1
    mount /dev/loop0 /tmp/das1

    # if your internal netblock is not 192.168, change the next line
    # man exports to learn more about the options
    echo /tmp/das1
"192.168.0.0/16(rw, sync, insecure, no_root_squash, no_subtree_check)"
> >/var/state/exports
    rm -f /etc/exports
    ln -s /var/state/exports /etc/exports
    /etc/init.d/nfs start
    echo -en "add~${prefix}_server=~${prefix}_server=ok\ndone\n" |
nc $evlport
    ;;
*)
    # We're a client.
    # Wait for the server.
    echo -en "fetch~1~${prefix}_server=ok\ndone\n" | nc $evlport
    # Note, the sleep has to be AFTER the evl step, because the
    # whole point is to stagger the mounts. The other way, if the
    # evl setup is slower than the sleep, all the mounts will still be
    # synchronized

    echo "Nfs client sleeping $_SCv_my_modnum seconds"
    sleep $_SCv_my_modnum
    mkdir -p /tmp/das1
    mount -t nfs -o tcp sf1-m0n0.scsystem:/tmp/das1 /tmp/das1
    ;;
esac

exit 0

```





# Chapter 11 Lustre File Systems

This chapter provides some information and tips on configuring Lustre file systems for use with a SiCortex system.

If you have large datasets, and are interested in high throughput streaming IO, Lustre will perform far better than NFS, because it can use parallel IO paths on a SiCortex system. In addition, a cluster can cause an NFS server to fail under the load of too many simultaneous requests. Lustre distributes the load, which allows better scaling.

Another reason Lustre runs fast on the SiCortex system is that it runs directly on the fabric, using the Lustre network driver.

There are several ways you can configure an external Lustre file system on a SiCortex system. In this chapter:

- Lustre Overview
- Lustre File System with External Server
- Lustre File System with SiCortex Nodes as Servers

## Lustre Overview

The following table summarizes the components in a Lustre file system:

Lustre Component	Explanation
MGS - Management Server	Each MGS serves N file systems. Each file system has 1 MDS and n OSSes. Each OSS has n OSTs.
MDS - Metadata Server	One per file system. May reside on a node or an external host, depending on configuration.
MDT - Metadata Target	One per MGS. Disk/LUN connected to external hosts, disk/LUN connected to nodes, or ramdisk pieces.
OSS - Object Storage Server	Many per file system. May reside on a node or an external host, depending on configuration.

Lustre Component	Explanation
OST - Object Storage Target	Many per OSS. Disk/LUN connected to external hosts, disk/LUN connected to nodes, or ramdisk pieces.
Clients	SiCortex nodes

## Lustre File System with External Server

Configuring a Lustre file system that has an external server is similar to configuring an NFS file system.

Set up network routing so that all client nodes on which you will mount this Lustre file system have a route to each of the Lustre external servers.

### Methods for Mounting the Lustre File System

The following table describes the various methods you can use to mount the Lustre file system.

Mount Goal	Mount Location and Method
Always mount this Lustre file system everywhere	Put the mount commands into a conditionalized script that runs them on <i>all</i> the nodes but not on the SSP.  Put that script into the <code>/opt/sicortex/config/local.d</code> directory. Scripts that <code>scboot</code> finds in <code>.../local.d</code> are executed on every node at boot time.
Mount the Lustre file system in selective locations, but do so automatically every time the nodes boot.	Put the mount commands into a conditionalized script that runs them only on the selected nodes.  Put that script into the <code>/opt/sicortex/config/local.d</code> directory, so that it will be executed every time the nodes are booted.
Mount the Lustre file system only on selective locations, and only at times of your choosing.	Use the <code>srun</code> command to execute the mount commands in the desired locations at the desired times.

# Lustre File System with SiCortex Nodes as Servers

Alternatively, you can designate a number of SiCortex nodes as the servers for the Lustre file system. This case is similar to the basic direct-attached storage case described in *Direct-Attached File Systems* on page 177.

1. Configure the storage arrays according to the instructions from the manufacturer. Consult the documentation available from [www.clusterfs.com](http://www.clusterfs.com) for ways to configure them sensibly.
2. Connect to multiple I/O nodes, via FibreChannel connected to QLogic PCIExpress modules that are plugged into the PCIExpress ports on the modules.
3. On the IO nodes defined as the file system servers, load the QLA drivers.
4. Configure the Lustre file system. For an example of how to do this, see *Lustre File System Configuration Example* on page 185.
5. Start the Lustre file system. You do this, on the server nodes, by mounting the Lustre file system components:

For this example, there is one MGS.

- Set up the mount points: On each of the MGS/MDSes and OSSes, create a mount point for each MGT/MDT/OST served by that server, and mount the target on that point.
    - First, on the one MGS, mount the one MGT.
    - Next, mount the corresponding OSTs.
6. From each of the client nodes in the Lustre file system, mount the Lustre file system.

## Lustre File System Configuration Example

This section explains how to configure a sample Lustre file system. The sample file system is shown Figure 1:

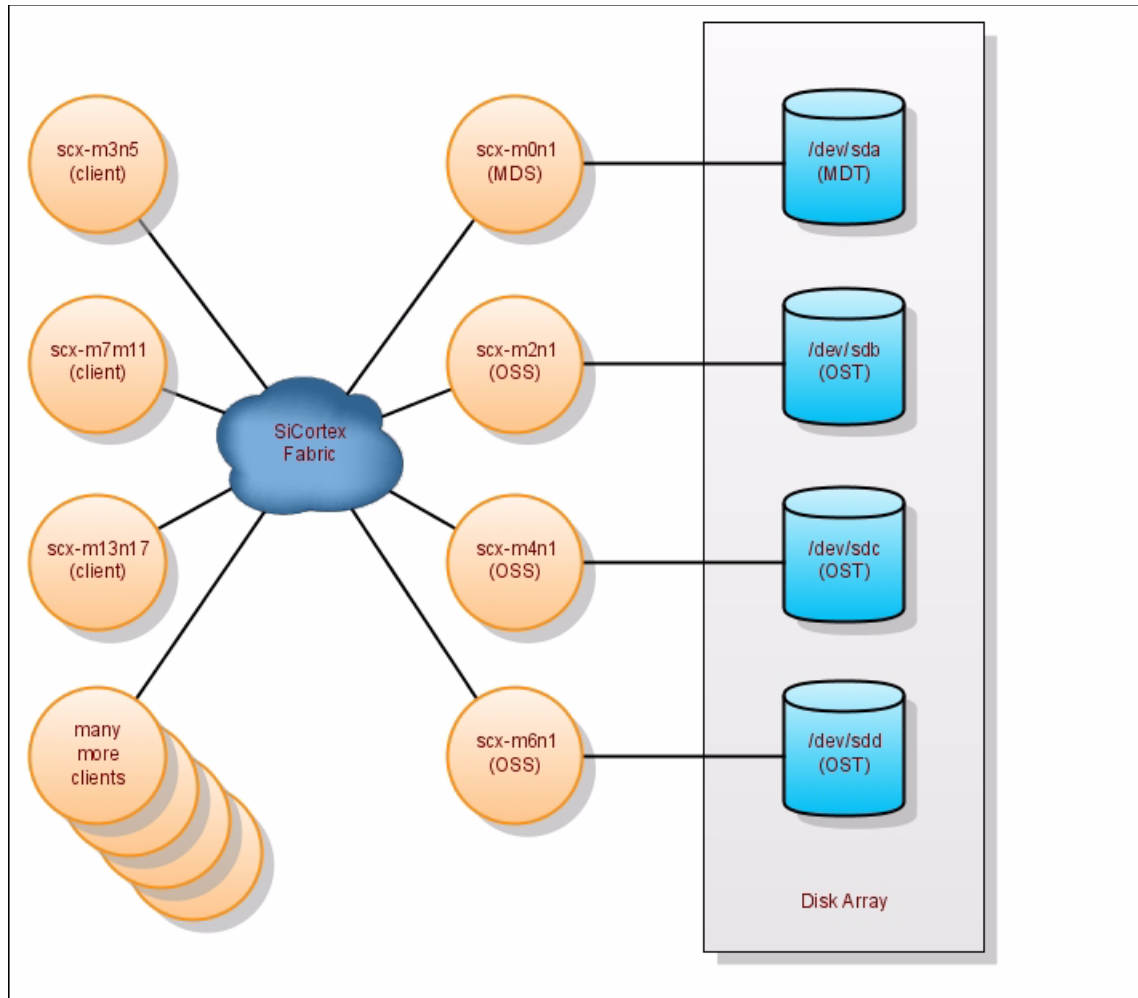



Figure 1. Sample Lustre file system

The illustration above shows four I/O nodes, one MDS and three OSSes, connected to four LUNs on some (unspecified) kind of disk array. These will all be combined into a single Lustre filesystem, which will then be mounted by all of the clients.

## Creating the Lustre File System

To create a Lustre file system like the one shown above, follow these steps.

1. Ask your storage administrator to create and export four LUNs (essentially the same as the previous example).
2. Check the cables, then reboot the nodes.

 You must not load Lustre yourself by hand, either with your own explicit `modprobe` command or implicitly by mounting a Lustre file-system.

3. As your very first Lustre-related action:

- Obtain the `lustre_funcs.sh` script from your SiCortex Application Engineer (AE). This script was not included with Version 2.1 but will be included with Version 2.2.
- Source this file using the bash `source` command:

```
# source /usr/sbin/lustre_funcs.sh
```

- Depending on whether you are using `socklnd` or `sc1nd`, use one of the two commands below to invoke the script to load Lustre:

```
# load_lustre sock
```

or

```
# load_lustre sc
```

After `load_lustre xxx`, the Lustre network modules are now loaded, with parameters that tell Lustre to operate in the specified mode.

Also, note that the Lustre logging code is located in the `lnet` module and can be a significant cause of instability if configured in the default way. Therefore, the `lustre_funcs.sh` script provided by SiCortex sets up Lustre logging so that it is directed to files in a dedicated `ramfs` file system mounted at `/tmp/lustre_logs`.

At this point, you have configured Lustre networking and logging. In the next steps, you will configure the Lustre server and client modules, and the mounts using those modules.

`Socklnd` is a standard LND (Lustre network driver) provided by CFS/Sun, which operates over TCP. Therefore it has very broad compatibility and all of the features (e.g. routing) that you get with using IP. It also has all of the overhead. Given the computation/communication balance on a SiCortex system, that overhead is costly. Therefore, SiCortex provides a custom alternative: `sc1nd`. `sc1nd` uses an interface that is tailored to the way Lustre works, instead of mapping that functionality onto sockets. This approach avoids all of the overhead that mapping to sockets would cause. As

a result, `sc1nd` can potentially (with the right tuning) be much faster than `sock1nd`. However, it's important to note that `sc1nd` is more specialized than `sock1nd`. `sc1nd` can only be used between SiCortex nodes on the same fabric, and `sock1nd` is still necessary to communicate with external Lustre servers. Much of what the `load_lustre` command does is actually concerned with the complexity of enabling both `sc1nd` and `sock1nd` simultaneously.

If you are not already familiar with LNDs, see the Sun/CFS Lustre documentation at <http://www.clusterfs.com/> <http://www.sun.com/software/products/lustre/index.xml> for more information.

Other Lustre facts:

- Lustre NIDs (see the CFS/Sun documentation) are expressed as `node-id@1nd`, where "`1nd`" specifies which LND to use and "`node-id`" is a unique node identifier in an LND-specific format.
- This means that you can omit the "`@1nd`" portion of the NID specifier. If you do so, Lustre assumes you are using `sock1nd`, and you can specify `node-id` in that case as either a name or an IP address. The examples in this chapter omit the "`@1nd`" and use names to specify the nodes.
- To use `sc1nd` instead of `sock1nd`, you would need to change all of the NIDs in the configuration commands to use `nnn@sc`, where `nnn` is the sequential numeric node number - e.g. 5 for `scx-m0n5` or 65 ( $27*2+11$ ) for `scx-m2n11`, for example. For details about sequential node numbering in lower-level contexts, see *Calculating sequential node number* on page 265.
- The numbering format for Lustre NIDs will change in the next release. The NID will be specified using the more familiar `mxy` format. This affects, for example, the `mgsnode` argument passed to all of the `mkfs.lustre` commands in the example in this chapter, as well as the client mount commands.

4. Format the MDT. For example, on `scx-m0n1`, you would type:

```
root@scx-m0n1 ~ $ mkfs.lustre --mgs --mdt --fsname=lustre1
--mgsnode=scx-m0n1 --reformat --comment="lustre1 MDT" /dev/sda
```

- `mkfs.lustre` is the command to format a Lustre filesystem, whether it's an MDT or OST.

- `--mgs --mdt` tells `mkfs.lustre` that we're formatting both an MGT and an MDT.
- `--fsname=lustre1` specifies that "lustre1" is the name for this file system. This name is the "key" that everyone - MDS, OSSes, and clients - uses to know which pieces are associated with which file system.



The name of the Lustre file system absolutely positively must be seven characters or less. This limitation is inherited from CFS. Unpredictable failures will occur if you attempt to give your Lustre file system a name that exceeds seven characters.

`--mgsnode=scx-m0n1` specifies that `scx-m0n1` will be the MGS for this filesystem, and it stores that information within the MDT or OST header. You could in theory format the MDT on one node and then actually mount it on another, though it's not clear why you'd bother.

`--reformat` specifies that `mkfs.lustre` is to format the volume even if it thinks it's already formatted. This option may seem pro forma, but sometimes `mkfs.lustre` refuses to continue without it.

`--comment` arguments are not required by Lustre. However, our sample scripts use the comments to find and identify the parts of the Lustre file system. Therefore they are included in the examples in this chapter.

`/dev/sda` is the device we're going to use for the MDT. This device might be different depending on what LUNs the you actually created before.

5. Format the OSTs. This can actually be done in parallel, not just across the OSTs but while the MDT is being formatted as well, since they're all separate volumes.

The commands are as follows, one per OSS. This example formats three OSTs, on devices `/dev/sdb`, `sdc`, and `sdd`:

```
root@scx-m2n1 ~ $ mkfs.lustre --ost --fsname=lustre1
--mgsnode=scx-m0n1 --reformat --comment="lustre1 OST0" /dev/sdb
root@scx-m4n1 ~ $ mkfs.lustre --ost --fsname=lustre1
--mgsnode=scx-m0n1 --reformat --comment="lustre1 OST1" /dev/sdc
root@scx-m6n1 ~ $ mkfs.lustre --ost --fsname=lustre1
--mgsnode=scx-m0n1 --reformat --comment="lustre1 OST2" /dev/sdd
```

The only things that change, compared to the previous MDT-format command, are the device names, and the fact that we use `--ost` instead of `--mgs --mdt` to indicate that this time we're formatting an OST. As in the MDT case, the identity of the MDS is stored in the OST header.

## Mounting the Lustre File System

The next step is to mount the Lustre file system. There are three stages, and they must occur in sequence:

1. Assume that step 3 from above has been completed.
2. Create a mountpoint and mount the MDT on the MDS. In this example, the MDS is also being used as the MGS. The commands are:

```
root@scx-m0n1 ~ $ mkdir -p /tmp/lustre_mdt
root@scx-m0n1 ~ $ mount.lustre /dev/sda /tmp/lustre_mdt
```

3. Mount the OSTs on the OSSes. Each OSS finds the MDS based on the information stored within the OST itself, and connects to it. This time, the only thing that changes is the device name. The commands (two per OSS) are:

```
root@scx-m2n1 ~ $ mkdir -p /tmp/lustre_ost
root@scx-m2n1 ~ $ mount.lustre /dev/sdb /tmp/lustre_ost
```

```
root@scx-m4n1 ~ $ mkdir -p /tmp/lustre_ost
root@scx-m4n1 ~ $ mount.lustre /dev/sdc /tmp/lustre_ost
```

```
root@scx-m6n1 ~ $ mkdir -p /tmp/lustre_ost
root@scx-m6n1 ~ $ mount.lustre /dev/sdd /tmp/lustre_ost
```

4. Mount on the clients. Each client finds the MGS based on information *on the command line* (since it doesn't have an MDT/OST to look at) and connects to it. Note that in this example, the MGS happens to be the same as the MDS. The MDS then uses the provided filesystem name to tell the client about all of the relevant OSSes, and the client connects to those as well. Since this is all automatic, there are only two commands which must be issued *on each client* (i.e. everyone except the MDS and the three OSSes).

```
root@scx-mXnY ~ $ mkdir -p /tmp/mylustre
root@scx-mXnY ~ $ mount.lustre scx-m0n1:/lustre1 /tmp/mylustre
```



## Mounting a Lustre File System with a Script

It can be time-consuming to perform all of the commands shown above by hand. The alternative is to run a script which does all the coordination as well as the actual mount commands.

### **lustre\_mount.sh script example**

An example of such a script is shown below (`lustre_mount.sh`). This script would be placed in `/opt/sicortex/config/local.d`. Scripts that `sboot` finds in `.../local.d` are executed on every node at boot time. The `lustre_mount.sh` script example shown below creates a configuration very similar *but not identical* to the one used for the above example.

The script shown below is useful as an example of the preferred way to create and mount all components of the file system automatically in the right sequence every time the system boots.

#### **lustre\_mount.sh (part 1)**

```
#!/bin/bash

# Standard boilerplate: grab boot args, preclude running on SSP.
if [ ! -L /var/state/boot_args ]; then
    # Don't run this on the SSP.
    exit 0
fi
. /var/state/boot_args

# Allow these to be overridden for debugging.
if [ -z "$me" ]; then
    me=$(hostname -s)
fi
if [ -z "$evlport" ]; then
    evlport="ssp 1234"
fi

# This is the prefix that all of our evld entries will use.
prefix="${SCv_name}_b4fs1"

# Common code since three nodes do this but a bit differently for each.
# Args: $1 = device name, $2 = mount point
function setup_oss
{
    # Wait for the MDS.
    echo -en "fetch~1~${prefix}_mds=ok\ndone\n" | nc $evlport
    mkdir -p $2
    mount -t lustre -o abort_recov $1 $2
    # Let clients know we're done.
    echo -en "update~${prefix}_oss=~${prefix}_oss=X\ndone\n" | nc $evlport
}

# Text of file continues on next page...
```

**lustre\_mount.sh (continued, part 2)**

```

# Everyone does these steps.
. /usr/sbin/lustre_funcs.sh
load_lustre sc

function find_dev
{
    local d

    for d in /dev/sd[a-d][1-2]; do
        tuneufs.lustre --dryrun $d | egrep "Comment: $1" > /dev/null
        if [ $? = 0 ]; then
            echo $d
            return 0
        fi
    done

    return 1
}

case $me in
scx-m0n1)
    # Find devices. Do this first for all devices because tuneufs.lustre
    # complains if run against a device that's mounted.
    mdt_dev=$(find_dev "b4fs1 MDT")
    if [ $? != 0 ]; then
        echo "Could not find MDT"
        exit 1
    fi
    echo "MDT is $mdt_dev"
    # Mount MDT.
    mkdir -p /tmp/lustre_mdt
    mount -t lustre -o abort_recov $mdt_dev /tmp/lustre_mdt/
    # Set up ev1 entries for ourselves and others.
    echo -en "add-{{prefix}}_oss=\ndone\n" | nc $ev1port
    echo -en "add-{{prefix}}_mds=ok\ndone\n" | nc $ev1port
    ;;
scx-m2n1)
    ost_1_dev=$(find_dev "b4fs1 OST 1")
    if [ $? != 0 ]; then
        echo "Could not find OST 1"
        exit 1
    fi
    echo "OST 1 is $ost_1_dev"
    ost_4_dev=$(find_dev "b4fs1 OST 4")
    if [ $? != 0 ]; then
        echo "Could not find OST 4"
        exit 1
    fi
    echo "OST 4 is $ost_4_dev"
    # Mount OSTs.
    setup_oss $ost_1_dev /tmp/lustre_ost_1
    setup_oss $ost_4_dev /tmp/lustre_ost_4
    ;;
# Text of file continues on next page...

```

**lustre\_mount.sh (continued, part 3)**

```

scx-m4n1)
# Find devices. See scx-m2n1 comment about order of operations.
ost_2_dev=$(find_dev "b4fs1 OST 2")
if [ $? != 0 ]; then
    echo "Could not find OST 2"
    exit 1
fi
echo "OST 2 is $ost_2_dev"
ost_5_dev=$(find_dev "b4fs1 OST 5")
if [ $? != 0 ]; then
    echo "Could not find OST 5"
    exit 1
fi
echo "OST 5 is $ost_5_dev"
# Mount OSTs.
setup_oss $ost_2_dev /tmp/lustre_ost_2
setup_oss $ost_5_dev /tmp/lustre_ost_5
;;
scx-m6n1)
ost_3_dev=$(find_dev "b4fs1 OST 3")
if [ $? != 0 ]; then
    echo "Could not find OST 3"
    exit 1
fi
echo "OST 3 is $ost_3_dev"
ost_6_dev=$(find_dev "b4fs1 OST 6")
if [ $? != 0 ]; then
    echo "Could not find OST 6"
    exit 1
fi
echo "OST 6 is $ost_6_dev"
# Mount OSTs.
setup_oss $ost_3_dev /tmp/lustre_ost_3
setup_oss $ost_6_dev /tmp/lustre_ost_6
;;
*)
# We're a client. The number inside the braces in this fetch command
# is the number of OSSes we expect to precede us.
echo -en "fetch~1~${prefix}_oss=X{3}\ndone\n" | nc $ev1port > /dev/null
echo "Lustre client sleeping $_SCV_my_modnum seconds"
sleep $_SCV_my_modnum
mkdir -p /tmp/lustre
mount -t lustre 1@sc:/b4fs1 /tmp/lustre
;;
esac
exit 0

```

### Using `ev1d` for coordination

The `lustre_mount.sh` script uses `ev1d` for coordination. All lines of code that refer to `$ev1port` are referring to `ev1d`.

`ev1d` is a system service. `ev1d` allows you to add or modify lines of text to a global pool, and to fetch lines that match a pattern either immediately or when they become available.

In the `lustre_mount.sh` script, the OSSes issue blocking fetch commands for a line indicating that the MDS is done, and the MDS adds such a line when it completes.

Similarly, the clients wait for a line indicating that some number of OSSes have finished, and the OSSes increment that count as they complete.

Using `ev1d` in this way, every routine waits for its dependencies to be satisfied before it issues its own mount commands, without over-serializing, as would happen with traditional barriers.

## Unmounting a Lustre File System with a Script

To unmount a disk-based Lustre filesystem before a reboot, use the `lustre_umount.sh` script. This script is available on the nodes:

1. Log into an appropriate node.
2. The following sample commands show how to invoke the `lustre_umount.sh` script from a node. The script is provided on the next page:


```
srunk -p scx -N 972 /sbin/lustre_umount.sh b4fs1 client
srunk -p scx -N 972 /sbin/lustre_umount.sh b4fs1 mds
srunk -p scx -N 972 /sbin/lustre_umount.sh b4fs1 oss
```

In this example, `scx` is the name of the partition for the entire system, `972` is the number of nodes in the partition, and `b4fs1` is the name of the Lustre filesystem to be unmounted (as in the site-customized mount script).

3. To unmount your Lustre file system, execute a series of commands like those above. Change the values to match your Lustre filesystem configuration.



**The order of the three commands is important!** Clients must be unmounted first and OSSes last.

 If the filesystem is busy on a client, that client will be unable to unmount. To avoid this, ensure that the target filesystem has no open files or user sessions' current directories before unmounting.

For the text of the `lustre_umount.sh` script, see *lustre\_umount.sh* on page 195.

or `/opt/sicortex/rootfs/default/sbin/lustre_umount.sh`.

## lustre\_umount.sh

### lustre\_umount.sh

```
#!/bin/bash

me=$(hostname -s)

function umount_client
{
    for i in /proc/fs/lustre/mdc/${1}-MDT*; do
        if [ ! -d $i ]; then
            # This can happen if the specified filesystem does not
            # exist in any form on this node, so "for i in x*"
            # executes the loop body with "x*" as a value (broken
            # bash behavior IMO but we have to work around it).
            continue
        fi
        nid=$(cat $i/mds_conn_uuid)
        while read src on dst rest; do
            echo "Client $me unmounting $dst"
            umount $dst
        done <<(mount | egrep "^${nid}.* type lustre ")
    done
}

function umount_mds
{
    for i in /proc/fs/lustre/mds/${1}-MDT*; do
        if [ ! -d $i ]; then
            # See comment in umount_client.
            continue
        fi
        dev=$(cat $i/mntdev)
        while read src on dst rest; do
            echo "MDS $me unmounting $dst"
            umount $dst
        done <<(mount | egrep "^${dev} .* type lustre ")
    done
}

# Text of file continues on next page...
```

**lustre\_umount.sh (continued)**

```

function umount_oss
{
    for i in /proc/fs/lustre/obdfilter/${1}-OST*; do
        if [ ! -d $i ]; then
            # See comment in umount_client.
            continue
        fi
        dev=$(cat $i/mntdev)
        while read src on dst rest; do
            echo "OSS $me unmounting $dst"
            umount $dst
        done < <(mount | egrep "^${dev} .* type lustre ")
    done
}

case $2 in
client)
    umount_client $1
    ;;
oss)
    umount_oss $1
    ;;
mds)
    umount_mds $1
    ;;
*)
    echo "Usage: $0 filesystem client|oss|mds" > /dev/stderr
    exit 1
esac

exit 0

```

## Unmounting a FabriCache File System

To unmount a FabriCache filesystem, use `fc_destroy` instead of the above procedure.

See Chapter 12 - *FabriCache File Systems* on page 197.

## Chapter 12 FabriCache File Systems

This chapter describes how to configure a FabriCache™ file system on a subset of the SiCortex nodes.

The SiCortex system's FabriCache™ feature allows you to set up a Lustre file system on a subset of the nodes on the system, using the nodes' RAM instead of external disks, as the Lustre block devices. For a description of Lustre file systems, see Chapter 11 - "Lustre File Systems".

In this chapter:

- FabriCache—A Lustre File System on Nodes, Not Disks
- Setting up a FabriCache File System
- Using a FabriCache File System

### FabriCache—A Lustre File System on Nodes, Not Disks

You can use FabriCache to set up a Lustre file system on SiCortex nodes instead of on external storage devices.

#### May Save on Purchasing Additional External Storage

FabriCache allows you to set up a Lustre parallel file system directly on a group of SiCortex nodes, using the DIMMs instead of external disks as the storage medium. As a result, FabriCache provides many of the benefits of a global shared memory without the cost of purchasing additional external storage devices.

Because FabriCache uses RAM for storage, that memory is not persistent across a system power-down/power-up, or a reboot of the nodes. So FabriCache is not a substitute for external storage, but rather, can be used as a very large high-speed cache.

#### Faster, Too

Because the clients communicate with the MDSes and OSSes using the SiCortex interconnect fabric, the I/O waits are even shorter than for a Lustre file system on direct-attached storage or external storage.

A SiCortex system has a large amount of RAM. The Lustre chapter describes storage units (OSTs) as disks. But you can configure the system to use pieces of RAM on the nodes as ramdisks, and use Lustre to tie them together to make a high performance RAM-based filesystem. This is FabriCache.

Building a Lustre file system using the RAM on the nodes is possible because an OST can be built on any Linux block device. Most block devices are spinning media, but there are other kinds, such as ramdisks. A ramdisk "device", to Linux, appears just like any other kind of block device. The relevant differences are that they unlike external disks, ramdisks do not require external hardware, and they generally perform better than most external storage media, because reading and writing them are both RAM to RAM copy operations.

These facts make it possible to imagine a new way to use the SiCortex system. The system has a large group of nodes which have 4 or 8 GB of memory each.

For example, you could use all 102 computing nodes\* on an SC648 to create a FabriCache file system. You could allocate 51 nodes for computing, using one node as the head node and 50 nodes as client nodes, and use the other 51 nodes as server nodes, where the file system resides.

Assuming the system had nodes with 4GB of RAM, each node must reserve 2GB for its own functionality, so a 4GB node can donate 2GB of RAM to the FabriCache file system.

Therefore, you could take those 54 server nodes, make one an MDS/MDT, make the other 53 into OSS's, and use 2GB of each of their RAM as OSTs. That's  $53 \times 2\text{GB} = 106\text{GB}$  of storage. For many applications, 54 client nodes computing and 106 GB of storage which runs at approximately ramdisk speeds is an effective and efficient combination. This line of reasoning is the basis for Fabricache.

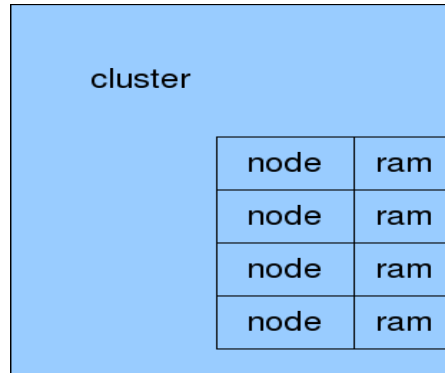
\* Excluding the 6 or so nodes used as the head node, IO nodes, and rootfs nodes.

## **Lustre on FabriCache**

For a quick review of the architecture of a Lustre file system, see Chapter 11 - "Lustre File Systems".



A Lustre file system on FabriCache is very similar to a Lustre file system on external storage. The main difference is that all the Lustre components reside on SiCortex nodes.




The following table summarizes the components of a Lustre file system as they are used in a FabriCache file system:


Lustre Component	In External Lustre File System, Resides on:	In FabriCache File System, Resides on:
MGS - Management Server		
MDS - Metadata Store	External server	SiCortex nodes
MDT - Metadata Target	External storage	SiCortex nodes
OSS - Object Storage Server - a machine or node that provides access to one or more OSTs	External server	SiCortex nodes
OST - Object Storage Unit - a slice of storage	External storage	Memory within SiCortex nodes, with a one-to-one mapping of OSS to OST
Clients		SiCortex nodes

## Requirements and Limitations

- **Nodes must be dedicated to FabriCache.** You cannot use nodes that are part of a FabriCache file system for any other purpose, while the file system is in place.
- **Always specify a partition that excludes all special purpose nodes,** ie. head node, IO gateway nodes, and rootfs server nodes.
- **Servers and clients must be on separate notes.**


- **Nodes must start in an idle state and remain idle while creating the FabriCache file system.** The nodes in the partition you allocate must be idle before the `fc_create` command starts, and must remain idle while the `fc_create` utility sets up the FabriCache file system.

 If you or other users attempt to use any of these nodes for other purposes, so that some nodes become unavailable while `fc_create` is running, creation of the FabriCache file system may fail and leave the system in an undetermined and unusable state. If this happens, the only recovery option is to reboot the nodes.

 **FabriCache should be treated as cache.** Because FabriCache uses the nodes' RAM as its memory, the information you store in FabriCache is **not persistent across a power outage or a reboot of the nodes.**

## Setting up a FabriCache File System

To set up a FabriCache file system, you invoke the `fc_create` utility on the SSP. This utility builds the file system based on the parameters you supply.

 You must configure your FabriCache file system at a time when the system is idle for all other purposes, to avoid conflicts with other processes or applications.

### `fc_create` Command

The `fc_create` command creates a FabriCache file system of the specified size in the specified partition.

#### Usage

```
fc_create [ options ] partition fs-name fs-size nservers
```


## Required Parameters

<i>partition</i>	Text. Specify an existing SLURM partition to be used for FabriCache servers and clients. This is the SLURM partition within which the FabriCache file system will be created. Only nodes that SLURM considers “idle” will be used; of these, one will be selected as an MDS, a number equal to <i>nserver</i> s (see below) will be selected as OSSes, and the remainder of the idle nodes will be allocated as clients.  <b>NOTE: Never specify the boot partition (sc1 or scx). Always specify a partition such as sc1-comp or scx-comp, that excludes the head node, IO nodes, and rootfs server nodes.</b>
<i>fs-name</i>	Text. The name for the FabriCache filesystem. Must conform to the rules for a Lustre file system name, ie. must be a maximum of seven characters long. The filesystem will appear as:  /tmp/fabclient/<fs_name> on clients.
<i>fs-size</i>	Integer. The size of the filesystem in megabytes (MB).
<i>nserver</i> s	Integer. The number of nodes to assign as OSSes for the new filesystem. This is the number of OSSes to stripe the filesystem across.

## Options


-h, --help	Show the help message for the command and exit.
-s, --sc1nd	Use <b>sc1nd</b> instead of <b>sock1nd</b> (fail if sock1nd is not available). This may improve performance.
-n, --noslurm	Do not create SLURM partition for clients.

## Usage Notes

-  **Never interrupt `fc_create` before it finishes.** Interrupting or terminating `fc_create` prematurely leaves the entire SiCortex system in an indeterminate and unusable state. Doing so is likely to necessitate rebooting the nodes to recover.

### **Basic Parameters: *partition, fs-name, fs-size***

You must specify the first three parameters, partition, file system name, and file system size, to give `fc_create` the basic information to create your FabriCache file system.


 **NOTE:** Never specify the boot partition (`sc1` or `scx`) that includes all nodes. Always specify a partition such as `sc1-comp` on an SC648 or `scx-comp` on an SC5832, that *excludes* the head node, IO nodes, and rootfs server nodes.

### **Number of Servers: *nserver***

The number of servers is an integer that specifies the number of nodes to assign as OSSes for the new filesystem. This is the number of OSSes across which to stripe the filesystem.

The number of servers is a simple function of the file system size. Each server node can contribute a maximum of (node RAM - 2GB) to the file system, because each server node must reserve 2GB for its own operations. This means that a server node with 4GB of RAM can contribute 2GB to the Lustre file system. A server node with 8GB of RAM can contribute 6GB to the file system.


Subject to the above limit, the number of servers is a performance tradeoff. If you know you will use `N` nodes for computation, you can achieve maximum performance by assigning all but `N` nodes as servers (including one as the MDS).

 Failing to reserve 2GB per server is extremely risky as it might lead to memory exhaustion and node failure which usually leaves all clients hung on FabriCache I/O.


The `nserver` parameter allows you to tune the size of each server, so that each server can be smaller. This might help performance for some applications, although in internal testing at SiCortex, a stripe count of more than eight did not appear to provide any additional benefit.

### **When `fc_create` Completes...**

When `fc_create` completes, it creates a SLURM partition named `<fs_name>_clients` so that users will know where to run jobs that use the newly created file system.

 Creating multiple FabriCache filesystems at once has not been well tested and may lead to unanticipated problems. For example,

`fc_create` does not mark the servers for the first file system as being unavailable, so they would probably be selected again for the second file system, a scenario which could fail in various catastrophic ways.

 If creating multiple FabriCache file systems on the same SiCortex system is a priority, create separate SLURM partitions to ensure that no overlap occurs in allocating nodes.

## Using Other `fc_create` Options

The `-s` option specifies the use of `sc1nd` instead of `sock1nd` as the internal connection method. Note that `sc1nd` may be faster, but has not been tested as well. `fc_create` (actually, the script it runs on the nodes) will ensure that Lustre has been configured for the right connection method, loading it itself if necessary or failing if it's already loaded the "wrong" way.

The `-n` option tells `fc_create` not to create a SLURM partition.

## `fc_destroy` Command

To dismantle the FabriCache file system you built with the `fc_create` command, use the `fc_destroy` command.

`fc_destroy` takes only the *partition* and *fs\_name* arguments as described for `fc_create`.

`fc_destroy` dismantles everything that `fc_create` set up, including the MDSes, OSSes, client role for nodes, and the SLURM partition if present.

## Usage

```
fc_destroy partition fs-name
```

## Required Parameters

<i>partition</i>	Text. Specify the same existing SLURM partition within which you created this FabriCache file system. <b>fc_destroy</b> dismantles all MDSEs and OSSEs that were part of the FabriCache file system. It also removes all remaining client nodes from their role as FabriCache Lustre file system clients.
<i>fs-name</i>	Text. The name for the FabriCache filesystem to be dismantled. May be any name that is legal as a file name. The filesystem will appear as:  /tmp/fabclient/<fs_name> on clients.

## Options

-h, --help	Show the help message for the command and exit.
------------	---

## FabriCache Example

The following example shows the creation of a FabriCache file system in partition `sc0`, on however many idle nodes are in that partition, of 40000MB (~40GB), with 20 OSTs served by 20 OSSEs.

This example uses the same 1:1 server/target mapping mentioned earlier in this chapter:

```
ssp020 sysadmin # time fc_create -s sc0 my_test 40000 20
srun -p sc0 -w sc0-m0n0 /usr/sbin/fabricache.sh my_test 2000 0@sc 1 mds
2000+0 records in
2000+0 records out
2097152000 bytes (2.1 GB) copied, 14.654 s, 143 MB/s
```

*[ some output omitted for brevity ... ]*

```
ssp020 sysadmin # sinfo -p my_test_clients
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
my_test_cl up infinite 86 idle
sc0-m0n[22-26],sc0-m1n[0-26],sc0-m2n[0-26],sc0-m3n[0-26]
```

```
ssp020 sysadmin # salloc -p my_test_clients -N 1 bash -c 'ssh $SLURM_NODELIST'
salloc: Granted job allocation 30
```

```
sc0-m0n22 ~ # df
Filesystem          1K-blocks      Used Available Use% Mounted on
rootfs              40322048    8394080  29879680  22% /
172.31.150.1:/opt/sicortex/rootfs/build
```

```

40322048 8394080 29879680 22% /
172.31.150.1:/tftproot/scboot_tmp/sc0-modules/lib/modules
40322048 8394080 29879680 22% /lib64/modules
tmpfs    262144    192    261952    1% /tmp
tmpfs    524288    4352    519936    1% /var
udev     10240    1216    9024    12% /dev
tmpfs    524288    4352    519936    1% /opt/sicortex/config
0@sc:/my_test 40316480 721592 37546888 2% /tmp/fabclient/my_test

```

## Using a FabriCache File System

There are two rules for using a FabriCache filesystem:

- When you use the `fc_create` command to create a FabriCache file system, if you name the filesystem `xyz`, the client nodes that use file system `xyz` are placed in a SLURM partition called `xyz_clients`.

You should use the `-w` option with `srun` or `salloc` to specify that a particular SLURM job or allocation should be limited to the client nodes in the `xyz_clients` partition. This ensures:

- That SLURM uses only the client nodes within the FabriCache file system partition to run jobs that access that file system.
- That no FabriCache server nodes are used to run jobs.
- That no nodes outside the FabriCache client nodes are used to run jobs accessing the FabriCache file system.
- On those client nodes, the FabriCache file system will be available—with all of the usual file system behavior—in the directory `/tmp/fabclient/xyz`.





# Chapter 13 Running and Monitoring Jobs

As soon as user accounts have been added to the system, any user with an account and password can log into the head node and use SLURM commands to run jobs on the nodes.

This chapter provides information about running and monitoring jobs that may be useful to the system administrator. In this section:

- Tips for Running SLURM Jobs
- Running SLURM MPI Jobs
- Raising Ulimits for SLURM Jobs

This chapter does not cover:

Information about how to build, use, and modify SLURM partitions. See *Creating and Using Partitions* on page 126.

General information about how to use SLURM to run applications. See the *Programming Guide* for details on using SLURM commands such as `srun` to run jobs.

## Tips for Running SLURM Jobs

These SLURM tips may be useful to the system administrator. For additional information about running SLURM jobs, see the *Programming Guide*.

### SLURM Job Logging Is On by Default

SLURM job logging is enabled. SLURM places messages regarding completed jobs in `/var/log/slurm-comp.log`.

This log keeps a record of the jobs that have been run.

### Turning SLURM Logging On or Off

To turn SLURM job logging on or off, make the appropriate changes as shown below in the file `/opt/sicortex/config/slurm-core.conf`.

### Turning SLURM Job Logging ON

The settings shown below are the default as the system is shipped.

```
# LOGGING / ACCOUNTING
SlurmctldDebug=3
SlurmdDebug=3
JobAcctType=jobacct/none
JobCompType=jobcomp/filetxt
JobCompLoc=/var/log/slurm-comp.log
ProctrackType=proctrack/linuxproc
```


### Turning SLURM Job Logging OFF

Modify the above section of the file so that it looks as shown below:

```
# LOGGING / ACCOUNTING
SlurmctldDebug=3
SlurmdDebug=3
JobAcctType=jobacct/none
JobCompType=jobcomp/none
```

## When to Specify the Full Path for an Executable

SLURM resolves the path of an executable on the machine where `srun` executes.

-  If you as system administrator are running SLURM jobs from the SSP, and if the path for an executable is different on the SSP than on the nodes, then you must specify the full path for the executable as found on the nodes.

## Finding Out How Many Files a Job Has Open

To find out how many open files a job has, you can use the following commands:

```
cd /proc/<pid>/fd
ls |wc
```

These commands are helpful in locating outliers and transient behavior.

## Running SLURM MPI Jobs

If you are running SLURM MPI jobs from the SSP, a limitation in SLURM makes it necessary to set the `SLURM_SRUN_COMM_IFHN` environment variable to ensure the jobs work properly.

### How `SLURM_SRUN_COMM_IFHN` Works

Before you initiate a job from the SSP to run on a node, you must set the `SLURM_SRUN_COMM_IFHN` environment variable on the SSP (in your shell on the command line):

```
$ export SLURM_SRUN_COMM_IFHN=ssp
```

Then give the `srun` command:

```
sysadmin@ssp024:~/mpi>srun -p sc0 hello-mpi
Hello from mpi task number 0
Task 0 is running on the processor named sf0-m0n0.scsystem.
```

The job above is initiated from the SSP to run on a node, and the setting of the `SLURM_SRUN_COMM_IFHN` environment variable allows the nodes to route back to the SSP for their control connection.

If you issue the following command:

```
sysadmin@ssp024:~/mpi>srun -p sf0 --batch hello-mpi.sh
```

Then the batch file is submitted from the SSP, but when the job runs, the job itself is initiated by a node. Therefore, the other nodes need to route back to the initiating node (not to the SSP) for their control connection.

In this second case, the environment variable must **not** be set as `SLURM_SRUN_COMM_IFHN=ssp`, or the nodes will route to the SSP, which is incorrect.

The following example shows how to use the `unset` command to unset the variable:

```
sysadmin@ssp024:~/mpi>unset SLURM_SRUN_COMM_IFHN
sysadmin@ssp024:~/mpi>srun -p sf0 --batch hello-mpi.sh
srun: jobid 27141 submitted
sysadmin@ssp024:~/mpi>cat slurm-27141.out
Hello from mpi task number 0
Task 0 is running on the processor named sf0-m0n0.scsystem
```

It is not possible to set this environment variable simultaneously to facilitate running jobs both from the SSP and from the nodes as well.

## SLURM\_SRUN\_COMM\_IFHN Is Set to SSP by Default

Currently, the default SiCortex system initialization sets the SLURM\_SRUN\_COMM\_IFHN environment variable to point to the SSP.

This means that MPI jobs submitted on the SSP for immediate execution will work, however MPI jobs submitted on the SSP for batch execution will not work.

## Unset SLURM\_SRUN\_COMM\_IFHN for Batch Jobs

To remove SLURM\_SRUN\_COMM\_IFHN from your environment on the SSP, use the `unset` command, as shown in the example above. You can execute the `unset` command in any of the following ways:

Location of <code>unset SLURM_SRUN_COMM_IFHN</code> command	Effect on environment variable setting
In your <code>.bashrc</code> file	Removes it from all your shells.
On the command line, as shown in the example above	Removes it from just that shell.
In parentheses, like this: <pre>sysadmin@ssp024:~/mpi&gt;(unset SLURM_SRUN_COMM_IFHN; srun -p sf0 -b hello-mpi.sh)</pre>	Removes it from the environment for just that command.

## Batch Job Fails If Variable Is Not Set Correctly

If you do not unset the SLURM\_SRUN\_COMM\_IFHN environment variable before running an `srun --batch` job, the error messages from SLURM are mysterious.

For example, suppose you fail to unset the variable and you invoke the following job:

```
# srun -K --batch -p sc1-comp1 -n600 hpcc
```

The output captured by SLURM is a series messages (generally over 1000 such lines) that look like the following:

```
/net/home/jrandom/hpcc/hpcc.snow/hpcc: (^_&#65533;U: Unknown error
1099501590328
/net/home/jrandom/hpcc/hpcc.snow/hpcc: (^_&#65533;U: Unknown error
1099411347256
/net/home/jrandom/hpcc/hpcc.snow/hpcc: (^_&#65533;U: Unknown error
1099419408184
/net/home/jrandom/hpcc/hpcc.snow/hpcc: (^_&#65533;U: Unknown error
1099402958648
/net/home/jrandom/hpcc/hpcc.snow/hpcc: (^_&#65533;U: Unknown error
1099440248632
.
.
srun: error: sc1-m2n5: task312: Exited with exit code 128
srun: error: sc1-m3n6: task474: Exited with exit code 128
srun: error: sc1-m2n7: task329: Exited with exit code 128
srun: error: sc1-m2n8: task330: Exited with exit code 128
srun: error: sc1-m3n9: task494: Exited with exit code 128
```

## Raising Ulimits for SLURM Jobs

The nodes currently have soft/hard ulimits for open files, as shown below:

```
sysadmin@sc1-m0n0:~>ulimit -S -n; ulimit -H -n
8192
10000
```

But SLURM jobs on the nodes actually receive a much lower open file ulimit:

```
sysadmin@sc1-m0n0:~>srun -p sc1 bash -c 'ulimit -S -n; ulimit
-H -n'
1024
1024
```

ulimits for SLURM jobs are inherited from `slurmstepd`, which inherits them from `slurmd`, which inherits them from `postinit`. So if you want to raise the hard ulimit for a SLURM job, you have to raise it in `postinit` before it spawns `slurmd`, and then reboot the nodes to activate the higher limit.

1. Connect to the SSP.
2. `su` to root.
3. `cd /opt/sicortex/bootscripts/`

4. Determine which directory you are currently booting from. If you haven't specified an alternative bootscript directory to `scboot`, then `scboot` uses the default bootscript directory pointed to by the `/opt/sicortex/bootscripts/default` symlink. For this example, suppose it's 2.0.0.3.45281.
5. To make a copy of your bootscripts directory, enter a command similar to this:

```
sysadmin@ssp027:/opt/sicortex/bootscripts>cp -a 2.0.0.3.45281 2.0.0.3.45281.my_ulimits
```

6. Edit the `postinit` script. For example:

```
sysadmin@ssp027:/opt/sicortex/bootscripts>cd 2.0.0.3.45281.my_ulimits/var/state/  
vi postinit
```

7. Locate the line where `slurmd` is started. It should look something like this:

```
(ulimit -l 64000; /usr/sbin/slurmd -c)
```

8. Change ulimits as necessary:

```
(ulimit -l 64000; ulimit -n 8192; /usr/sbin/slurmd -c)
```

This line executes as `root`, so you can set any ulimits you like.

9. Reboot the nodes with this command:

```
SCBOOT_BOOT_SCRIPTS=/opt/sicortex/bootscripts/2.0.0.3.45281.my_ulimit scboot -p <my_partition>
```

You'll need to use this `scboot` command line every time you reboot the nodes.

For additional information about ulimits, see `man ulimit`.

# Chapter 14 Monitoring the System

The SiCortex system provides monitoring of key aspects of system health and performance, at both the hardware and software levels. In this chapter:

- Overview of System Monitoring Features
- How System Monitoring Is Implemented
- Configuring System Monitoring
- Environmental Data Logged to `policyd.log`
- Locating Log Files

## Overview of System Monitoring Features

This section describes the system monitoring features. The policy daemon, `policyd`, implements the system's monitoring features.

**Alert conditions** The SiCortex system monitoring feature generates alerts for these conditions:

- Over temperature conditions
- Power supply voltages out of tolerance
- Node kernel panics
- Node memory ECC errors
- Node communications fabric problems
- Problems with the monitoring system itself (for example, `policyd` is not running)

**Notifications** The system monitoring facility (`policyd`) communicates with the user in two ways:

- Nagios alerts
- Log file entries

There is not a one-to-one correspondence between log file entries and Nagios alerts. In general, the log file provides more detail than Nagios alerts.

## Nagios Alerts

The system is designed to report alerts to a Nagios server running elsewhere on your network. You must configure `policyd` to send alerts to your specific Nagios server. This is explained in *Specifying Your Nagios Server to Receive Alerts* on page 217.

`policyd` delivers SiCortex system alerts to Nagios as passive service checks.

**Example** The following example of a SiCortex system alert, as sent to Nagios, shows a warning that a voltage is out of spec with a value of 948 millivolts.

```
***** Nagios *****

Notification Type: PROBLEM

Service: SiCortex power
Host: ssp024
Address: 10.4.0.24
State: WARNING

Date/Time: Wed Nov 28 12:15:05 EST 2007

Additional Info:

1196270098 env sf0-msp0 MspEnv_Power_Po1_02 948 mV
```

## Log File Entries

The log file entries from `policyd` vary depending on the source and the event or condition. Here are some examples of log file entries that `policyd` could post in `/var/log/policyd.log`:

```
ssp040 log # tail policyd.log
[2008-07-23 13:16:36] WARNING: sicortex.policyd.Temperature: TIMEOUT on scx-msp22
MspEnv_Temp_AD_04: no reading for 180 seconds
[2008-07-23 13:16:36] WARNING: sicortex.policyd.Temperature: TIMEOUT on scx-msp22
MspEnv_Temp_AD_05: no reading for 180 seconds
```



```
[2008-07-23 13:30:14] INFO: sicortex.policyd: 1216834213 env scx-msp34 MspEnv_Temp_Node_08
45250 mC
[2008-07-23 13:30:14] INFO: sicortex.policyd.Temperature: RECOVERY on scx-msp34
MspEnv_Temp_Node_08
Why? 1216834213 env scx-msp34 MspEnv_Temp_Node_08 45250 mC
[2008-07-23 13:30:14] INFO: sicortex.policyd: 1216834213 env scx-msp34 MspEnv_Temp_Node_09
51250 mC
[2008-07-23 13:30:14] INFO: sicortex.policyd.Temperature: RECOVERY on scx-msp34
MspEnv_Temp_Node_09
Why? 1216834213 env scx-msp34 MspEnv_Temp_Node_09 51250 mC
```

**Warning example** This is an example of a warning message sent through Nagios:

```
[2008-04-01 16:47:41] WARNING: sicortex.policyd.Temperature: WARNING on
sf0-msp0
Why? Temperature above 75 C
1207082859 env sf0-msp0 MspEnv_Power_Po1_00 76000 mC
1207082860 env sf0-msp0 MspEnv_Power_Po1_00 76000 mC
1207082861 env sf0-msp0 MspEnv_Power_Po1_00 76000 mC
```

**Shutdown example** This is an example of a shutdown message sent through Nagios:

```
[2008-04-01 16:49:47] ERROR: sicortex.policyd.Power: SHUTTING DOWN sf0-msp0
Why? Voltage out of range: 10.800 V - 13.200 V
1207082985 env sf0-msp0 MspEnv_Power_Dpm_Ibv 13301 mV
1207082986 env sf0-msp0 MspEnv_Power_Dpm_Ibv 13301 mV
1207082987 env sf0-msp0 MspEnv_Power_Dpm_Ibv 13301 mV
```

## How System Monitoring Is Implemented

All temperature and power monitoring is handled by the policy daemon, `policyd`.

**Policy Daemon** `policyd` takes in events from the rest of the system, corresponding to conditions in the hardware and software which are of interest to the system administrator. Examples of such conditions include:

- Out of range temperature readings
- Fan problems
- Fabric link problems
- ECC errors
- Anomalous conditions on the network

The policy daemon examines the SiCortex system's event stream and takes actions based on the event values or combinations of event values. Possible recovery actions include:

- `policyd` may notify the system administrator via Nagios
- The system administrator may selectively reboot a node
- The system administrator may declare a node or module inactive and exclude it from the node boot process until it can be fixed

All messages sent to `policyd` are now logged at log level INFO, which is enabled by default.

# Configuring System Monitoring


Configuration of system monitoring should be done immediately after booting the SSP and nodes. While it is not required for running the system, until you configure it, you don't get any notifications to or from Nagios, and you don't get any customizations you might want in how the notifications occur.

## System Monitoring Files

SiCortex system monitoring features refer to these two SiCortex system files:

- `/etc/conf.d/policyd`
- `/etc/conf.d/watchdogd`

 **Do not change the thresholds in `policyd` or `watchdogd`, unless instructed to do so by SiCortex Customer Support.**

 Configuration changes made to the `policyd` and `watchdogd` files will not persist across a software upgrade.

You configure `policyd` and `watchdogd` by putting command line options into the `POLICYD_OPTS` and `WATCHDOGD_OPTS` variables, respectively.

**policyd as shipped** As shipped, the file `/etc/conf.d/policyd` looks like this:

```
# $Id: conserver 44661 2007-09-17 14:44:15Z dbertrand $  
## policyd configuration file
```

```

# Options:
#
# --envmond-port    envmond port      (default 8081)
# --watchdogd-port  watchdogd port    (default 8083)
# --plugin-dir      plugin directory  (default /opt/sicortex/policyd/plugins)
# --heartbeat       heartbeat interval (default 120 seconds)
# --log             log destination: syslog, stderr, or file
#                  (default file)
# --log-dir         log directory    (default /var/log/)
# --log-level       log level: debug, info, warning, error, or critical
#                  (default error)
# --send-nsca       Nagios Service Check Acceptor client
#                  (default /usr/nagios/libexec/send_nsca)
# --nagios-server   Nagios server    (default localhost)
# --nagios-port     Nagios port      (default 5667)
# --nagios-config   Nagios config file (default /etc/nagios/send_nsca.cfg)

POLICYD_OPTS=""

```

**watchdogd as shipped** As shipped, the file `/etc/conf.d/watchdogd` looks like this:

```

# $Id: conserver 44661 2007-09-17 14:44:15Z dbertrand $

## watchdogd configuration file

# Options:
#
# --envmond-host    envmond host (default localhost)
# --envmond-port    envmond's watchdog port
# --policyd-host    policyd host (default localhost)
# --policyd-port    policyd's watchdog port
# --timeout         timeout value (default 300 seconds)
# --nagios-config   nagios configuration file
# --nagios-server   nagios host (default localhost)
# --nagios-port     nagios port
# --send-nsca       nsca utility
# --debug          verbose log output

WATCHDOG_OPTS=""

```

## Specifying Your Nagios Server to Receive Alerts

The SiCortex system monitoring feature sends alerts to a Nagios server that is assumed to be running somewhere on your network. To enable alerts, and to view SiCortex system monitoring messages and alerts in Nagios, you need to configure the `policyd` and `watchdogd` files to identify your Nagios server:

In most cases, you will need to set `--nagios-server` in both files. In most cases, you will not need to set any other options related to system monitoring.

**Modify policyd** In the file `/etc/conf.d/policyd`, modify the `POLICYD_OPTS` variable as shown below, to insert a command line option that specifies the name of your Nagios server:

```
POLICYD_OPTS="--nagios-server my-nagios-server.my-company-com"
```

**Modify watchdogd** In the file `/etc/conf.d/watchdogd`, modify the `WATCHDOGD` variable as shown below, to insert a command line option that specifies the name of your Nagios server :

```
WATCHDOGD_OPTS="--nagios-server my-nagios-server.my-company-com"
```

## Environmental Data Logged to `policyd.log`

Environmental data is logged to this file:

```
/var/log/policyd.log
```

Environmental messages explain what even happened, and what if any action the system has taken as a result of the event.

Here are examples of the `WARNING` and `SHUTDOWN` messages from `policyd.log`:

```
[2008-04-01 16:47:41] WARNING: sicortex.policyd.Temperature:
WARNING on sf0-msp0
Why?    Temperature above 75 C
        1207082859 env sf0-msp0 MspEnv_Power_Po]_00 76000 mC
        1207082860 env sf0-msp0 MspEnv_Power_Po]_00 76000 mC
        1207082861 env sf0-msp0 MspEnv_Power_Po]_00 76000 mC
[2008-04-01 16:49:47] ERROR: sicortex.policyd.Power: SHUTTING
DOWN sf0-msp0
Why?    Voltage out of range: 10.800 V - 13.200 V
        1207082985 env sf0-msp0 MspEnv_Power_Dpm_Ibv 13301 mV
        1207082986 env sf0-msp0 MspEnv_Power_Dpm_Ibv 13301 mV
        1207082987 env sf0-msp0 MspEnv_Power_Dpm_Ibv 13301 mV
```

The `/var/log/policyd.log` file rotates daily into a series of archived files:

```
policyd.log.1.gz
policyd.log.2.gz
...
policyd.log.14.gz
```

`policyd` log files are reopened on `kill -HUP`.

The following related log files all use the same rotation scheme for archiving older versions:

```
/var/log/envmond.log
/var/log/kernmond.log
/var/log/watchdogd.log
```

## Raw Environmental Data in RRD Databases

The raw environmental monitoring data is stored in RRD databases. You can dump the raw data from an RRD database with a command of the following form:

```
rrdtool fetch \
/var/log/RRD/Temperature/sc0-msp0/MspEnv_Power_Po1_00.rrd \
LAST
```

For more information, see the RRDtool documentation at the following website:

<http://oss.oetiker.ch/rrdtool/doc/index.en.html>

## Locating Log Files

There is a variety of log files that can be used to monitor the system. This section describes the various log files and where to find them. All files in this section are on the SSP.

### Log File Handling

- Logs are cumulative** All log files listed here are cumulative. Messages accumulate in the log file until the file rolls over to a new version, as explained below. So for example, the console logs for the nodes provide a history of events on that node including each time the node has been booted.
- yyyymm[dd] rollover** Many log file names incorporate a date value in the form of *yyyymm* or *yyyymmdd*. When the date changes at the start of a new month or year, or day in the case of *yyyymmdd*, the system starts a new instance of that log file using the new date value in the file name.
- Permanent log files** Some log file names do *not* include a date value, such as `sboot.log`. In these cases, the system appends data to the same log file forever. These are log files that grow slowly.
- Unused log files compressed** The system runs a cron job that compresses any log files that have not been accessed in 7 days.

**Reading compressed log files** The system uses `gzip` to compress the files. You can use `gunzip` to uncompress a compressed log file, or you can use `zcat` to read the file in its compressed form.

## Tailing Any Log File

As on any Linux system, to check the completion of any operation, `tail` the log file for that operation. For example:

```
$ tail <file.log>
```

will show you the last few lines of a log file

```
$ tail -n 100 <file.log>
```

will show you the last 100 lines of the log file.

```
$ tail -f <file.log>
```

will show you the end of the log file as it currently exists, and will display new lines as they are written to the log file.

```
$ watch sinfo
```

You can use the `watch` command, as above, to repeatedly display the output of a given command.

A good way to see which log files are being used is by checking the modification timestamp. You can use the following method:


```
ls -l --reverse --sort=time /var/log
ls -l --reverse --sort=time /var/log/sf0
ls -l --reverse --sort=time /var/log/nodes-200708
```

A shorter command that provides a compressed version of the same output is:

```
ls -lt | head
```

This command lists the ten most recently changed files in the directory.

## Log Files by Purpose and Location

Log Name or Type	Purpose	File Location and Name Format
<b>SSP syslog</b>	Syslog messages from the SSP accumulate in files with names in this format. A new syslog file is created each day. When the SSP boots, the version of system software that was booted is logged to this file.	<p>/var/log/messages-&lt;yyyymmdd&gt;</p> <p>where &lt;yyyymmdd&gt; is a four digit year, a two-digit month, and a two-digit day.</p> <p>For example: /var/log/messages-20080125</p>
<b>MSP logs</b>	The MSP log files contain the combined console and syslog messages for the MSP.	<p>/var/log/msp-messages-&lt;yyyymm&gt;</p> <p>where &lt;yyyymm&gt; is a four digit year and a two-digit month.</p> <p>For example: /var/log/msp-messages-200708</p>
<b>Pre-initialization node messages</b>	During the pre-initialization (preinit) phase of the boot process, node messages are sent to this file.	<p>/var/log/msp-messages-&lt;yyyymm&gt;</p>
<b>Node syslogs</b>	<p> [As of this release] Syslog on the nodes is turned on by default. The node kernel messages are sent to the node console logs.</p> <p>There is one node syslog for each node. Messages in the node syslog apply to that entire node, including messages that indicate whether the node is up or down.</p>	<p>/var/log/nodes-&lt;yyyymm&gt;/&lt;hostname&gt;.log</p> <p>where &lt;hostname&gt; is: &lt;partition&gt;-m&lt;module-number&gt;n&lt;node-number&gt;</p> <p>For example: /var/log/nodes-200708/sc1-m0n0.log</p>
<b>MGTnet node log files</b>	MGTnet nodes write log files of the following name:	<p>/var/log/mgtstate/&lt;partition&gt;-mgt&lt;0-4&gt;.log</p>

## Log Files by Purpose and Location

Log Name or Type	Purpose	File Location and Name Format
<b>/var/log/ log files</b>	All the other log files reside on the SSP in the following directory.  This directory contains the logs described below.	<code>/var/log/&lt;partition-name&gt;/&lt;name-of-log&gt;</code>
<b>Node console logs</b>	In addition to the node syslogs described above, conserver creates one console log file for each node. This is not a true syslog, but it receives kernel messages from the node, and also output from the startup scripts. The console log for a given node contains the log of the virtual console for that node.	<code>/var/log/&lt;partition&gt;/&lt;hostname&gt;.console</code>  where <hostname> is: <code>&lt;partition&gt;-m&lt;module-number&gt;n&lt;node-number&gt;</code>  For example: <code>/var/log/sc1/sc1-m0n0.console</code>  If the nodes are up but you cannot connect to their consoles, you need to restart <b>sconserver</b> and <b>conserver</b> . Type the following command from the SSP to restart both these services:  <code># /etc/init.d/sconserver restart</code>
<b>MFD log</b>	MFD, the master fabric daemon, keeps one log file, called mfd.log.  All messages related to the hardware components that implement the interconnect fabric are logged to mfd.log.  One of the boot steps that must complete successfully is the initialization of the fabric. The fabric interconnects all the nodes on the system.  The master fabric daemon (mfd) is the process that controls the fabric.	<code>/var/log/&lt;partition&gt;/mfd.log</code>  When the MFD is finished pushing software out to the nodes, it logs a message to mfd.log that says:  <code>All nodes finished for &lt;partition&gt;; exiting. PROGRESS: All nodes finished.</code>  In the example below, sc1 is the name of the partition:  <pre>sysadmin@ssp026 /var/log/sc1 \$ tail mfd.log &lt;&lt;&lt; 54 (fd=46): &lt;done&gt; node 54 says it's done &lt;&lt;&lt; 55 (fd=105): &lt;done&gt; node 55 says it's done &lt;&lt;&lt; 58 (fd=49): &lt;done&gt; node 58 says it's done &lt;&lt;&lt; 59 (fd=6): &lt;done&gt; node 59 says it's done All nodes finished for sc1; exiting. PROGRESS: All nodes finished</pre>
<b>dnsmasq log</b>	The DNS server, dnsmasq, creates its own log file.  This log receives dnsmasq syslog data, separate from the SSPs other messages.	<code>/var/log/dnsmasq-&lt;yyyymm&gt;</code>



Log Name or Type	Purpose	File Location and Name Format
<b>scboot log</b>	A record of scboot invocations is kept in this log file.  Its contents are kept separate from syslog data.	/var/log/scboot.log
<b>SLURM job log</b>	SLURM job logging is enabled (the default was changed as of v2.2). Messages are logged in this file.	/var/log/slurm-comp.log
<b>/dev/log</b>	On SiCortex system nodes, /dev/log is symlinked to /var/tmp/syslog, so that every node can bind to its own /var/tmp/syslog, and applications can log messages by connecting to /dev/log.	<p>SYSLOG messages for the nodes appear in this directory on the SSP:</p> <p style="padding-left: 40px;">/var/log/nodes-&lt;yyyymm&gt;/&lt;node-name&gt;.log</p> <p>On a typical Linux system, /dev/log is a special file where applications send log data (i.e. applications write to /dev/log). The system logger (syslog) reads log data from /dev/log.</p> <p>On SiCortex systems where the root file system is served with NBD, the nodes are unusual in that the root file system is shared and is read-only. This means that a node cannot change the root file system.</p> <p>So on the SiCortex system, /dev/log needs to be node-specific, because it needs to contain node-specific log data. The /var sub-directory is a tmpfs (temporary in-memory file system) mount point that is specific to each node and is also writeable.</p> <p>The SiCortex system nodes make /dev/log a symlink to /var/tmp/syslog so that logging to /dev/log can be node-specific.</p>

## Other SiCortex Log Files

The table below lists other log files specific to the SiCortex system:

Log File	Description
/var/log/part/bamf_diagcomm.log	Logs MSP RPC protocol messages generated by scboot
/var/log/diagcomm_client.log	Logs MSP RPC protocol messages generated by envmond

## Other SiCortex Log Files

<b>Log File</b>	<b>Description</b>
/var/log/envmond.log	Generated by the environmental monitoring daemon (envmond).
/var/log/ev1d.log	
/var/log/install-ssp.log	
/var/log/kernmond.log	Generated by the kernel message monitoring daemon (kernmond).
/var/log/msp-env/*	MSP environmental syslog data.
/var/log/mspenv.log	Generated by mspenv (mspenv is a command line utility used by envmond to communicate with the MSPs).
/var/log/mspstate/*	
/var/log/policyd.log	Logs messages from policyd about system monitoring conditions and events.
/var/log/powerutil.log	Generated by the command line power utility (powerutil).
/var/log/scboot/*	
/var/log/sconserver.log	Generated by the SiCortex console server multiplexer.
/var/log/sicortex.log	
/var/log/watchdogd.log	Generated by the environmental monitor watchdog daemon.

# Chapter 15 Checking System Status

This section provides guidance about improving the performance of your SiCortex system, and debugging performance issues. In this chapter:

- Checking Performance on a Node
- Checking Performance System-Wide

## Checking Performance on a Node

This section provides some tips for checking on system performance.

### Linux performance measurement commands

You can use the standard Unix tools to measure system performance including:

- `top`
- `uptime`

These tools provide feedback about what's going on on an individual node.

### Load average

The `top` and `uptime` commands display the load average in different ways. Load average is a quick method for calculating system load. For a general introduction to load average, see this article:

[http://en.wikipedia.org/wiki/Load\\_\(computing\)](http://en.wikipedia.org/wiki/Load_(computing))

### top example

To see what's happening on a specific node, `ssh` to that node and run the `top` command. `Top` is an interactive utility that opens a new monitoring window. The window displays what is happening on the node at intervals of 3 seconds (default). For complete information, see `man top`.

The example below shows the output of `top` on a node of an SC5832:

```
top - 09:29:39 up 39 min,  3 users,  load average: 0.60, 1.79,
1.22
Tasks: 397 total,   2 running, 395 sleeping,   0 stopped,   0 zom-
bie
Cpu(s):  5.5% us, 13.0% sy,  0.0% ni, 66.4% id,  5.7% wa,  2.0%
hi,  7.5% si
Mem:  2058336k total,   829404k used, 1228932k free,   87108k
buffers
Swap: 3998392k total,         0k used, 3998392k free,   345312k
cached

  PID USER  PR  NI  VIRT  RES  SHR  S  %CPU  %MEM  TIME+  COMMAND
 7825 root  15   0 18488 6892  636  R  44.4   0.3  1:33.33 syslog-ng
 8598 root  15   0     0     0     0   S   1.0   0.0  0:00.24 nfsd
 8604 root  15   0     0     0     0   S   0.3   0.0  0:00.48 nfsd
```

```

8605 root 15 0 0 0 0 S 0.3 0.0 0:00.43 nfsd
8615 root 15 0 0 0 0 S 0.3 0.0 0:00.39 nfsd
8620 root 15 0 0 0 0 S 0.3 0.0 0:00.79 nfsd
9855 root 15 0 27528 2644 1204 S 0.3 0.1 0:00.14 conserver
9874 root 15 0 27528 2644 1204 S 0.3 0.1 0:00.15 conserver
14430 root 15 0 9356 4292 852 R 0.3 0.2 0:03.35 top

```

## Checking Performance System-Wide

To see what's going on on the system as a whole, the following SLURM commands are helpful:

**sinfo** `sinfo` reports the state of partitions and nodes managed by SLURM. It has a wide variety of filtering, sorting, and formatting options.

`sinfo` displays a summary of the information available about partitions and nodes, such as partition names, nodes per partition, and cores per node. `sinfo` does not provide information on jobs.

**sinfo -i** `sinfo` with the `-i` option is an economical alternative to using the `watch` utility to run `sinfo`. It prints out the `sinfo` status at a specified interval. For example, the command:

```
watch sinfo -p scx
```

is substantially equivalent to:

```
sinfo -i 2 -p scx
```

but the latter should take up significantly less CPU time.

**srun** `srun` is used to submit a job for execution or initiate job steps in real time. `srun` has a wide variety of options to specify resource requirements, including: minimum and maximum node count, processor count, specific nodes to use or not use, and specific node characteristics (so much memory, disk space, certain required features, etc.). A job can contain multiple job steps executing sequentially or in parallel on independent or shared nodes within the job's node allocation.

You can use the `srun` command to run the same command on all nodes at once. For example, you can use `srun` to run `uptime` on all the nodes of an SC648 to see the load average of each node:

```
sysadmin$ssp001% srun -p sc1 -N 108 uptime
```

This command runs `uptime` on all 108 nodes of partition `sc1` on an SC648.

# Chapter 16 Managing Installed Software

The SiCortex system ships with a Gentoo Linux software package. SiCortex provides all software on an open-source basis.

This section describes how to add software to a SiCortex system. In this chapter:

- Gentoo Packaging Concepts
- Searching Packages
- Installing and Customizing Packages
- Querying Packages
- Managing Packages on a Running Node
- Removing Packages
- Installing the n32 ABI Environment
- Licensing Third-Party Software

## Gentoo Packaging Concepts

### Introduction

The Linux distribution that runs on the SSP and the cluster nodes is a form of Gentoo Linux that is customized for the SiCortex system. It is important to understand that there are two separate Gentoo distributions that are part of every SiCortex system: one for the SSP and one for the nodes. The Gentoo distribution for the nodes is stored on the SSP by default at `/opt/sicortex/rootfs/default/` (unless you specified an alternate rootfs in the `sicortex-install.conf` file, see *10b—Using a Non-Default Root File System Image* on page 84.)

The structure of the node root-filesystem is very similar to the one for SSP but the executables and libraries for the SSP are built for the x86\_64 architecture and on the nodes they are built for the MIPS64 architecture. The software for the SSP and for the nodes are managed separately by the Gentoo package management tools.

For detailed information about Gentoo and Gentoo package management, consult the Gentoo handbook:

[www.gentoo.org/doc/en/handbook/](http://www.gentoo.org/doc/en/handbook/)

## Portage and Portage Files

Portage is the name of Gentoo's extremely powerful and flexible package management system. In one sense, it is the core of Gentoo, because Gentoo is a meta-distribution and the portage system is the interface that is used to construct customized distributions of Gentoo. Portage is written in Python and Bash and therefore fully visible to the users as both are scripting languages.

There are several files and directories that are relevant to the operation of portage. The files are described in more detail in following sections.

<b>/directory/file</b>	<b>Description</b>
/usr/portage	The portage tree
/opt/sicortex/overlays	SiCortex portage tree overlays
/etc/make.conf	Portage configuration for the SSP
/var/db/pkg/	The installed package database for the SSP
/opt/sicortex/rootfs/default/var/db/pkg	The installed package database for the nodes
/opt/sicortex/rootfs/default/etc/make.conf	Portage configuration for running portage on the nodes
/opt/sicortex/rootfs/default/etc/config_root/etc/make.conf	Portage configure for running portage on the SSP while targeting the nodes.

## Portage Commands

There are two main commands that are used to invoke portage:

<b>emerge</b>	<b>emerge</b> is an end-user oriented program and can be used to search for, install and remove packages. Also, <b>emerge</b> provides automatic dependency resolution.
<b>ebuild</b>	<b>ebuild</b> is a lower level interface to portage that is targeted more at developers of Gentoo software packages (called <b>ebuilds</b> ). The <b>ebuild</b> command operates directly on an <b>ebuild</b> file (rather than a package name as <b>emerge</b> does), and it does not do automatic package dependency resolution.

The `emerge` command can be used for basic package searches. However, for more advanced searching and to query installed packages the `equery` and `epm` commands are provided. The `epm` command has a very similar syntax to the `rpm` command on `rpm`-based Linux distributions.

By default all portage commands operate against the root filesystem of the system they are run on. When the commands are run on the SSP they use configuration in `/etc/make.conf` and `/etc/portage/` and operate against the package database at `/var/db/pkg`. When the same commands are run on a booted node they also use configuration from `/etc/make.conf` and `/etc/portage` and operate against `/var/db/pkg`.

However these files come from the following locations on the SSP:

```
/opt/sicortex/rootfs/default/etc/make.conf
/opt/sicortex/rootfs/default/etc/portage/
/opt/sicortex/rootfs/default/var/db/pkg
```

(Or elsewhere, if you specified an alternate rootfs in the `sicortex-install.conf` file; see *10b—Using a Non-Default Root File System Image* on page 84.)

Some portage commands have wrapper scripts that allow them to be run on the SSP but operate against the root filesystem for the nodes. The current portage commands that have this capability are `scemerge`, `scebuild` and `scepm`. When the `scemerge` and `scebuild` commands are used they actually use a `make.conf` configuration from a third location that causes all ebuild phases to be run in cross-compile mode. This file is located at `/opt/sicortex/rootfs/default/etc/config_root/etc/make.conf`.

## Ebuilds, Portage Tree and Overlays

On a Gentoo system, all software packages are represented by `ebuild` files (`ebuilds`). The ebuilds are used by portage to install, search, query, upgrade and remove software packages. `Ebuilds` are actually just `bash` shell code that is sourced (executed) by portage within a special environment. Each `ebuild` contains information such as description, home page, source URL, license, package options (`IUSE`), dependencies, which architectures this package is valid for (`KEYWORDS`), along with information about how to unpack, patch, compile and install the package. The `ebuilds` that are available on a SiCortex system are stored in two places: the portage tree and portage tree overlays.

The portage tree is stored at `/usr/portage` on the SSP. The portage tree is a collection of ebuilds provided by the Gentoo community.

Currently the portage tree contains over 24,000 ebuild files that represent over 13,000 different software packages (each software package may have multiple `ebuilds` for different versions of that software package).

The other location where `ebuilds` are stored on a SiCortex system is in portage tree overlay which are located at `/opt/sicortex/overlays`. This location contains software packages that were created or modified by SiCortex specifically to support the operation of the SiCortex system.

`Ebuilds` in the portage tree and overlays are grouped into categories based on the type of software package. The `/usr/portage/app-admin` directory contains applications related to system administration. The category of an `ebuild` is considered to part of its name. This is important because sometimes the package name itself is ambiguous without the category. For example: `sys-libs/zlib` and `dev-haskell/zlib`.

## Ebuild Versions and Selection Criteria

In the portage tree and portage overlays there may be multiple `ebuild` versions defined for a single software package. When the `ebuild` command is used then the full path to the a specific `ebuild` file must be used. However, the `emerge` command allows `ebuilds` to be referenced using partial information. There are several factors that determine which version (`ebuild` file) of a package is selected.

Within the each ebuild is a `KEYWORDS` variable that determines which architectures this version of the package works on and has been tested on. For example an `ebuild` might contains `KEYWORDS="amd64 ~mips"`. The 'amd64' value means that this `ebuild` has been tested and is considered stable on amd64 (x86\_64) Gentoo systems. The '~mips' (tilde mips) value means that this package has been shown to work on MIPS systems but is not consider tested and stable yet.

The `ACCEPT_KEYWORDS` environment variable is used by `emerge` to determine which `ebuild` versions are valid for selection. This variable can be set globally in the `/etc/make.conf` configuration file, or it can be set per package in the `/etc/portage/package.keywords` file, or per `emerge` call by setting `ACCEPT_KEYWORDS` environment variable. See the `portage` and `make.conf` man pages for more information. The default `ACCEPT_KEYWORDS` value for the SSP is "amd64" and for the nodes it is "mips ~mips".

Another criterion that the `emerge` command uses is any version information that is specified on the command line. To indicate that a requested package has version information in addition to the name, the parameter



must be prefixed with an equal sign '='. For example, ``emerge =netcdf-3*`` means that `portage` should only search for packages that are named "netcdf" and the first element of `ebuild` version must be "3".

The `emerge` command selects the most recent package version that matches any version information specified on the command line and where the `ebuild`'s `KEYWORDS` values match one of the `ACCEPT_KEYWORDS` values. If the same version of the package matches in both the `portage` tree and in an overlay, then `portage` will use the `overlay ebuild` file.

## Searching Packages

To search for a package by name you can use `emerge` or `equery`.

This example uses `emerge` to search for the `netcdf` package.

```
ssp # emerge -s netcdf
Searching...
[ Results for search key : netcdf ]
[ Applications found : 1 ]

* sci-libs/netcdf
  Latest version available: 3.6.2
  Latest version installed: [ Not Installed ]
  Size of files: 5,188 kB
  Homepage:      http://my.unidata.ucar.edu/content/software/netcdf/index.html
  Description:   Scientific library and interface for array oriented data access
  License:       UCAR-Unidata
```

This indicates that a package matching "netcdf" was found in either the `portage` tree or `portage` overlays (`emerge` search does not indicate which). The category of `netcdf` is "sci-libs" and version 3.6.2 is the most recent version where `ACCEPT_KEYWORDS` matches the `ebuild`'s `KEYWORDS` setting.

---

```
ssp # equery list -p -o zsh
[ Searching for package 'netcdf' in all categories among: ]
  * installed packages
  * Portage tree (/usr/portage)
[-P-] [M~] sci-libs/netcdf-3.6.1-r1 (0)
[-P-] [ ] sci-libs/netcdf-3.6.2 (0)
  * overlay tree (/usr/local/portage /opt/sicortex/overlays/auto-multilib /opt/sicortex/overlays/default /opt/sicortex/overlays/inhouse)
[--0] [ ] sci-libs/netcdf-3.5.0-r3 (0)
[--0] [ ] sci-libs/netcdf-3.6.0-r1 (0)
[--0] [M~] sci-libs/netcdf-3.6.1 (0)
[--0] [M~] sci-libs/netcdf-3.6.1-r800 (0)
```

The `equery` command shows all the matching `ebuild` versions in the `portage` tree and `portage` overlay (the `-p` and `-o` options). Note that packages with "[M~]" are masked because the `KEYWORDS` value in the `ebuild` contains a tilde (i.e. "~amd64") but the `ACCEPT_KEYWORDS` settings do not have a tilde (i.e. "amd64"). This is known as a keyword mask.

# Installing and Customizing Packages

The `emerge` command is used to install software packages. Note that `ebuilds` can often have surprising dependencies on other `ebuilds`. The `emerge` command will automatically install dependencies. Sometimes these dependencies will unintentionally upgrade packages that are critical for normal system operation, so it is important to run the command in `pretend` mode first, to show the complete list of packages selected before actually executing the `emerge` command.

Here is a list of some packages that you should avoid unintentionally upgrading (in descending order of risk):

```
sys-libs/glibc
sys-apps/baselayout
sys-apps/portage
net-nds/openldap
net-nds/nss_ldap
sys-auth/pam_ldap
sys-libs/*
dev-lang/perl
dev-lang/python
dev-libs/*
```

Most `ebuilds` have optional features which can be turned on and off using `USE` flags. `USE` flags can have a significant effect on package dependencies and are the most common way to tune the `emerge` command to avoid upgrading or installing dependent packages. The `USE` flags that an `ebuild` supports are specified in the `ebuild` file using the `IUSE` value. The global `USE` flag settings can be specified in `/etc/make.conf`, specified per package in `/etc/portage/package.use`, and per `emerge` call using the `USE` environment variable.

For more information about `USE` flags see the `portage` and `emerge` man pages.

Here is an example of running the `emerge` command in `pretend` mode:

```
ssp # emerge -p predict
```

These are the packages that would be merged, in order:

```
Calculating dependencies... done!
[ebuild N   ] sci-astronomy/predict-2.2.3 USE="nls -gtk -
xforms
-xplanet" MULTILIB_ABIS="amd64 -x86"
```

Note that the `predict` `ebuild` supports four different USE flags. The `'nls'` USE flag is on by default (system-wide setting). If you manually turn on the `gtk` flag and try the `pretend` again you will see that `emerge` automatically adds a number of dependent `ebuilds` to the list of packages that would be `emerged` (installed). This example also adds the verbose flag to show information about where the `ebuild` file is located (portage tree or overlays) and the size of the source tarballs that will be downloaded:

```
ssp # USE="gtk" emerge -p -v predict
```

These are the packages that would be merged, in order:

```
Calculating dependencies... done!
[ebuild NS  ] dev-libs/glib-1.2.10-r805 USE="-hardened" MULTILIB_ABIS="amd64 -x86" 411 kB [2]
[ebuild N   ] dev-perl/XML-Parser-2.34-r1 MULTILIB_ABIS="amd64 -x86" 224 kB
[ebuild N   ] dev-util/intltool-0.35.5 MULTILIB_ABIS="amd64 -x86" 131 kB
[ebuild N   ] x11-libs/gtk+-1.2.10-r12 USE="nls -debug" LINGUAS="-az -ca -cs -da -de -el -es
-et -eu -fi -fr -ga -gl -hr -hu -it -ja -ko -lt -nl -nn -no -pl -pt -pt_BR -ro -ru -sk -sl -sr
-sv -tr -uk -vi" MULTILIB_ABIS="amd64 -x86" 2,880 kB
[ebuild N   ] sci-astronomy/predict-2.2.3 USE="gtk nls -xforms -xplanet" MULTILIB_ABIS="amd64
-x86" 0 kB
```

Total size of downloads: 3,648 kB

Portage overlays:

- [1] /usr/local/portage
- [2] /opt/sicortex/overlays/auto-multilib
- [3] /opt/sicortex/overlays/default
- [4] /opt/sicortex/overlays/inhouse

Here is an example of the output from `emerging` the `predict` package (the `-a` option means show what would be done with confirmation).

```
ssp016 ~ # emerge -a predict
```

## Installing and Customizing Packages

These are the packages that would be merged, in order:

```
Calculating dependencies... done!
[ebuild N ] sci-astronomy/predict-2.2.3 USE="nls -gtk -xforms -xplanet"
MULTILIB_ABIS="amd64 -x86"

Would you like to merge these packages? [Yes/No] y

>>> >>> Emerging (1 of 1) sci-astronomy/predict-2.2.3 to /
>>> >>> Downloading 'http://mirrors.acm.cs.rpi.edu/gentoo/distfiles/predict-2.2.3.tar.gz'
--15:06:16-- http://mirrors.acm.cs.rpi.edu/gentoo/distfiles/predict-2.2.3.tar.gz
=> `/usr/portage/distfiles/predict-2.2.3.tar.gz'
Resolving mirrors.acm.cs.rpi.edu... 128.213.5.34
Connecting to mirrors.acm.cs.rpi.edu|128.213.5.34|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1,639,784 (1.6M) [application/x-tar]

100%[=====>] 1,639,784 256.20K/s ETA 00:00

15:06:22 (279.68 KB/s) - `/usr/portage/distfiles/predict-2.2.3.tar.gz' saved [1639784/1639784]

* checking ebuild checksums ;-)... [ ok ]
* checking auxfile checksums ;-)... [ ok ]
* checking miscfile checksums ;-)... [ ok ]
* checking predict-2.2.3.tar.gz ;-)... [ ok ]
>>> >>> Unpacking source (for ABI=amd64)...
>>> >>> Unpacking predict-2.2.3.tar.gz to /var/tmp/portage/predict-2.2.3/work
* Applying predict-2.2.3-xforms.patch ... [ ok ]>>> Source
unpacked (for ABI=amd64).
>>> >>> Compiling source in /var/tmp/portage/predict-2.2.3/work/predict-2.2.3 (for ABI=amd64)
...
* compiling predict
* compiling vocalizer
* compiling clients
* compiling kep_reload
>>> >>> Source compiled (for ABI=amd64).
>>> >>> Test phase [not enabled]: sci-astronomy/predict-2.2.3

>>> >>> Install predict-2.2.3 into /var/tmp/portage/predict-2.2.3/image/ category sci-astronomy
(for ABI=amd64)
>>> >>> Completed installing predict-2.2.3 into /var/tmp/portage/predict-2.2.3/image/ (for
ABI=amd64)

man:
gzipping man page: predict.1
strip: x86_64-pc-linux-gnu-strip --strip-unneeded
usr/bin/vocalizer
usr/bin/kep_reload
usr/bin/predict
>>> >>> Merging sci-astronomy/predict-2.2.3 to /
--- /usr/
--- /usr/share/
>>> >>> /usr/share/predict/
>>> >>> /usr/share/predict/vocalizer/
>>> >>> /usr/share/predict/vocalizer/visible.wav
>>> >>> /usr/share/predict/vocalizer/three.wav
...
...
>>> >>> /usr/share/predict/vocalizer/elevation.wav
>>> >>> /usr/share/predict/vocalizer/zero.wav
>>> >>> /usr/share/predict/vocalizer/forty.wav
```

```

>>> >>> /usr/share/predict/default/
>>> >>> /usr/share/predict/default/predict.db
>>> >>> /usr/share/predict/default/predict.qth
>>> >>> /usr/share/predict/default/predict.tle
--- /usr/share/man/
--- /usr/share/man/man1/
>>> >>> /usr/share/man/man1/predict.1.gz
--- /usr/share/doc/
>>> >>> /usr/share/doc/predict-2.2.3/
>>> >>> /usr/share/doc/predict-2.2.3/CREDITS.gz
>>> >>> /usr/share/doc/predict-2.2.3/NEWS.gz
>>> >>> /usr/share/doc/predict-2.2.3/CHANGES.gz
>>> >>> /usr/share/doc/predict-2.2.3/HISTORY.gz
>>> >>> /usr/share/doc/predict-2.2.3/README.gz
>>> >>> /usr/share/doc/predict-2.2.3/predict.ps.gz
>>> >>> /usr/share/doc/predict-2.2.3/INSTALL.kep_reload.gz
>>> >>> /usr/share/doc/predict-2.2.3/README.kep_reload.gz
>>> >>> /usr/share/doc/predict-2.2.3/predict.pdf.gz
--- /usr/bin/
>>> >>> /usr/bin/vocalizer
>>> >>> /usr/bin/kep_reload
>>> >>> /usr/bin/predict
>>> >>> /usr/bin/predict-update
>>> >>> /usr/share/predict/vocalizer/vocalizer -> /usr/bin/vocalizer
* to use the clients the following line will
* have to be inserted into /etc/services
* predict    1210/udp
* the port can be changed to anything
* the name predict is what is needed to work
* after that is set run 'predict -s'
*
* to get list of satellites run 'predict-update'
* before running predict this script will also update
* the list of satellites so they are up to date.
>>> >>> Regenerating /etc/ld.so.cache...
>>> >>> sci-astronomy/predict-2.2.3 merged.
>>> >>> Recording sci-astronomy/predict in "world" favorites file...

>>> >>> No packages selected for removal by clean.

>>> >>> Auto-cleaning packages...

>>> >>> No outdated packages were found on your system.

* GNU info directory index is up-to-date.

```

The output from **portage** itself is generally prefixed with ">>>". The other output is from the stages defined in the ebuild itself. The phases of an emerge run are:

- Depend: Package dependencies are determined emerged first
- Fetch: The source tarball(s) for the package is downloaded.
- Unpack: The source tarball(s) is unpacked and possibly patched.
- Compile: The sources are configured and built.
- Install: The built package is installed to a temporary location.

- Merge: The installed files are merged to the real file-system.

## Querying Packages

The `equery` and `epm` commands can both be used to query information about installed packages. The `epm` command has a very similar syntax to the `rpm` command on `rpm` distributions. The `epm` command is also much faster but it has some serious limitations such as only being able to query packages by name without category or version. The `equery` command does not have this limitation. The two commands will often have during order for output items.

Here are some package query examples:

List all installed packages:

```
ssp # equery list
[ Searching for all packages in all categories among: ]
* installed packages
* [I--] [ ] app-admin/apache-tools-2.2.6 (0)
* [I--] [ ] app-admin/conserver-8.1.14-r1 (0)
* [I--] [ ] app-admin/eselect-1.0.10 (0)
* [I--] [ ] app-admin/eselect-emacs-1.1 (0)
...
```

```
-----or-----
ssp # epm -qa
ptables-1.3.8-r1
sicortex-romfstool-3.0.0.1.43301-r1
sicortex-sboot-3.0.0.48.48465-r16
sicortex-sample-scripts-3.0.0.2.46122-r1
grub-0.97-r3
sicortex-msp-libmspxdr-3.0.0.3.47489-r3
...
```

List all files owned by the `predict` command:

```
-----
ssp # equery files predict
[ Searching for packages matching predict... ]
* Contents of sci-astronomy/predict-2.2.3:
*/usr
/usr/bin
/usr/bin/kep_reload
```

```

/usr/bin/predict
/usr/bin/predict-update
/usr/bin/vocalizer
/usr/share
...

```

```

-----or-----
ssp # epm -ql predict
/usr/share/predict/vocalizer/visible.wav
/usr/share/predict/vocalizer/three.wav
/usr/share/predict/vocalizer/sixty.wav
/usr/share/predict/vocalizer/los.wav
...
-----

```

List package(s) that own the file `/usr/bin/predict`:

```

-----
ssp # equery belongs /usr/bin/predict
[ Searching for file(s) /usr/bin/predict in *... ]
sci-astronomy/predict-2.2.3 (/usr/bin/predict)

```

```

-----or-----
ssp # epm -qf /usr/bin/predict
predict-2.2.3
-----

```

Modify some files installed by the `predict` package and then list files owned by the `predict` package that have changed since installation:

```

-----
ssp # touch /usr/bin/predict
ssp # echo >> /usr/bin/predict-update
ssp # mv /usr/bin/kep_reload /tmp

```

```

-----
ssp # equery check predict
[ Checking sci-astronomy/predict-2.2.3 ]
!!! /usr/bin/kep_reload does not exist
!!! /usr/bin/predict-update has incorrect md5sum
!!! /usr/bin/predict has wrong mtime (is 1197864311, should be
1197864240)
* 64 out of 67 files good

```

```

-----or-----
ssp # epm -V predict
missing    /usr/bin/kep_reload
.....T    /usr/bin/predict
..5....T  /usr/bin/predict-update

```

-----

Here are some examples of using `scepm` on the SSP to query the package database in the node root file system. The output is not shown:

```
ssp # scepm -qa
ssp # scepm -ql predict
ssp # scepm -qf /usr/bin/predict
ssp # scepm -V predict
```

## Managing Packages on a Running Node

The normal operating state of the nodes in a SiCortex system does not include mounts for the portage tree and overlays. In addition, the default way to boot the nodes on an SC1458 or SC5832 uses NBD and provides a read-only root file system.

To install, remove, and search for software packages using portage, or to run emerge, you must use the `chrootfs` facility. These tasks require that you use `chrootfs`, regardless of whether your system uses NBD or NFS to serve the rootfs to the nodes.

SiCortex provides the `chrootfs` script on the nodes. For details and the procedure, see *Using chrootfs to Edit the Root File System* on page 270.

## Removing Packages

The `emerge` command can also be used to remove a package by adding the `-C` option:

```
ssp # emerge -C predict

sci-astronomy/predict
selected: 2.2.3
protected: none
omitted: none

>>> >>> 'Selected' packages are slated for removal.
>>> >>> 'Protected' and 'omitted' packages will not be
removed.

>>> >>> Waiting 5 seconds before starting...
```



```

>>> >>> (Control-C to abort)...
>>> >>> Unmerging in: 5 4 3 2 1
>>> >>> Unmerging sci-astronomy/predict-2.2.3...
No package files given... Grabbing a set.
<<<      obj /usr/share/predict/vocalizer/zero.wav
<<<      sym /usr/share/predict/vocalizer/vocalizer
<<<      obj /usr/share/predict/vocalizer/visible.wav
...
<<<      dir /usr/share/predict
--- !empty dir /usr/share/man/man1
--- !empty dir /usr/share/man
<<<      dir /usr/share/doc/predict-2.2.3
--- !empty dir /usr/share/doc
--- !empty dir /usr/share
--- !empty dir /usr/bin
--- !empty dir /usr
>>> >>> Regenerating /etc/ld.so.cache...

* GNU info directory index is up-to-date.

```

The '<<<' prefix indicates files and directories that were removed during the uninstallation process.

## Installing the n32 ABI Environment

By default, the compilers and all other tools on the system use the n64 Application Binary Interface (ABI) for native compiles. You can optionally install the n32 ABI on the nodes, in addition to n64. This allows you to compile natively on the nodes using the n32 ABI.

Alternatively, you could compile on a cross-development workstation that has been set up for the n32 ABI. See Chapter 17 - *Installing the Cross-Development Toolkit* on page 245.

The procedure described here makes the n32 ABI available on the nodes, in addition to the n64 environment. n32 does not replace n64.

The procedure below installs the n32 ABI on the SSP at:

```
/opt/sicortex/rootfs/build.n32
```

The n32 buildroot contains the headers and libraries necessary for building and running applications using the n32 ABI.

## Installing the n32 ABI Environment

The n32 libraries are visible in `/lib32` and `/usr/lib32` on the nodes but are not actually installed there. Instead, they are virtually mapped there using a mount binding.

## Installing the Buildroot for the n32 ABI

To install the n32 buildroot on the SSP:

1. Run the following command on the SSP:

```
install-n32
```

Many warnings will display on the console saying that the tar is extracting files with timestamps in the future. You can safely ignore these warnings.

2. When the installation finishes, you see the message:

```
Install complete
```

However, it is possible that the SSP will hang at this point, and you will not be able to login or perform any other actions.

If this happens, press the ON button to give the SSP a hard restart.

3. Reboot the nodes:

```
root@ssp ~ $ sboot
```

## Making the n32 ABI Available on the Nodes

As system administrator, you must make the n32 ABI available on the nodes before application developers can compile natively for n32 on those nodes.

To make the n32 buildroot (headers and libraries for n32) available on just a few nodes:

1. Install the n32 buildroot on the SSP as above.
2. Run `sboot` to reboot the nodes.
3. `ssh` to *each node* where you want the n32 ABI to be available, and run the following command:

```
/usr/sbin/mount-n32
```

To make the `n32` ABI always available on all nodes automatically, every time you boot:

1. Create the following script:

```
#!/bin/bash
# Standard boilerplate: grab boot args, preclude
# running on SSP.

if [ ! -L /var/state/boot_args ]; then
    # Don't run this on the SSP.
    exit 0
fi

/usr/sbin/mount-n32
```

2. Add the script to this directory:

```
/opt/sicortex/config/local.d
```

3. Ensure that you set the executable bit on the script that you just created.

The next time and every time you boot the nodes, `scboot` will execute the script on all nodes automatically.

For more information about adding custom scripts to `.../local.d`, see *Executing Custom Boot Scripts on the Nodes* on page 93.

## Licensing Third-Party Software

SiCortex systems ship with certain third-party software packages already installed. These pre-integrated third-party packages include:

- TotalView® Debugger
- VampirServer®

### Using TotalView

**Licensing TotalView** If you want to run the TotalView Debugger on the system, you must purchase an appropriate license from TotalView Technologies.

## Using Vampir

The VampirServer ships preinstalled on SiCortex systems. However, you must purchase a license to use the Visualization Client vng or the VNG Server.

### **Licensing VampirServer**

Contact Paratools at [vampir@paratools.com](mailto:vampir@paratools.com) or <http://www.paratools.com/> in the U.S., or at TU Dresden at <http://www.vampir.eu/> in Europe.



# Chapter 17 Installing the Cross-Development Toolkit

Applications that users intend to run on a SiCortex system must be compiled specifically for the system. There are two ways to compile an application:

- *Native* compile—Compile the application on the SiCortex system's MIPS64 processors (the nodes).
- Cross-compile—Compile the application on an x86\_64 Linux workstation that has the Cross-Development Toolkit installed on it, so that the application executable runs on the system's MIPS64 processors.

☀ For details on compiling and linking applications, see the *Programming Guide*.

This chapter documents how to install the Cross-Development Toolkit on a compliant x86\_64 Linux workstation.

In this chapter:

- Overview of the Cross-Development Environment
- Installing the Software
  - System Requirements
  - Installing on a dedicated cross-development workstation from the DVD
  - Installing on a user workstation over a site LAN
- Updating the Cross-Development Software

## Overview of the Cross-Development Environment

The Cross-Development Toolkit provides the same suite of compilers that run on the nodes in the system:

- PathScale compiler suite for Fortran 77/90/95, C and C++
- Gnu compilers for C and C++ (gcc v.4.1)

- Both compiler suites can produce either n32 or n64 ABI object code (n64 is the default).
- Binaries produced by the GNU and PathScale compilers are interoperable (as long as they are generated using the same ABI), so users can link together applications and libraries that are compiled separately, using any of the supplied compilers.

On the cross-development workstation, you call the cross-compilers and associated utilities by adding the `sc` prefix to the name of the tool; for example the `gcc` cross-compiler is `scgcc` and the linker is `scld`. The `sc` prefix instructs the cross-compiler and linker to look in the correct directories for the cross-compile headers and libraries supplied with the SiCortex cross-development software suite.

For more details, see *Programming Guide*.

## Installing the Software

Installing the cross-development toolkit involves three main steps: unpacking the main tarball, unpacking the buildroot tarball, and setting `PATH`, `PATHSCALE_BIN_DIR`, and `SYSROOT` environment variables.

The instructions provided here demonstrate how to install the software in two scenarios, in which the workstations are running SUSE and the `bash` shell:

- On a dedicated cross-development workstation from the install DVD
- On individual user workstations over the site LAN

These scenarios are not all-inclusive. They are example scenarios intended to present the basic steps for installing the software.

### System Requirements

The workstation on which you install the Cross-Development Toolkit must meet these requirements:

- An x86\_64 workstation with 1GB memory and 2Gb of free disk space
- Linux operating system

The Cross-Development Toolkit has been successfully run on SUSE Linux 10, Ubuntu 7.04, and Fedora Core 3 operating systems. We expect the toolkit to run successfully on later versions of these Linux operating systems.



## Installing on a dedicated cross-development workstation from the DVD

This procedure assumes that the System Administrator is installing the software on a workstation that presents all users easy access to the cross-development toolkit.

1. Log in as root on the workstation.
2. Put the *SiCortex Install DVD* in the workstation's DVD drive.
3. Mount the DVD device to make the install DVD accessible.

```
# mount /media/<dvd_mount> # Example: /media/dvdrecorder
```

4. Select an existing directory or create a new directory in which to install the Toolkit. Make it the current working directory.

```
# mkdir -p /path/to/install_dir
# cd /path/to/install_dir
```

5. Get the version of the tarball in the `/sicortex` directory on the installation DVD.

```
# ls /media/dvd_mount/sicortex
/media/dvd_mount/sicortex/sicortex-toolchain-<version>.tgz
```

6. Unpack the main tarball into the current working directory. Supply the version information to the tar command:

```
# tar xzf /media/dvd_mount/sicortex/sicortex-toolchain-
<version>.tgz
```

☀ Now you can access the README file, which provides up-to-date version information for the current installation.

7. Get the version of the buildroot tarball.

```
# cd sicortex-toolchain-<version>/buildroot
# ls
node-<buildrootversion>.tgz
```

8. Unpack the buildroot tarball. Supply the version information to the tar command:

```
# tar xzf node-<buildrootversion>.tgz --exclude ./dev
--exclude ./var-shared/cache
```

9. You can now delete the buildroot tarball if you need to recover disk space.

```
# rm node-<buildrootversion>.tgz.
```

10. Make the tools globally available to all users.

In `/etc/profile.local` or `/etc/csh.cshrc`, add three elements to `PATH` and set the `PATHSCALE_BIN_DIR` and `SYSROOT` variables.

For example, in `/etc/profile`, add the lines:

```
# PATH=$PATH:<install_dir>/sicortex-toolchain-<version>/  
gcc-<version>/bin  
  
# PATH=$PATH:<install_dir>/sicortex-toolchain-<version>/  
gcc-<version>/x86_64-pc-linux-gnu/  
mips64el-gentoo-linux-gnu/  
gcc-bin/<version>  
  
# PATH=$PATH:<install_dir>/sicortex-toolchain-<version>/  
perf-<date>/usr/bin  
  
# PATHSCALE_BIN_DIR=<install_dir>/sicortex-toolchain-  
<version>/pathscale-<version>/bin  
  
# SYSROOT=<install_dir>/sicortex-toolchain-<version>/  
buildroot
```

### **Installing on a user workstation over a site LAN**

This procedure assumes that the user is installing the software on his or her workstation.

Ask your System Administrator where on the site LAN the `sicortex-toolchain-<version>.tgz` is located.

1. Log in to your workstation.
2. Change to the directory in which you want to install the cross-development toolkit. If you need to, create a new directory.

```
# mkdir ./home/<install_dir>  
# cd ./home/<install_dir>
```

3. Navigate to the directory on the site LAN where the main tarball is located and get the version of the tarball.

```
.../path/to/sicortex-toolchain-<version>.tgz
```

4. Unpack the main tarball into your current working directory. Supply the version information to the `tar` command:

```
# tar xzf .../path/to/sicortex-toolchain-<version>.tgz
```

☀️ Now you can access the `README` file, which also provides complete installation instructions.

5. Get the version of the buildroot tarball.

```
# cd sicortex-toolchain-<version>/buildroot
```

```
# ls
node-<buildrootversion>.tgz
```

6. Unpack the buildroot tarball. Supply the version information to the tar command:

```
# tar xzf node-<buildrootversion>.tgz --exclude ./dev
    -exclude ./var-shared/cache
```

7. You can now delete the buildroot tarball if you need to recover disk space.

```
# rm node-<buildrootversion>.tgz.
```

8. Make the tools available in your environment each time you log in.

In your `/home/<user>/.bashrc` or `/home/<user>/.bash_profile` or `/home/<user>/.cshrc`, add three elements to `PATH` and set the `PATHSCALE_BIN_DIR` and `SYSROOT` variables.

For example, in your `/home/<user>/.bashrc`, add the lines:

```
# PATH=$PATH:<install_dir>/sicortex-toolchain-<version>/
    gcc-<version>/bin

# PATH=$PATH:<install_dir>/sicortex-toolchain-<version>/
    gcc-<version>/x86_64-pc-linux-gnu/
    mips64el-gentoo-linux-gnu/
    gcc-bin/<version>

# PATH=$PATH:<install_dir>/sicortex-toolchain-<version>/
    perf-<date>/usr/bin

# PATHSCALE_BIN_DIR=<install_dir>/sicortex-toolchain-
    <version>/pathscale-<version>/bin

# SYSROOT=<install_dir>/sicortex-toolchain-<version>/
    buildroot
```

☀ For details on cross-compiling applications, see *The SiCortex<sup>®</sup> System Programming Guide*.

## Updating the Cross-Development Software

As long as you have the required free disk space on the workstation, you can install the new version of software without first uninstalling the old version.

## Updating the Cross-Development Software

Regardless of whether you uninstall a previous version of the software, you need to edit the appropriate start up/login shell scripts to specify the new PATH, PATHSCALE\_BIN\_DIR, and SYSROOT values.

Follow the instructions in *Installing on a dedicated cross-development workstation from the DVD* on page 247 or *Installing on a user workstation over a site LAN* on page 248 to install the new software.

# Chapter 18 Shutting Down and Rebooting the System

There may be a need from time to time to shut down the entire SiCortex system. This may occur on a planned basis or as an emergency. In this chapter:

- Manual System Shutdown
- Automatic Emergency System Shutdown

## Manual System Shutdown

To shut down the entire system, perform these steps:

1. To stop a partition from accepting jobs, set its state to “drain”:

```
sysadmin@ssp # sinfo
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
sc1        up infinite 108  idle sc1-m0n[0-26],sc1-m1n[0-26],\
          sc1-m2n[0-26],sc1-m3n[0-26]

sysadmin@ssp # scontrol update NodeName=sc1-m0n[0-26],sc1-m1n[0-26],sc1-m2n[0-26],sc1-m3n[0-26]
State=drain
sysadmin@ssp # sinfo
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
sc1 up infinite 108 drain sc1-m0n[0-26],sc1-m1n[0-26],\
sc1-m2n[0-26],sc1-m3n[0-26]
```

2. To see if there are any jobs still running on the system, use the `squeue` command:

```
sysadmin@ssp # squeue
JOBID PARTITION  NAME      USER  ST TIME  NODES NODELIST(REASON)
16429      sc1 sleep-12 j_random  R 0:09      4 sc1-m1n[3-21]
```

3. To kill all jobs remaining on the system, use `scancel` like this:

```
sysadmin@ssp # scancel -p sc1
sysadmin@ssp # squeue
JOBID PARTITION  NAME      USER  ST TIME  NODES NODELIST(REASON)
sysadmin@ssp #
```

4. When you run `scboot` to reboot the nodes, `scboot` resets the nodes. However, Lustre file systems are not unmounted cleanly - including both clients and various types of servers. To fix this, run the script described below, to avoid having to wait for a period, for

the recovery process to complete. The recovery contributes about five minutes to the reboot time.

On the SSP, run the `lustre_umount.sh` script to unmount each Lustre file system, as described in *Unmounting a Lustre File System with a Script* on page 194.

5. Run `umount` commands to unmount any other file systems you may have mounted at boot time or later.
6. On the SSP, type the following command to shut down the system:

```
# shutdown -h now
```

This command powers down the SSP. However, it does not shut down the nodes. The MSPs and the nodes continue to receive power and to run at a basic level.

7. Power down to complete the shutdown:

**SC072:** To power down the MSP and nodes on an SC072, execute any required commands to gracefully shut down all applications and the desktop environment. Then press the power rocker switch to power off the system.

**SC648:** To power down the MSPs and nodes on an SC648, disconnect the main system power cord(s) from the wall.

NOTE: After you have completed all the procedures detailed above, it is safe to pull the plug to complete the power-off sequence and to power down the nodes. The power system is designed to automatically power down the hardware components in a safe and controlled manner when the power plug is disconnected from the power source.

**SC5832:** To finish powering down an SC5832, flip the main (left) breaker on the PDU at the rear of the system.

This step finishes powering down the system.

## Rebooting after Manual Shutdown

If you have shut down the entire system manually, including the SSP, provided that no emergency conditions exist that require repairs, just follow the instructions on how to boot the entire system starting with *Step 1—Prepare to Boot the System* on page 52.

If the nodes were shut down but the SSP remains up, reboot the nodes by performing the instructions starting with the `scboot` command in *Step 11—Boot the Nodes* on page 85.

## Automatic Emergency System Shutdown

When system monitoring detects any of the following conditions, the SiCortex system shuts down automatically:

- Fan tray failure
- Excessive internal temperature
- Internal voltages outside safe limits

There are three ways the system can automatically shut itself off:

1. The SSP periodically queries the MSPs. If any MSP reports dangerous voltages or temperatures, the SSP shuts off the main DC power supply to the processor modules and fan trays.
2. Each MSP (one on each processor module) monitors voltages and temperatures on its module. Dangerous voltages or temperatures (slightly higher limits than above) cause the MSP to remove power from everything on the module but the MSP itself. In other words, each MSP has the ability to shut down all the nodes on its own board.
3. There is a temperature sensor at the top of the cabinet. If a sufficiently high temperature is reached (higher still than above), it shuts off the main DC power supply to the processor modules and fan trays.

### After an Automatic Emergency Shutdown

Recovering after an automatic emergency shutdown requires additional steps prior to rebooting the SSP and the nodes.

** DO NOT ATTEMPT TO REBOOT YOUR SYSTEM.**

**If your system experiences an automatic emergency shutdown, immediately contact SiCortex Customer Support for assistance.**



## Temperature Monitoring Details

The SiCortex system contains several temperature sensors. The following table describes the default thresholds and actions for each type of sensor.

Sensor	Threshold	Action
Ice9 (Processor chip)	Temp. below 0°C	Sends a syslog message.
	Temp. above 100°C	Sends an alarm - 3 consecutive readings over 100°C sends a syslog message at level LOG_EMERG (/var/log/msp-messages-<yyyymm> )
	Temp. above 110°C	Shutdown - 3 consecutive readings over 110°C shuts down power to everything on the module except the MSP
POL (Point of Load Voltage Regulator)	Temp. below 0°C	Sends a syslog message
	Temp. above 85°C	Sends an alarm - 3 consecutive readings over 85°C sends a syslog message at level LOG_EMERG (/var/log/msp-messages-<yyyymm>)
	Temp. above 95°C	Shutdown - 3 consecutive readings over 95°C shuts down power to everything on the module except the MSP
PCB (Printed Circuit Board)	Temp. below 0°C	Sends an alarm
	Temp. above 65°C	Sends an alarm - 3 consecutive readings over 65°C sends a syslog message at level LOG_EMERG (/var/log/msp-messages-<yyyymm>)
	Temp. above 75°C	Shutdown - 3 consecutive readings over 75°C shuts down power to everything on the module except the MSP
OT thermal switch in the top of the card cage in the exhaust air stream.	Temp. above 55°C	Causes the main breaker on the SC5832 to open, shutting off all power to the system.

## Voltage Monitoring

The SiCortex system contains a number of voltage sensors. The acceptable voltage range for each sensor depends on a number of factors, including the system size, internal component being measured, and in some cases the model of the internal component.

If you have reason to believe that your system is having voltage problems, contact SiCortex Customer Support and provide as much information as possible, including console and log files related to the problem.

## Automatic Emergency System Shutdown

# Appendix A Release Profile

This appendix provides information about the software versions shipped with the current release.

## Current Release: Version 3.0

Software Package	Version Shipped with Current Release	Command to Check Version on System
Linux kernel	2.6.18	<code>uname -a</code>
Lustre	1.6.3	<p>On any node:</p> <pre>lctl lustre_build_version</pre> <p>For example, on an SC648:</p> <pre>sc1-m0n2 ~ # lctl lustre_build_version Lustre version: 1.5.97-19691231190000-PRISTINE-..opt.sicor- tex.kernel.linux.default.built-2.6.15-sc-lustre-perfmon lctl version: 1.5.97-19691231190000-PRISTINE--</pre>
SLURM	1.3.3	<code>sinfo -v</code>
gdb	6.8-r1	Use the <code>equery</code> or <code>epm</code> command to query the version of any package installed on the system. For more information, see <i>Querying Packages</i> on page 236.
binutils	2.18-r2	
gcc	4.1.2	
glibc	2.5-r4	

## Previous Release: Version 2.2

Software Package	Version Shipped with Current Release	Command to Check Version on System
Linux kernel	2.6.18	<code>uname -a</code>
Lustre	1.6.3	<p>On any node:</p> <pre>lctl lustre_build_version</pre> <p>For example, on an SC648:</p> <pre>sc1-m0n2 ~ # lctl lustre_build_version Lustre version: 1.5.97-19691231190000-PRISTINE-..opt.sicor- tex.kernel.linux.default.built-2.6.15-sc-lustre-perfmon lctl  version: 1.5.97-19691231190000-PRISTINE--</pre>
SLURM	1.2.20	<code>sinfo -v</code>

## Previous Release: Version 2.1

Software Package	Version Shipped with Current Release	Command to Check Version on System
Linux kernel	2.6.15	<code>uname -a</code>
Lustre	1.6-beta7	<p>On any node:</p> <pre>lctl lustre_build_version</pre> <p>For example, on an SC648:</p> <pre>sc1-m0n2 ~ # lctl lustre_build_version Lustre version: 1.5.97-19691231190000-PRISTINE-..opt.sicor- tex.kernel.linux.default.built-2.6.15-sc-lustre-perfmon lctl  version: 1.5.97-19691231190000-PRISTINE--</pre>

# Appendix B Licenses

This appendix provides information about licensing for products used in conjunction with SiCortex systems.

## *Myricom, Inc.*

The following license covers the binary modules provided by Myricom, Inc. SiCortex offers a 10-gigabit Ethernet PCI-Express card from Myri-net, and includes the Linux device drivers for this device in source and binary form in the SiCortex system software release.

\*\*\*\*\*

Copyright (C) 2005 - 2008 Myricom, Inc.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of Myricom, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES

(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*\*\*\*\*

# Appendix C Directories and Files

This appendix describes the important system files and where they reside on the SSP.

## Location of Files on the SSP

The root file system on the SSP contains all the software that runs on the SSP and all the software that will be booted on the nodes.

File Name	Description
/etc/sicortex-release /opt/sicortex/rootfs/default/etc/sicortex-release	Contains running SiCortex software release version. Same files exists in the node rootfs.
/opt/sicortex/bootscripts/default/	Scripts used during the preinit phase of node boot.
/etc/init.d/*	SSP boot scripts
/opt/sicortex/rootfs/default/etc/init.d/*	Node boot scripts
/etc/conf.d/*	SSP boot script configuration files
/opt/sicortex/rootfs/default/etc/conf.d/*	Node boot script configuration files
/var/db/pkg/	SSP package database
/opt/sicortex/rootfs/default/var/db/pkg/	Node package database
/etc/make.conf	Custom portage configuration for the SSP
/opt/sicortex/rootfs/default/etc/make.conf	Custom portage configuration for the nodes
/var/lib/misc/dnsmasq.leases	Active DHCP lease file for dnsmasq running on SSP.
/var/log/mspstate/	Log of MSP state changes used by sbboot to determine if MSPs are running correct software versions.
/usr/portage/	Portage tree. Contains standard Gentoo ebuild definitions files.
/opt/sicortex/overlays/	Symlink to current portage overlay directories. Contains SiCortex created or modified ebuild definitions.
/opt/sicortex/profile/	Symlink to the current portage profile. This profile location contains SiCortex portage configuration for both the SSP and nodes. The symlink is relative to /opt/sicortex/overlays.
/boot/	Contains SSP kernels. Not overwritten during software update. Partition 1 on the SSP disk.

## Location of Files on the SSP

File Name	Description
/boot/grub/grub.conf	Grub boot menu options. Each software release installed on the SSP is represented by three netmask options in this file. You can have up to three software release installed simultaneously to partitions 5, 6 and 7 on the SSP disk. The software release version on each partition is record in this file as part of the boot option text.
/var/state/	Contains persistent system/hardware configuration state. Not overwritten during a software update. Files here are either generated automatically, or they are the target of symlinks from /etc and should be edited from /etc. Symlink to /boot/state. For details on how these files are generated and handled, see <i>Status of /var/state Files Across Releases</i> on page 263.
/var/state/etc/	The target of symlinks from /etc for configuration files that should persist across software updates.
/var/log/	Contains SSP and node logs. Not overwritten during software update. Partition 8 on the SSP disk.
/lclhome/	NFS exported directory for shared scratch space between SSP and node. On a blizzard if you mount this location on all nodes you should be careful to use some mechanism to avoid overloading the SSP when mounting and using this area.
/tftpboot/	TFTP exported area used to transfer files to the MSPs. For example, during boot the boot loader programs, node kernel and initramfs are copied here before being loaded onto the nodes via the MSPs.



## Status of /var/state Files Across Releases

This section lists the main files in /var/state, and describes whether they persist across the installation of a new release, and are therefore shared across software releases, and the source of those files.

### Shared across releases, and user editable:

```
/var/state/etc/sicortex/policyd.conf
/var/state/etc/sicortex/watchdogd.conf
/var/state/etc/sicortex-system.conf
```

### Shared across releases, created by install:

```
/var/state/scprofile
```

### Shared across releases, created by system daemons / sbboot:

```
/var/state/hosts
/var/state/slurm.conf
/var/state/dnsmasq.conf
/var/state/conserver_mode
/var/state/config/sci/main-config.sh
/var/state/dnsmasq.conf.dnsmasq
/var/state/dimmtypes.sci
/var/state/hosts.fan_right
/var/state/hosts.fan_left
/var/state/hosts.fan_front
/var/state/hosts.fan_back
/var/state/hosts.power
/var/state/route_info.sci
```

# Finding Nodes

## Node Addressing

Each node can be specified and addressed individually. The following table describes the SCethernet addressing scheme for the nodes. These are the addresses used to specify the nodes in all SLURM commands

System	Node Address Format	Meaning
SC5832	<b>scx-mjn<math>\zeta</math></b>	Where: <b>scx</b> indicates the SC5832 <b>m</b> stands for module $y$ is an integer from 0 to 35 <b>n</b> stands for node $\zeta$ is an integer from 0 to 26  Range: <b>scx m0n0</b> to <b>scx-m35n26</b>
SC1458	<b>sci-mjn<math>\zeta</math></b>	Where: <b>sci</b> indicates the SC1458 <b>m</b> stands for module $y$ is an integer from 0 to 8 <b>n</b> stands for node $\zeta$ is an integer from 0 to 26  Range: <b>scx m0n0</b> to <b>scx-m8n26</b>

System	Node Address Format	Meaning
SC648	<b>sc1-mynz</b>	Where: <b>sc1</b> indicates the SC648 mounted in the right side of the cabinet <b>m</b> stands for module <i>y</i> is an integer from 0 to 3 <b>n</b> stands for node <i>z</i> is an integer from 0 to 26  Range example: <b>sc1-m0n0</b> to <b>sc1-m3n26</b>
SC072	<b>sca-mynz</b>	Where: <b>m</b> stands for module <i>y</i> is 0 <b>n</b> stands for node <i>z</i> is an integer from 0 to 11  Range: <b>sca-m0n0</b> to <b>sca-m0n11</b>

## MSPnet Addressing

MSPnet uses an alternate method of addressing the nodes. Under this scheme, nodes are numbered sequentially starting with m0n0 as 0.

### Calculating sequential node number

In `mfd.log` and several other low-level contexts, *node-number* is an enumerated value:

On an SC072, *node-number* is [ 0 - 11 ], where:

m0n0 is numbered 0, up to m0n11, which is numbered 11

On an SC648, *node-number* is [ 0 - 107 ], where:

m0n0 is numbered 0, up to m3n26, which is numbered 107

On an SC1458, *node-number* is [ 0 - 242 ], where:

m0n0 is numbered 0, up to m8n26, which is numbered 242

On an SC5832, *node-number* is [ 0 - 971 ], where:

m0n0 is numbered 0, up to m35n26 which is numbered 971

To translate this enumerated node-number to the host name, use modular arithmetic. Each module in a SiCortex system has 27 nodes.

So for example, to find the host name for node-number 137, type:

```
# echo sc1-m$(( nodenum / 27 ))n$(( 137 % 27 ))
```

Which will yield:

```
sc1-m5n2
```

Adjust the prefix (in this example "sc1-") to match the node number prefix of the system in question.

Note that the conversion equation show above above can be used on all SiCortex systems that use the 27-node CPU module, including the SC5832, SC1458, and SC648.

### **Sequential node numbering details**

The details for sequential node numbering are as follows:

**SC648** On an SC648, sequential node-number is [ 0 - 107 ], where:

m0n0 is numbered 0

m0n26 is numbered 26

m1n0 is numbered 27 (26 + 1)

m1n26 is numbered 53 (27 + 26)

m2n0 is numbered 54 (53 + 1)

and so forth, up to m3n26, which is numbered 107

**SC5832** On an SC5832, sequential node-number is [ 0 - 971 ], where:

Same numbering as above for modules 0 - 3.

For modules 4 - 35:

m4n0 is numbered 108 (107 + 1)

m4n26 is numbered 134 (108 + 26)

m5n0 is numbered 135 (134 + 1)

m5n26 is numbered 161 (135 + 26)

m6n0 is numbered 162 (161 + 1)

m6n26 is numbered 188 (162 + 26)

for each module, add 27, such that

m35n26 is numbered 971

## I/O-Ready Nodes

Specific nodes on each module also have the hardware to be connected and used as the head node, router nodes, or storage I/O nodes. Of the 27 nodes on each processor module, only four nodes can be used for IO:

- One node has an on-board dual gigabit Ethernet port:
  - Node 6 - Hard-wired (Node number sc\*-m\*n6)
- Three nodes are connected to PCI Express® ports:

PCIExpress Port Number	Port Position on Module Front Panel	Connected to Node Number
Port 1	Top	sc*-m*n1
Port 2	Middle	sc*-m*n3
Port 3	Bottom	sc*-m*n0



# Appendix D Common Procedures

This appendix describes procedures you may need to use in the course of administering a SiCortex system.

## Logging into the System

### Logging into the SSP as root

To log in to the SSP as `root`:

1. Press RETURN on the keyboard until you get a LOGIN prompt on the monitor.
2. At the prompt, type `root` and the default root password:

```
LOGIN: root  
PASSWORD: sicortex
```

### Logging into the SSP

As shipped, the system is set up so that only `root` users are allowed to log into the SSP. The intention is that only the system administrator should be logging into the SSP. This restriction protects the SSP, which is the management appliance, from unauthorized access or overuse by application users of the system.

To allow users other than `root` to ssh to the SSP, perform these steps:

1. As `root` on the SSP, edit the file `/etc/ssh/sshd_config`.
2. Locate the line:

```
AllowUsers root
```

3. After `root`, add the names of the additional users who shall be allowed to ssh to the SSP. For example:

```
AllowUsers root nigel graciela yu-ching david
```

4. To allow all users to ssh to the SSP, comment out the line like this:

```
# AllowUsers root
```

5. Save the file.

## Logging into the Head Node as a Non-root User

After user accounts have been set up on the system, any user with an account can log into the head node.

Note that the head node should not be included in any of the compute partitions, and users should be discouraged from logging into nodes in compute partitions.

If a logical name has been defined for the head node on your LAN, you can log into the head node from anywhere on the LAN with a network connection to the system nodes, using this command:

```
$ ssh <logical-head-node-name>
```

From either the SSP or another node on the same system, you can use the internal node address to log into the head node:

```
$ ssh <head-node-address>
```

where `head-node-address` is:

```
<partition-name>-m<module-number>n<node-number>
```

For example:

```
$ ssh sc1-m0n6
```

or

```
$ ssh sc1-m3n26
```

## Using chrootfs to Edit the Root File System

There are several system administration tasks that involve modifying the root file system for the nodes. You need the `chrootfs` command for these tasks, because on a machine that uses NBD to serve the rootfs to the nodes, the rootfs is normally read-only.



**chrootfs command** The `chrootfs` command gives you read/write access to the root file system in a writeable NFS-mounted `chroot` area on a node.

When you exit from the `chrootfs` environment, the mounts are cleaned up. To make the changes live, you have to reboot the nodes.

### Tasks requiring chrootfs

You must use `chrootfs` for the following tasks:

- Editing any files that reside in the root file system on an NBD machine ( SC1458, SC5832), such as configuration files or executables. On an NBD machine, the rootfs is normally read-only.

On the SC1458 and the SC5832, by default the root file system is booted with NBD (instead of NFS). NBD provides a read-only root file system on the nodes after you boot.

For systems running a read-only root file system, you must first establish write access before you can edit files there.

- Running an `emerge` command natively (on the nodes) to install a piece of software, on both NBD and NFS machines. `chrootfs` NFS-mounts directories from the SSP that are required to use the `emerge` command.
- Editing the root passwords on the SSP and the nodes.

The root file-system for the nodes is stored on the SSP in this directory:

```
/opt/sicortex/rootfs/default/
```

The rest of this section explains how to use `chrootfs` to get write access to the root file system or to run `emerge` on the nodes.


### 2 copies of rootfs on SSP

There are two copies of the root file system on the SSP:

- Rootfs for the SSP—The rootfs for the SSP is mounted at `/` on the SSP, namely the top-level directory.
- Rootfs for the nodes—The rootfs for the nodes resides on the SSP in the directory `/opt/sicortex/rootfs/default`. When you run `scboot` to boot the nodes, `scboot` takes this root file system and installs it at the top level at `/` on the nodes.

## Using chrootfs

Using `chrootfs` is a three-step process. You set up the `chroot` environment, do your task, then exit from the `chroot` environment.

-  Before you begin, make a copy of the SSP's original version of the node root file system.

`/opt/sicortex/rootfs/default`

Copy this original rootfs, and store the copy in a safe location.

1. Identify a node that does *not* serve the root file system (NFS or NBD) to use for this procedure.

Do not use these nodes:

Model	Default Rootfs Server Node(s)
SC648	m0n6
SC1458	m0n6
SC5832	m0n6, m2n6, m4n6, m6n6

2. Log in as `root` on your chosen node.
3. At the command prompt, type:  

```
root# chrootfs
```
4. The `chrootfs` command creates all the mounts you need to edit the node root file system in read/write mode. Then `chrootfs` puts you into an environment where you have read/write access, so that you can edit the rootfs in `/opt/sicortex/rootfs/default`.
5. Perform your task, which might be:
  - Editing the root file system.
  - Running an `emerge` command on the nodes.
  - Editing the password on the SSP and the nodes.
6. When you are done with your task, exit from the `chrootfs` environment by typing:

```
chrootfs# exit
```

Exiting from `chrootfs` cleans up all NFS mounts that pointed to the SSP's copy of the node root file system.

7. Reboot the nodes. Rebooting propagates the newly edited version of the node root file system from the SSP to the nodes that serve the root file system, and from there to the rest of the nodes.

```
root# sboot -p scX
```


## Using Portage on a Running Node (chrootfs)

The normal operating state of the nodes in a SiCortex system does not include mounts for the `portage` tree and overlays. In addition, the default way to boot the nodes on an SC648, SC1458, or SC5832 uses NBD which provides a read-only root file system.

SiCortex provides the `chrootfs` script on the nodes to allow you to install, remove, and search for software packages using `portage`. This script runs on a single node to set up a special `chroot` area. The `chroot` area has all the read-write mounts from the SSP necessary to edit the root file system or run `portage` commands.

After setting up all the mounts, the `chrootfs` command starts a `chroot'd` shell in this `chroot` area. The shell prompt is changed so that "[chrootfs]" becomes the prefix. When you exit the shell, `chrootfs` removes all the read/write mounts and closes the `chroot` shell.

By default, the `chrootfs` command uses `/opt/sicortex/rootfs/default` from the SSP as the root file system for the `chroot` area. However, `chrootfs` accepts an optional parameter to specify a different rootfs location from the SSP. This is useful for modifying a copy of the node root file system to test before making it the default.

 You should not run the `chrootfs` command simultaneously on all nodes of a SiCortex system cluster (i.e. with `srun`) because this will overload the SSP.

## chrootfs Command

### Description

Creates mounts on the current node for a read/write node root file system tree and chroots into it.

Log in as `root` to a node that is not an NBD root file system server node. At the prompt, type:

```
root# chrootfs
```

You enter an environment where you have read/write access to the root file system.

### Exiting the chrootfs environment

To exit, at the prompt type:

```
chrootfs# exit
```

When the shell exits, it unmounts the mounts in the chroot area.

### Usage

```
chrootfs [-v] [rootfs-path]
```

### Options

-v	Verbose output
<i>rootfs-path</i>	Path to NFS mount from the SSP Default: /opt/sicortex/rootfs/default By default, the chrootfs command provides read/write access to the original root file system on the SSP, in the above location. You can use this option to specify a different copy of the root file system as the target for the chrootfs command.

### Related Tools

The normal version of a command like **emerge** points at the root file system on the SSP. The **sc** prefix modifies the command to point at the node version of the same entity.

This command	Points at this entity
<b>emerge</b>	/ If you run <b>emerge</b> on the SSP, it installs software for the SSP. If you run it on a node using chrootfs, then it installs software for the nodes.
<b>scemerge</b>	The root file system for the nodes instead, ie. /opt/sicortex/rootfs/default. Runs on the SSP but installs software for the nodes.

This command	Points at this entity
<code>gcc</code>	<p>Accesses the C/C++ compiler on a cross-compile workstation.</p> <p>Native <code>gcc</code> on the SSP generates code for the SSP.</p> <p>Native <code>gcc</code> on the nodes generates code for the nodes.</p>
<code>scgcc</code>	<p>Accesses the <code>gcc</code> compiler on the nodes. Runs on the SSP, but generates code that runs on the nodes.</p>
<code>epm</code>	<p>Used to query packages on the SSP.</p>
<code>scepm</code>	<p>Used to query packages in the node rootfs.</p> <p>For example:</p> <p style="padding-left: 40px;"><code>scepm -qa</code></p> <p>Lists all the software packages installed in the node root file system (from the SSP).</p>



# Appendix E Customized System Files— Checklist

This appendix lists the system files that you are likely to edit in configuring your system. Many of these files are described in procedures in this guide.

## Backing up Your Modified System Files

Backing up your system files on the SSP offers moderate security. The safest location on the SSP to back up your modified system files is in the `/var/log/` directory. The `/var/log/` directory is preserved when you do a normal install of a new version of software into the next available disk partition.

Backing up your system files outside the system offers the greatest security. The `install-ssp` command's `--repartition` option reformats the entire SSP disk. You would only use that option in the unlikely event that your SSP has become corrupted. Nevertheless, to ensure that your backup copies of modified system files will be available regardless of how you need to install the next version, choose a secure location to back the files up:

- On another system (not on the SSP).
- On a CD or DVD stored in a secure location.



Use the right column to record the location where you back up your modified copy of the file. Use the backup copy to restore your custom settings, each time you install a new software release.

Filename	Purpose	Location of My Backup Version
<code>/opt/sicortex/rootfs/default</code>	Root file system or <i>rfs</i> (below)	
<code>build/build.&lt;date&gt;.tgz</code>	Optional backup of root file system	
<code>/etc/localtime</code>	Time zone on SSP or nodes	
<code>/etc/conf.d/clock</code>	Add <code>CLOCK="local"</code>	

<b>Filename</b>	<b>Purpose</b>	<b>Location of My Backup Version</b>
/rfs/etc/localtime	Sets nodes time zone	
/var/state/msp/etc/TZ	Sets time zone for MSPs, on SSP	
/etc/conf.d/net	Network configuration file for SSP.	
/etc/conf.d/net.example	Sample file showing multiple options for configuring networking for SSP	
sicortex-system.conf	System configuration parameters file. Your edited version of this file controls many of the key configuration parameters on your system.	
sicortex-system.conf.example	Sample version of above file, showing options for configuring your system.	
sicortex-install.conf	System installation parameters file.	
/etc/sicortex-install.conf.example	Sample version of above file, that documents all the options for this file.	
/var/state/route_info.<partition>	File automatically generated by scboot every time you boot. Specifies any nodes, modules, or links that are disabled and should be excluded from the boot, and any placeholder modules the system contains.	N/A. Do not back up this file. Scboot generates the correct version every time you boot the nodes.
/tmp/my-first-scboot-session.txt	Location where you may want to use script to record all console messages before you issue the scboot command.	
/etc/passwd /etc/shadow /etc/group	Define root and user passwords for the SSP.	
/rfs/etc/passwd /rfs/etc/shadow /rfs/etc/group	Define the root and user passwords for the nodes.	
/root/.ssh/*	The SSP root user ssh configuration.	



<b>Filename</b>	<b>Purpose</b>	<b>Location of My Backup Version</b>
<i>/rfs/root/.ssh/*</i>	The node root user ssh configuration.	
<i>/etc/openldap/slapd.conf</i>	Holds the LDAP root password on the SSP.	
<i>/etc/openldap/ldap.conf</i>	LDAP configuration file.	
<i>/etc/pam.d/system-auth</i>	LDAP configuration file.	
<i>/etc/ldap.conf</i>	LDAP configuration file.	
<i>/etc/nsswitch.conf</i>	LDAP configuration file.	
<i>/etc/fstab</i>	SSP fstab configuration.	
<i>/rfs/etc/fstab</i>	Nodes fstab configuration.	
<i>/etc/ntp.conf</i>	NTP configuration	
<i>/opt/sicortex/config/local.d</i>	Directory for SSP and node custom startup scripts	
<i>/etc/make.conf</i>	SSP portage configuration.	
<i>rfs/etc/make.conf</i>	Node native portage configuration.	
<i>rfs/etc/config_root/etc/make.conf</i>	Node cross-compile portage configuration.	
<i>rfs/etc/sysctl.conf</i>	Node kernel settings.	



# Appendix F sicortex-system.conf.example

This appendix shows the complete contents of the following file as of the current release of this document:


```
/etc/sicortex-system.conf.example
```


This is a symlink that actually points to:

```
/var/state/etc/sicortex-system.conf.example
```

## Applies only to the SC5832, SC1458, and SC648

The file shown is shown below as it appears on the SC5832, SC1458, and SC648 in the current version.

 **The file below is *not the same* as the version used on the SC072.**

 **For all configuration procedures and details for the SC072 Catapult, see the *SC072 Catapult System User's Guide*.**

```
#
# This is the main configuration file for the SiCortex system.
#   - Contains options specific to the SiCortex software release
#   - For options that begin with a partition, you must sbboot the
#     partition for changes to take affect.
#   - /etc/sicortex-system.conf should be a symlink to
#     /var/state/etc/sicortex-system.conf. This allows the system
#     configuration to be persistent across different software release
#     installs on the same SSP.

# If you uncomment any of the example lines in this file,
# check the system prefix and change it to the correct one for your system,
# as necessary:
#
# sca = SC072    (Catapult)
# scl = SC648    (snow)
# sci = SC1458   (hail)
# scx = SC5832   (blizzard)

# This header is necessary for the config parser module. Do not alter.
[DEFAULT]

#
# Warning! When editing this file, use comments only at the beginning
# of a line! Trying to do something like
#
# keyword = value    # comment
#
```

```

# will not do what you want!
#

#-----
# Basic cluster and boot parameters
#
# The way we serve the rootfs to the nodes.
#
# Default mode for serving rootfs on Catapult
# sca.boot.rootfs-mode = nfs
#
# Default mode for serving rootfs on SC648
# sc1.boot.rootfs-mode = nfs
#
# Default mode for serving rootfs on SC1458
# sci.boot.rootfs-mode = nfsnbd
#
# Default mode for serving rootfs on SC5832
# scx.boot.rootfs-mode = nfsnbd

#
# The node kernel's log level to use while booting
#
# sc1.boot.log-level    = 8

#
# Additional arguments to append to the node kernel command line
#
# sc1.boot.append-kargs = initcall_debug

# -----
# -----
# Configuration for specialized nodes, ie head nodes etc

#
# Which node is the head node, ie has external ethernet interfaces, users are
# expected to log in to it etc.  If not set, there is no designated head node.
#
# IMPORTANT!  When specifying a head node, you must also configure their
# network interfaces, in the network configuration section, below.

# sc1.cluster.head-node = sc1-m3n6

#
# Which nodes are router nodes.  The key name is an historical artifact; this
# list should include only *network* I/O nodes, not storage I/O nodes.
#
# IMPORTANT!  When specifying IO node(s), you must also configure
# their network interfaces, in the network configuration section, below.

# sc1.cluster.io-nodes = sc1-m2n1, sc1-m3n1

#-----
# Network configuration

# Specify the default gateway for the cluster
# If "ssp" is specified, the SSP is used as the default gateway.  Otherwise,
# traffic from internal nodes is distributed across the router nodes.
#
# sc1.cluster.default-router = gw.example.com

# Specify additional routes (optional).  This is a space-separated list of CIDR
# network numbers (a.b.c.d/e).  Routes to each of these will be added through

```

```

# the per-interface gateway addresses (below) - not the default router address
# (above) - and traffic from internal nodes will be distributed across the
# router nodes.
# sc1.cluster.external-network = 10.0.5.0/24 10.6.0.0/16

#
# Network configuration for exterior nodes
#
# sc1.node.sc1-m3n6.interfaces      = eth1
# sc1.node.sc1-m2n1.interfaces     = eth1
# sc1.node.sc1-m3n1.interfaces     = eth1

# For each router node, what kind of routing should it do? Legal values are:
# yes - simple routing, without NAT
# nat - routing with NAT
# no - do not route
# Specifying different values for the router nodes, or specifying "no" for
# any, will probably result in a system that cannot route properly or
# consistently and is strongly discouraged.
# sc1.node.sc1-m2n1.router         = nat
# sc1.node.sc1-m3n1.router         = nat

# If you are using static addresses, you need to specify
# address/netmask/gateway for each interface on each node.
# The per-interface gateway address is *not* used for
# default routes, but *is* used for external-network routes.
# sc1.node.sc1-m3n6.eth1.address   = 10.4.2.27
# sc1.node.sc1-m3n6.eth1.netmask   = 255.255.0.0
# sc1.node.sc1-m3n6.eth1.gateway   = 10.4.0.1
# sc1.node.sc1-m2n1.eth1.address   = 10.4.2.28
# sc1.node.sc1-m2n1.eth1.netmask   = 255.255.0.0
# sc1.node.sc1-m2n1.eth1.gateway   = 10.4.0.1
# sc1.node.sc1-m3n1.eth1.address   = 10.4.2.29
# sc1.node.sc1-m3n1.eth1.netmask   = 255.255.0.0
# sc1.node.sc1-m3n1.eth1.gateway   = 10.4.0.1

# If you are using dhcp, you need to specify it once per interface on each
# node
# sc1.node.sc1-m0n1.eth1.address   = dhcp
# sc1.node.sc1-m2n1.eth1.address   = dhcp
# sc1.node.sc1-m3n1.eth1.address   = dhcp

# -----
# -----
# Placeholder module configuration
#
# Specify system slots that contain placeholder modules
# sc1.cluster.placeholder-modules = 1, 3
#
# Specify links disabled by the placeholder configuration
# sc1.cluster.placeholder-disabled-links = m0n3-rx2

# -----
# for 12 slot blizzard
# scx.cluster.placeholder-modules =
0,2,3,4,5,8,9,12,13,14,15,16,17,18,19,21,23,25,26,29,31,32,33,35
# scx.cluster.placeholder-disabled-links = m1n20-rx1,m10n18-rx0,m28n14-rx2,m22n13-rx0,m1n24-
rx2,m10n19-rx0,m1n13-rx0,m1n26-rx1,m10n15-rx0,m1n23-rx0,m30n18-rx0,m27n18-rx0,m20n17-
rx0,m27n15-rx0,m24n19-rx0,m1n23-rx2,m27n16-rx0,m1n23-rx2,m6n24-rx0,m1n22-rx0,m22n23-rx0,m6n22-
rx0,m24n13-rx0,m6n15-rx0,m10n22-rx0,m27n19-rx0,m34n17-rx0,m24n24-rx0,m10n16-rx0,m11n15-
rx0,m11n16-rx0,m34n23-rx0,m24n18-rx0,m34n24-rx0,m24n22-rx0,m6n14-rx2,m1n24-rx0,m22n20-
rx1,m1n21-rx2,m7n23-rx2,m28n22-rx0,m20n21-rx1,m34n22-rx0,m6n16-rx0,m30n19-rx0,m1n18-rx0,m7n13-
rx0,m11n18-rx0,m11n13-rx0,m28n17-rx0

# -----

```

```

# for 20 slot blizzard
# scx.cluster.placeholder-modules = 2,3,4,8,9,16,18,19,21,22,24,25,26,28,29,34
# scx.cluster.placeholder-disabled-links = m0n18-rx0,m12n6-rx0,m20n17-rx0,m5n25-rx0,m27n20-
rx0,m0n4-rx0,m0n3-rx0,m30n20-rx0,m5n13-rx2,m11n26-rx1,m17n13-rx2,m13n18-rx0,m13n19-rx0,m17n2-
rx0,m0n19-rx0,m31n16-rx0,m31n15-rx0,m31n22-rx0,m0n24-rx0,m32n3-rx0,m15n20-rx1,m13n3-
rx0,m20n21-rx1,m35n13-rx0,m33n22-rx0,m14n6-rx0,m5n24-rx0,m13n4-rx0,m12n17-rx0,m17n14-
rx0,m35n18-rx0,m23n22-rx0,m33n20-rx0,m32n24-rx1

# -----
# for 28 slot blizzard
# scx.cluster.placeholder-modules = 0,3,9,13,14,19,26,31
# scx.cluster.placeholder-disabled-links = m18n17-rx0,m5n25-rx0,m11n26-rx1,m30n20-rx0,m11n26-
rx1,m11n17-rx1,m11n17-rx1

# -----
# for 3 slot hail
# sci.cluster.placeholder-modules = 0,2,3,5,7,8
# no disabled links in this configuration

# -----
# for 6 slot hail
# sci.cluster.placeholder-modules = 1,4,7
# no disabled links in this configuration

# -----
# Other cluster-wide configuration

# If some nodes are broken or misbehaving, they can be masked out of
# the system configuration by adding them to this list.
#
# Disabled nodes, modules and links
# sc1.cluster.disabled-nodes = m0n21, m2n23, m3n2
# sc1.cluster.disabled-modules = 1,2,3
# sc1.cluster.disabled-links = m1n20-rx0, m3n12-tx2

# -----

```