

# **Scali System Guide**

**Copyright © 1999, 2000 Scali AS. All rights reserved.**

# Table of contents

---

<b>Chapter 1 Introduction.....</b>	<b>9</b>
1.1 Purpose of the Scali System Guide.....	9
1.2 Overview of the Scali System Guide.....	9
1.3 How to read this guide .....	10
1.4 Acronyms and abbreviations .....	10
1.5 Terms and conventions .....	11
1.6 Typographic conventions.....	11
<b>Chapter 2 Scali system overview.....</b>	<b>13</b>
2.1 Description of a Scali system .....	13
2.2 Scali system example .....	15
<b>Chapter 3 Hardware installation .....</b>	<b>17</b>
3.1 Hardware organization .....	17
3.2 Node BIOS preparation.....	18
3.2.1 MP level .....	18
3.2.2 Interrupt level mapping.....	18
3.2.3 Power on startup .....	18
3.3 Dolphin SCI interconnect installation.....	18
3.3.1 SCI adapter installation .....	19
3.3.2 SCI Cabling .....	20
3.3.2.1 Single SCI ring cabling.....	21
3.3.2.2 Two-dimensional SCI torus cabling.....	22
3.4 Console management .....	23
3.5 Power switching.....	23
<b>Chapter 4 Software installation .....</b>	<b>25</b>
4.1 Software installation overview .....	25
4.2 Operating system installation .....	25
4.2.1 OS prerequisites.....	25
4.2.2 Linux specific setup.....	27
4.2.2.1 NIS setup on RedHat.....	27
4.2.2.2 NIS setup on SuSE.....	28
4.2.2.3 The bigphysarea kernel patch (upgrade only).....	28
4.2.3 Solaris specific setup.....	28
4.2.3.1 NIS setup on Solaris.....	28
4.3 Scali software installation.....	29

4.3.1 Install session example .....	30
4.3.2 Uninstall session example .....	37
<b>Chapter 5 The Scali Desktop GUI .....</b>	<b>39</b>
5.1 Overview .....	39
5.2 Getting started with the Scali Desktop .....	39
5.2.1 Configuration of the Scali Desktop .....	41
5.2.2 Logging in to a system .....	43
5.3 Using the Scali Desktop .....	44
5.3.1 Executing programs - Run menu.....	48
5.3.1.1 Executing MPI programs.....	48
5.3.1.2 Running Parallel Shell commands.....	51
5.3.2 Power and Console control - Management menu .....	52
5.3.3 High speed Interconnect configuration - Interconnect menu .....	53
5.3.4 System monitoring - Status menu.....	55
5.3.5 Software installation - Software menu .....	57
<b>Chapter 6 ScaSH - parallel shell tools .....</b>	<b>61</b>
6.1 scash - the parallel shell command.....	61
6.2 scahosts - nodes availability check .....	62
6.3 scarp - local file copy.....	62
6.4 scarcp - remote file copy .....	63
6.5 scaps - process information .....	64
6.6 scakill - parallel kill command.....	64
6.7 scarup - node status.....	65
6.8 scawho - user information .....	66
6.9 ScaSH configuration file.....	66
<b>Chapter 7 Scali system configuration .....</b>	<b>69</b>
7.1 The configuration server .....	69
7.2 The ScaConf daemons .....	70
7.3 ScaConf - command line interface .....	70
7.3.1 Starting scaconftool.....	70
7.3.2 Using scaconftool.....	71
7.3.2.1 scaconftool - node selection.....	71
7.3.2.2 scaconftool - verbose mode.....	72
7.3.2.3 scaconftool - truncation.....	72
7.3.2.4 ScaConf user modes. ....	72
7.3.2.5 scaconftool - batch mode .....	73
7.3.3 scaconftool - status check of nodes and daemons .....	73
7.3.4 The fix command .....	74
7.3.5 Setting SCI nodeID .....	75

7.3.6	Reloading SCI-driver .....	76
7.3.7	Powerswitching with scaconftool.....	76
7.3.8	Access to console on one node .....	76
7.3.9	On-line restart of the configuration server .....	76
7.3.10	Failing nodes .....	77
7.3.11	Getting updated information.....	77
7.3.12	Daemon control with scaconftool.....	77
7.3.13	Server options.....	77
7.4	Routing.....	78
7.4.1	Ring topology .....	78
7.4.2	Torus Topology .....	78
7.4.3	Dimensional routing .....	79
7.4.4	Maxcy routing algorithm .....	79
7.4.5	Routing partitions .....	80
7.4.6	Testing the routing.....	80
7.5	Installation and package dependencies.....	81
7.6	saconftool - command reference .....	81
7.6.1	ConfTool> console.....	81
7.6.2	ConfTool> daemon.....	82
7.6.3	ConfTool> fix .....	82
7.6.4	ConfTool> getopt .....	82
7.6.5	ConfTool> setopt.....	82
7.6.6	ConfTool> select .....	83
7.6.7	ConfTool> unselect.....	84
7.6.8	ConfTool> list .....	84
7.6.9	ConfTool> fail .....	85
7.6.10	ConfTool> reconnect.....	85
7.6.11	ConfTool> log.....	86
7.6.12	ConfTool> nodeid.....	86
7.6.13	ConfTool> power.....	86
7.6.14	ConfTool> reload .....	86
7.6.15	ConfTool> reroute .....	87
7.6.16	ConfTool> status .....	87
7.6.17	ConfTool> link.....	88
7.6.18	ConfTool> update .....	88
7.6.19	ConfTool> restart .....	88
7.6.20	ConfTool> sciping.....	88
7.6.21	ConfTool> help .....	88
7.6.22	ConfTool> quit.....	89

<b>Chapter 8 Scali System Monitoring</b> .....	<b>91</b>
8.1 Overview .....	91
8.2 The monitoring daemon scamond.....	91
8.2.1 Configuration file .....	91
8.2.2 Manual start/stop.....	91
8.3 The SNMP daemon scasnmpd .....	92
8.3.1 Manual start/stop.....	92
<b>Chapter 9 Troubleshooting</b> .....	<b>93</b>
9.1 Hardware installation problems.....	93
9.2 Software installation problems.....	93
9.3 License problems .....	94
9.4 MPI problems.....	94
9.5 SCI Interconnect problems .....	94
9.5.1 SCI error messages .....	94
9.5.2 SCI Troubleshooting .....	96
9.6 Technical Support.....	97
9.6.1 Support Contracts .....	97
9.6.2 How to report the problem.....	97
9.6.3 Other Feedback .....	98
9.6.4 Keeping informed .....	98
<b>Appendix A Scali software details</b> .....	<b>99</b>
A-1 General package information.....	99
A-1.1 Scali software platform CD-ROM.....	99
A-1.1.1 Install.....	99
A-1.1.2 Uninstall.....	100
A-1.2 Scali software directory structure .....	100
A-2 Scali package file naming convention.....	100
A-3 Scali software platform contents .....	101
<b>Appendix B ScaPkg - Scali Software installation program</b> .....	<b>103</b>
B-1 Using scapkg.....	103
B-2 Configuration .....	104
B-2.1 ScaPkg.conf - path to package files (repository) .....	104
B-2.2 ScaPkg.conf - package list.....	105
B-2.3 ScaPkg.conf - dependency list.....	105
B-2.4 ScaPkg.conf - nodelist .....	106
B-2.5 Package response files.....	106

<b>Appendix C Scali Software Licensing</b> .....	<b>107</b>
C-1 Introduction .....	107
C-2 Licensing Software .....	107
C-3 The license file .....	107
C-3.1 Default location .....	107
C-3.2 SCALM_LICENSE_FILE environment variable.....	108
C-3.3 Installation (distribution) .....	108
C-3.4 License file example .....	108
C-3.5 Adding or updating features (licenses).....	109
C-4 Requesting licenses.....	109
C-4.1 Demo licenses .....	109
C-4.2 Permanent licenses .....	109
C-5 Troubleshooting .....	110
C-5.1 Error: Invalid FEATURE line .....	110
C-5.2 Error: Invalid version x.x > y.y.....	110
C-5.3 Error: FEATURE has expired .....	110
C-5.4 Error: Host ID is not found! .....	110
<b>Appendix D SCI Utility Programs</b> .....	<b>111</b>
D-1 SCI hardware status programs.....	111
D-1.1 sciping .....	111
D-1.2 scimonitor .....	112
D-1.3 sciemsg.....	112
D-1.4 scidbx .....	112
D-1.5 scideb .....	112
D-1.6 scinode.....	113
D-1.7 scinfo .....	113
D-1.8 scireconf .....	114
D-1.9 scireload .....	114
D-1.10 scidle .....	115
D-1.11 scicards .....	115
<b>Appendix E Scali System Cabinet Assembly Guide</b> .....	<b>117</b>
E-1 Introduction .....	117
E-2 Cabinet assembly.....	117
E-3 Insertion of the processing nodes.....	123
E-4 Interconnection of nodes.....	126
<b>Appendix F Related Documentation</b> .....	<b>129</b>
F-1 References .....	129

Section:



# Chapter 1

# Introduction

---

## 1.1 Purpose of the Scali System Guide

The intention of the Scali System Guide is:

- to provide an overview of a Scali System.
- to provide instructions for building a Scali System with respect to hardware and software installation.
- to provide instructions on how to use and manage a Scali System.
- to provide references to additional information and software.

Scali deliver its products:

- as complete turnkey systems.
- to OEMs and VARs.
- as “build your own supercomputer” kits (WulfKits).

Consequently, the wrapping and contents of your purchased items may vary but it is still called a Scali system throughout this document.

## 1.2 Overview of the Scali System Guide

The Scali System Guide is organised as follows:

- Chapter 2 presents a schematic and graphical **overview** of a Scali system.
- Chapter 3 describes the **hardware** installation procedure including the high speed interconnect adapters.
- Chapter 4 describes the **software** installation procedure including the Scali Software Platform.
- Chapter 5 describes how to **use** and **administrate** the Scali System through the *Scali Desktop* graphical user interface.
- Chapter 6 describes the Scali parallel shell tool suite, starting with the fundamental **scash** and then moving on to the more specialised tools like **scaps**.
- Chapter 7 gives a detailed description of scali system configuration with the complete reference to the command line based **scaconftool**.
- Chapter 8) gives more detailed information on the monitoring system: ScaMon.
- Chapter 9 describes how to **troubleshoot** common problems and get assistance from Scali.
- Appendix A describes the contents of the distribution media and lists each individual Scali software package.

Appendix B is a detailed reference to **scapkg**, the Scali software installation program

Appendix C gives details of the Scali licensing system ScaLM.

Appendix D gives a list of **interconnect** utility programs including status and error diagnostics.

Appendix E is the Scali System Cabinet Assembly guide.

Appendix F provides a list of **related documentation** you may consult for additional information.

## 1.3 How to read this guide

This guide is written for skilled computer users and professionals. It is assumed that the reader is familiar with the basic concepts and terminology of computer hardware and software since none of these will be explained in any detail. Depending on your user profile, some chapters are more relevant than others. We recommend that:

- system installers should read: Chapters 2, 3, 4 and 9
- system administrators should read: Chapters 2, 5, 6, 7, 8 and 9
- ordinary users should read: Chapters 2, 5 and 6

## 1.4 Acronyms and abbreviations

Abbreviation	Meaning
HPC	High Performance Computer
NIC	Network Interface Card
MPI	Message Passing Interface
SCI	Scalable Coherent Interface
SSP	Scali Software Platform is the name of all Scali software packages.

**Table 1-1:** Acronyms and abbreviations

## 1.5 Terms and conventions

Unless explicitly specified otherwise, gcc (gnu c-compiler) and bash (gnu Bourne-Again-SHell) are used in all examples.

Term	Description.
Node	A single computer in an interconnected system consisting of more than one computer
Cluster	A cluster is a set of interconnected nodes with the aim to act as one single unit
Scali system	A cluster consisting of Scali components
Frontend	A computer outside the cluster nodes dedicated to run configuration, monitoring and licensing software
MPI process	Instance of application program with unique rank within MPI_COMM_WORLD
UNIX	Refers to all UNIX and lookalike OSes supported by the SSP, i.e. Solaris and Linux.
WinNT	Refers to Microsoft Windows NT

**Table 1-2:** Basic terms

## 1.6 Typographic conventions

Term	Description.
<b>Bold</b>	Program names, options and default values
<i>Italics</i>	User input
<code>even spaced</code>	Computer related: Shell commands, examples, environment variables, file locations(directories) and contents

**Table 1-3:** Typographic conventions

Term	Description.
#	Command prompt in shell with super user privileges
%	Command prompt in shell with normal user privileges
^ ◻ ! = ? " "	<p>Notation for compact description of GUI operations.</p> <ul style="list-style-type: none"> <li>^ is click</li> <li>^^ double click.</li> <li>! is uncheck.</li> <li>◻ is a menu selection chain.</li> <li>◻ is a window open for multiple selections,</li> <li>= is a binary operator, left operand is name or category of selection, right operand is value of selection.</li> </ul> <p>It is either a predefined value or a specified string " " or ? input according to user context requirements. E.g.,  File◻New◻Project◻, ◻=Win32 Console Application,  ◻project name=?, ◻location=?</p>

**Table 1-3:** Typographic conventions

## Chapter 2 Scali system overview

---

This chapter gives a brief introduction to a Scali system. The intention is to give a comprehensive first time overview.

### 2.1 Description of a Scali system

A Scali system can be described with a simple equation:

**standard nodes + high speed interconnect + Scali software = Affordable Supercomputer**

Since a Scali system is built using high volume, low cost “off the shelf” nodes and a very powerful interconnect, the price vs. performance is extremely good. A Scali system is available in the range from small two node systems to large HPC systems with hundreds of nodes.

This **Scali System Guide** shows how to assemble hardware and install software to get an operative Scali system ready for general use (and optional 3rd party software installation). Another main purpose is to explain how to operate the Scali system from the **Scali Desktop** graphical user interface.

A **Scali system** is best explained in a top-down manner as a single system from software point of view and not as a cluster of interconnected nodes from a hardware point of view.

The Scali software integrates the **cluster** (nodes + interconnect) to a single system (parallel computer). All of the different Scali software components together constitutes the **Scali Software Platform (SSP)**.

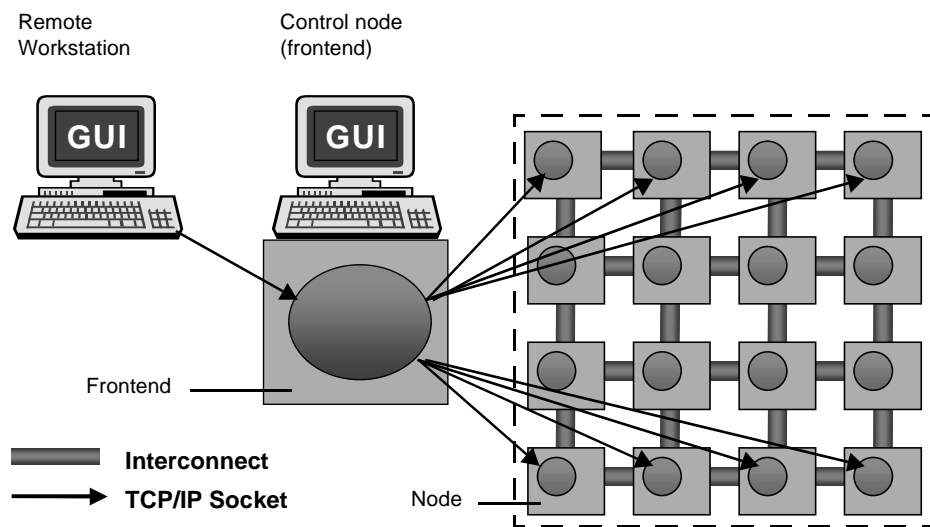
The Scali Software Platform (SSP) may be divided in 3 domains of usage:

- **Installation**
  - Operating system installation (ScaOSinstall)
  - Scali software installation (SSP install)
- **Administration/Maintenance**
  - Parallel software installation tool (ScaPkg)
  - Configuration/management of the SCI interconnect (ScaConf)
  - Power switching individually on each node (ScaConf)
  - Console management (ScaConf)
  - Running OS commands in parallel on the cluster (ScaSH)

- **Operational use**  
 Running OS commands in parallel on the cluster (ScaSH)  
 Running MPI applications (ScaMPI)

A Scali system is usually boxed in a Scali system cabinet (figure 2-2). Each compute node is put in its enclosure with interconnect cables on the rear (figure 2-3). On the front each node has a panel showing the CPUs activity. (Information about how to build a Scali system cabinet is included in the assembly guide in "Appendix E Scali System Cabinet Assembly Guide").

The Scali system usually has a frontend machine used for compiling, launching applications at the processing nodes and as a license server (figure 2-1). From the front-end or a compute node it is possible to install Scali software packages in parallel onto the entire cluster (or a smaller set of nodes).

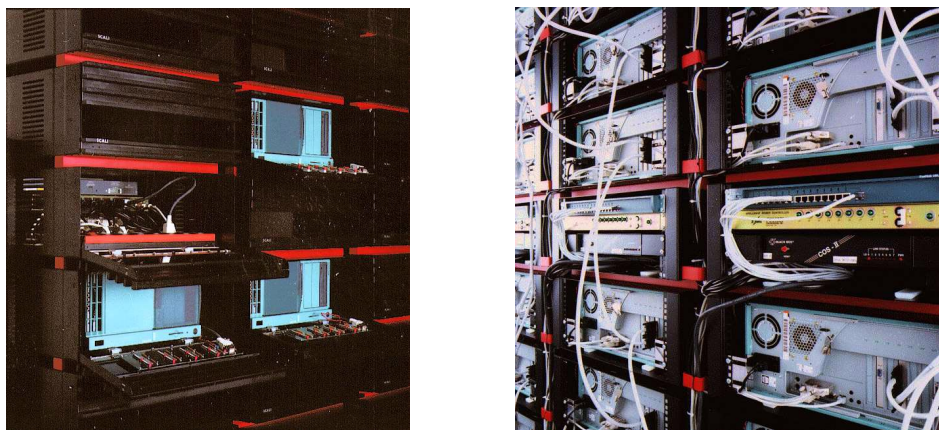


**Figure 2-1:** Basic structure of a 4x4 Scali system

## 2.2 Scali system example



**Figure 2-2:** A 4x8 Scali system boxed in Scali cabinets



**Figure 2-3:** Open front and rear side of a Scali system in a Scali cabinet



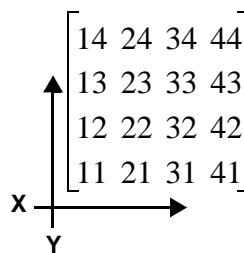


## Chapter 3 Hardware installation

This chapter explains how to organize your nodes, install and interconnect your hardware with main focus on the high speed interconnect.

### 3.1 Hardware organization

The most practical way to organize a cluster of interconnected computers is to stack them in a suitable system cabinet, usually in a 2-dimensional (2D) fashion. Scali recommend using a Scali system cabinet (see appendix E) or another boxed solution known to serve the requirements of cluster hardware management. We also recommend to label all nodes with a name reflecting physical location in the cabinet, i.e. node-XY where X and Y reflects the X and Y position respectively (figure 3-1).



**Figure 3-1:** A 4x4 cluster with XY coordinates: physical view

Some issues to remember when installing a cluster:

- The cluster must be placed in a location with sufficient cooling. Also be sure not to obstruct air ventilation channels on nodes and infrastructure equipment.
- The power outlets must be capable of supporting the cluster's accumulated power consumption.
- Ease of access to all equipment in the entire cluster for maintenance.
- A cluster has a lot interconnect cabling; try to keep the cables as short as possible.

## 3.2 Node BIOS preparation

### 3.2.1 MP level

In a multiprocessor node the BIOS sometimes has an option of configuring the so called 'MP level'. This specifies the protocol version used by the CPUs to interoperate. Possible values are usually 1.1 and 1.4. We recommend that you set this option to **1.4**.

### 3.2.2 Interrupt level mapping

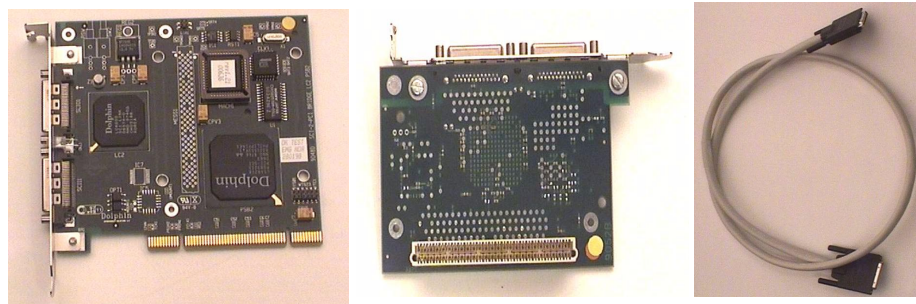
Some BIOSes have the option of enabling interrupt level mapping. If possible turn this feature **on** to reduce the possibility of conflicts with other devices.

### 3.2.3 Power on startup

If your system includes external power switches it is important that you configure the power section in the BIOS so that the node is automatically started when power is applied to the node.

## 3.3 Dolphin SCI interconnect installation

SCI is an international standard (IEEE1596) of a high speed interconnect. Dolphin Interconnect Solutions makes a number of products among which their PCI-SCI adapter cards with daughter cards are used by Scali.

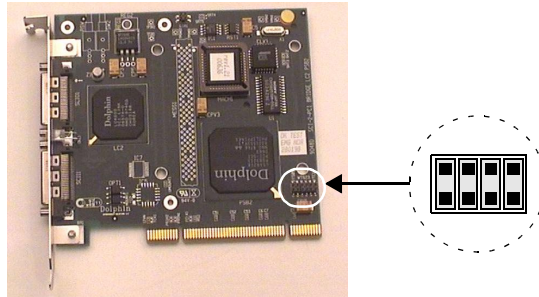


**Figure 3-2:** SCI hardware from Dolphin: PCI adapter, daughter card and cable

The performance of these cards are amazing: Several hundreds of megabytes per second and sub microsecond latency on the SCI link layer with PCI performance approaching the limits of the bus.

### 3.3.1 SCI adapter installation

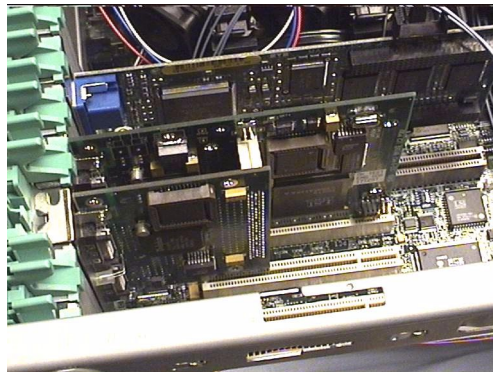
Before inserting the adapter card please make sure that all 4 jumpers are installed as indicated by figure 3-3.



**Figure 3-3:** SCI adapter card jumpers

If you have purchased a system with a two-dimensional SCI network you should have received a SCI daughter card. Please mount the daughter card on to the main SCI board if not already mounted.

Then insert the SCI adapter card (with daughter card if applicable) into a free PCI slot in the node. If there are ISA slots present you should if possible avoid the PCI slots sharing interrupt lines with ISA slots. Be sure to fix the board properly in the slot for mechanical stability.



**Figure 3-4:** Node with SCI adapter card and daughter card inserted

Please repeat the above procedure on all nodes in the cluster.

**Note:** When power is switched on, the LEDs on the SCI cards will be red or yellow (indicating error) and remain so until the cabling is done and software is loaded - this is quite normal.

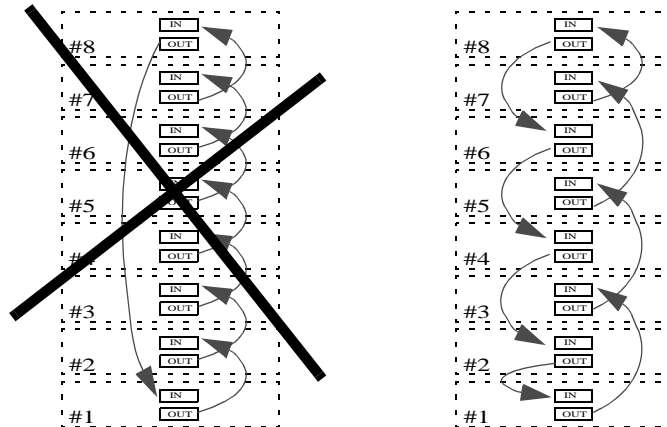
### 3.3.2 SCI Cabling

The recommended interconnect topology depends on the number of nodes in the system:

- **2-8 nodes** : Single SCI ring topology
- **4-> nodes** : Two-dimensional SCI ring (torus) topology (requires daughter card)

The two topologies are described separately. Along with this manual you should have received a description of which topology is used in your system. Interconnect cabling of simple torus topologies are somewhat complicated by system cabinets with another size and shape than the interconnect topology. The need to avoid long SCI cables adds another level of complexity to the interconnect cabling pattern. Some general rules apply though, when connecting the cables:

- Always connect the SCI cables from the OUT connector to the IN connector.
- Interleave cabling to avoid long cables (see figure 3-5).



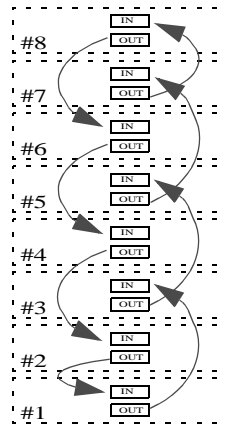
**Figure 3-5:** Avoid long cables by interleaving interconnect in each ring

**3.3.2.1 Single SCI ring cabling**

In small systems a single ring can be efficient. Figure 3-6 shows how to connect a 8 node SCI ring based on the table below:

From (OUT)	To (IN)
Node #1	Node #3
Node #3	Node #5
Node #5	Node #7
Node #7	Node #8
Node #8	Node #6
Node #6	Node #4
Node #4	Node #2
Node #2	Node #1

**Table 3-1:** Example of cabling a 8 node SCI ring



**Figure 3-6:** Example of 1x8 single ring cabling in a 1x8 cabinet

**3.3.2.2 Two-dimensional SCI torus cabling**

The two-dimensional cabling follows the same scheme in each ring as the single ring cabling. In addition we have the following guidelines:

- Connect SCI main cards in Y (vertical) direction

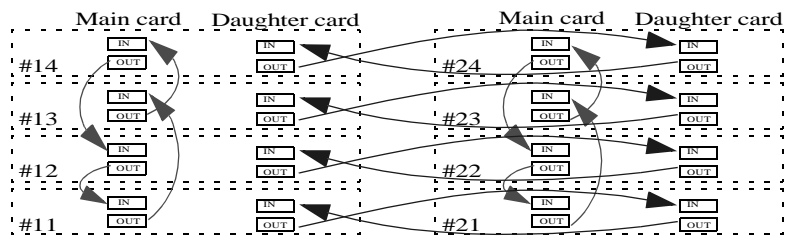
From (OUT)	To (IN)
Node #x1	Node #x3
Node #x3	Node #x4
Node #x4	Node #x2
Node #x2	Node #x1

**Table 3-2:** Example cabling of one vertical ring in a 4x4 system (x=[1-4])

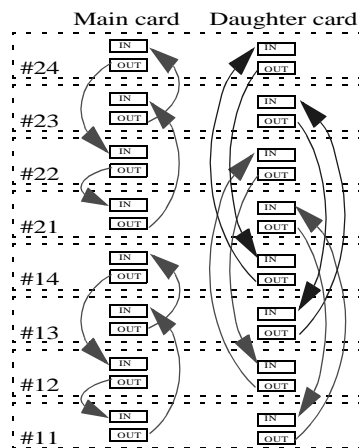
- Connect SCI daughter card in X (horizontal) direction

From (OUT)	To (IN)
Node #1y	Node #3y
Node #3y	Node #4y
Node #4y	Node #2y
Node #2y	Node #1y

**Table 3-3:** Example cabling of one horizontal ring in a 4x4 system (y=[1-4])



**Figure 3-7:** Example of 2x4 two-dimensional system cabling in a 2x4 cabinet



**Figure 3-8:** Example of 2x4 two-dimensional system cabling in a 1x8 cabinet

### 3.4 Console management

Scali systems can be equipped with support for accessing node consoles on the serial port on each node. The consoles are made accessible either through terminal servers converting the serial line to the telnet protocol over ethernet, or directly to terminal devices on a Unix server. The cabling is heavily dependant on the type of device used, and is described in separate datasheets where needed.

### 3.5 Power switching

Scali systems can be equipped with fully remote controlled power switches. This enables remote (and centralized) control for switching node power on and off on individual nodes through the configuration software. The cabling is heavily dependant on the type of power switching device used, and is described in separate datasheets where needed.





This chapter explains how to install Scali software on the cluster. Before software installation is attempted, the cluster and interconnect hardware must be installed according to the guidelines in Chapter 3

## 4.1 Software installation overview

To install the SSP onto a cluster you need to take the following steps:

- **OS installation:** Install and configure the operating system either with the *ScaOSinstall* program or manually with guidelines outlined later in this chapter.
- **Scali software installation:** Install Scali software with the SSP installation program: **install**. The installation program installs selected Scali software on the entire cluster. Installation includes configuration and testing of the installed hardware and software on different levels.

The required software and instructions exists either on the Scali SSP CD-ROM or can be downloaded from:

<http://www.scali.com/download>

## 4.2 Operating system installation

Each node in the cluster needs a proper OS installed before the Scali software platform can be installed on the Scali system. Please read the */doc/os* file included on your Scali software platform distribution media (CD-ROM).

### 4.2.1 OS prerequisites

Before installing the Scali software you should install and configure your operating system with the following in mind:

- From the host you install from (the frontend) you must have root `rsh` access to all other hosts/nodes. Please do the following steps on every node:
  - Insert the frontend name into the file `$HOME/.rhosts` as root.
- Enable `rlogin` as root from the frontend to all the nodes (it is nice to have `rlogin` access as root to each node, - also with `telnet`). Please do the following steps on each node:
  - Linux:
    - RH:

- Edit `/etc/pam.d/rlogin` and `/etc/pam.d/login`, remove the `pam_securetty` check or add hosts to pam authentication.
- SuSE:
    - Run `yast`, choose System administration, change configuration file and set the entry `ROOT_LOGIN_REMOTE` to yes.
  - Have a working 'at' daemon (`atd`) running on all nodes. Verify that it works:
 

```
# echo touch /tmp/atfile | at now
# test -r /tmp/atfile && echo OK; test ! -r /tmp/atfile && echo ERROR
```

 If not please do the following steps:
    - Linux:
      - Check that package is installed:
 

```
# rpm -q at
```
      - Ensure that the at daemon will be started at boot:
        - RH :
 

```
# chkconfig atd on
```
      - SuSE:
        - Run `yast`, choose System administration, change configuration file and set the entry `START_ATD` to yes.
    - Check if the daemon is running:
      - RH :
 

```
# /etc/rc.d/init.d/atd status
```
      - Start the daemon if not:
 

```
# /etc/rc.d/init.d/atd start
```
      - SuSE:
 

```
# ps aux | grep atd | grep -v grep
```
      - Start the daemon if not.
    - Solaris:
      - Ensure that the at cron daemon will be started at boot:
 

```
# test ! -r /etc/rc2.d/S75cron && ln -s /etc/init.d/cron /etc/rc2.d/S75cron
```
      - Check if the cron daemon is running:
 

```
# ps -ef | grep cron | grep -v grep
```
      - Start the cron daemon if not:
 

```
# /etc/init.d/cron start
```
  - Set up console over the serial port.
    - Linux:
      - Edit `/etc/inittab` to set up console over serial port
        - RH:
 

```
S0:2345:respawn:/sbin/getty ttyS0 9600 vt100
```
        - SuSE:
 

```
S0:123:respawn:/usr/sbin/getty ttyS0 9600 vt100
```
      - Add `ttys0` to `/etc/securetty`.
      - Edit `/etc/lilo.conf` to enable lilo to operate over serial port

and inform kernel to use console over serial port.

Include these lines in `/etc/lilo.conf`:

```
serial=0,9600n8
append="console=ttyS0"
```

and run `/sbin/lilo` to make the settings active.

Reboot the system

•Solaris:

Configure with eeprom as root:

```
# eeprom ouput-device=ttya
# eeprom input-device=ttya
```

- Have a common file system for MPI applications. Usually this means mounting `/home/*` to a common NFS server.
- We recommend using NIS on the cluster to ease management. Each OS specific description below includes a section about how to enable NIS.
- Desktop requires OpenGL library (workstation where Desktop is running):  
Linux : Mesa has been tested.  
Microsoft: MS OpenGL has been tested.  
SunOS/Solaris: - Mesa and SunOpenGL has been tested (on Sparc).

It is possible to use the automatic OS installation program **ScaOSinstall** but if this is not included in your Scali software option please follow the manual OS installation guidelines below.

## 4.2.2 Linux specific setup

### 4.2.2.1 NIS setup on RedHat.

On the NIS master (usually the frontend):

- set NIS domainname, run `# nisdomainname <yourdomain>`
- edit `/etc/yp.conf`
- edit `/etc/nsswitch.conf`
- edit `/var/yp/securenets`
- run `# sh /etc/rc.d/init.d/ypserv start`
- cd to `/var/yp` and run `# make`
- run `# sh/etc/rc.d/init.d/ypbind start`

On the NIS clients (all other nodes):

- run `# authconfig` and specify NIS domainname and request server via broadcast.

#### 4.2.2.2 NIS setup on SuSE.

On the NIS master (usually the frontend):

- Run `yast`, choose System administration. Change configuration file and:
  - set `YP_DOMAINNAME` to NIS domainname
  - set `YP_SERVER` to correct IP address
  - set `START_YPSERV` to yes
  - set `START_YPBIND` to yes

On NIS clients (all other nodes):

- Run `yast`, choose System administration, change configuration file and:
  - set `YP_DOMAINNAME` to NIS domainname
  - set `YP_SERVER` to correct IP address

#### 4.2.2.3 The bigphysarea kernel patch (upgrade only)

Prior to SSP2.0 the bigphysarea kernel patch was needed. In release 2.0 the memory provided by the patch is no longer used and the patch is not needed. (In fact it will only make a large portion of your memory inaccessible from the system and normal applications) If you have the patch installed, change the size of the memory maintained by the patch to 0:

- edit `/etc/lilo.conf`
- change append `bigphysarea=nnnn` to append `bigphysarea=0`
- run: `# lilo`

### 4.2.3 Solaris specific setup

#### 4.2.3.1 NIS setup on Solaris.

On the NIS master (usually the frontend):

- set NIS domainname, run `# domainname <yourdomain>`
- EITHER have these files under `/etc`:
  - `timezone`
  - `netgroup`
  - `bootparams`
 OR edit `/var/yp/Makefile` to not use these maps.
- run `# ypinit -m`
- edit `/etc/nsswitch.conf`
- edit `/var/yp/securenets`
- run `# /usr/lib/netsvc/yp/ypstop`
- run `# /usr/lib/netsvc/yp/ypstart`

On the NIS clients (all other nodes):

- set NIS domainname, run `# domainname <yourdomain>`
- run `# ypinit -c`

- `run # /usr/lib/netsvc/yp/ypstop`
- `run # /usr/lib/netsvc/yp/ypstart`

## 4.3 Scali software installation

When the interconnect is installed and the operating system configured to suit the Scali software, it is time to start the SSP installation program: **install**, on the front-end. The frontend is used to install all the later chosen software packages on to the cluster nodes.

You will need either a demo license (which you can ask for at **license@scali.com**) or an existing permanent license file before you can complete the installation. Please follow the instruction given by the installation program.

### 4.3.1 Install session example

```
[root@pc-5 SSP_2_0]# ./install

== Introduction =====

Welcome to the Scali Software Platform (SSP) installation program
This program will guide you through the installation procedure.

It is divided into several sections:
  Introduction
    - some general information
    - some sanity checks
  Configuration
    - specify nodes to install on
    - specify software to install
  Installation
    - check that the nodes are correctly prepared
    - the actual software installation on all nodes
    - post installation setup/configuration
  Verification
    - verify that the installation was successful

If you abort in the middle of the installation process you have the
option of saving the current state and restore it at the next invocation.
You may also confirm each single step of the restore to do modifications
anywhere in the restore process.

Whenever your input is required you will see a line like this:
  Q: ...? [<default option>]
The default option is given in square braces and is chosen if you
just press return.

If you experience problems using this program or in general have
comments please feel free to contact us at:
  support@scali.com

(version SSP_2_0 - install : 1.50 )

-- Restoring info from savefile ----- ? --
We have detected that you aborted a previous installation attempt.
You have the option of restoring the information recorded up to the
point where you last exited.
If you choose to restore you will also be able to qualify each step
if you want to change something in the recorded information.
Please choose one of the following:
  y : restore (default)
  s : restore with single step qualifying
  n : ignore restore possibility

  Q: Do you want to restore the recorded state: [y] n

-- Expert mode ----- ? --
This program tries to reduce user interaction to a minimum.
If you would like to be able to control the process in more detail:
  - Support diskless clients
  - Modify default install set
  - Choose a different package repository
```

```

- Modify node configuration in an upgrade
- Read HW and OS documentation
please answer yes to the question below.
Under normal circumstances this should not be necessary.

Q: Do you want to run it in expert mode: [n] y

-- Checking permissions ----- OK --
-- Checking source media ----- OK --

== Installation configuration =====

-- Checking upgradeability ----- INSTALL --
-- Checking frontend ----- ? --
This program should only be started on the machine you want to be
the frontend. The frontend is among other responsible of:
- installing Scali software and upgrades on the cluster
- running the licensing daemon
- running the configuration and monitoring servers
In small systems (< 8 nodes) this is usually one of the compute nodes.
In larger systems a separate node is preferred.

Q: Are you sure you want to use this machine as an frontend: [y]

-- Checking HW documentation ----- ? --
Do you want to read the HW installation guide before continuing?

Q: Read it now: [n]

-- Checking OS documentation ----- ? --
Do you want to read the OS installation guide before continuing?

Q: Read it now: [n]

-- Specifying node names ----- ? --
Please identify the nodes which has an SCI interface installed.
(NB: this has to match the real node names!)
Please select which method of node specification:
  1) Automatically create names by setting dimensions and base name
  2) Manually specify each node name and it's position in the mesh

Q: Please enter your selection: [1]

The nodes are always looked upon being arranged in a 2D mesh (X,Y)
If you only have a single ring, please set the X dimension to 1
and use the number of nodes as the Y dimension.

Q: Please specify the X dimension of the mesh (1-15): 1
Q: Please specify the Y dimension of the mesh (1-15): 2

The nodes are named according to the following scheme:
<prefix><x-position><separator><y-position><postfix>
(eg. scil-1 where prefix=sci, separator=- and postfix is empty)

Q: Please specify the prefix to use on the nodes (e.g. 'sci') : scali4-1
Q: Please specify the separator to use on the nodes (e.g. '-'):
Q: Please specify the postfix to use on the nodes (e.g. '') :
```

```

The following nodes has been defined (nodename xpos ypos):
  scali4-111    1
  scali4-121    2

  Q: Do you accept the above configuration: [y] y

-- Specifying Repository ----- ? --
Please give the path to where you want the package repository to be created
The package installation tool will find its packages in this directory.

  Q: Please enter repository path: [/opt/scali/repository]

Creating non-existent path /opt/scali/repository/Linux2.i86pc ...

-- Specify Software to Install ----- ? --
Please select which packages to install:
  1) Complete SSP
  2) Explicit package selection(s)

  Q: Please enter your selection: [1]

-- Preparing repository ----- ! --
-- Copying packages to repository ----- ! --
ScaSH      : Removing old versions and copying files to repository: +
ScaPkg     : Removing old versions and copying files to repository: ++
ScaSCI     : Removing old versions and copying files to repository: ++
ScaSISCI   : Removing old versions and copying files to repository: +
ScaMPI     : Removing old versions and copying files to repository: +
ScaMPItst  : Removing old versions and copying files to repository: +
ScaMPE     : Removing old versions and copying files to repository: +
ScaLM      : Removing old versions and copying files to repository: ++
ScaLMnode  : Removing old versions and copying files to repository: ++
ScaEnv     : Removing old versions and copying files to repository: +
ScaDesk    : Removing old versions and copying files to repository: +
ScaConfNd  : Removing old versions and copying files to repository: +
ScaConfSd  : Removing old versions and copying files to repository: +
ScaConfC   : Removing old versions and copying files to repository: +
ScaCmd     : Removing old versions and copying files to repository: +
ScaExecd   : Removing old versions and copying files to repository: +
ScaSNMPd   : Removing old versions and copying files to repository: +
ScaSNMPt   : Removing old versions and copying files to repository: +
ScaMond    : Removing old versions and copying files to repository: +
ScaFPDisp  : Removing old versions and copying files to repository: +

-- Copying config to repository ----- ! --
-- Diskless clients support ----- ? --
The following question is only valid if you have configured the nodes
to be diskless clients (having no disk on the nodes).
(NOTE: All or none of the nodes should be diskless; a mix is not supported)

  Q: Are all of the nodes diskless clients of the frontend: [n]

-- Modifying ScaSCI options ----- ! --
-- Modifying ScaLMnode options ----- ! --
-- Installing installation tools ----- ! --
ScaSH      : Preparing package configuration: +
ScaSH      : Installing package
ScaPkg     : Preparing package configuration:

```



```

ScaPkg      : Installing package

-- Modifying ScaPkg config      ----- !      --
-- Setup license(s)            ----- ?      --
Please select which type of license(s) you have:
  1) Demo license(s) (by specifying FEATURE line(s))
  2) Full license(s) (from an existing valid license file)

Q: Please enter your selection: [1]

We will use demo license(s) during the installation until everything
is up and running.
The demo license(s) will be replaced by permanent license(s) after sending
a license request to our licensing department (license@scali.com).

The following information you should have received either as part
of the installation media or in a separate e-mail:
(press <return> on an empty line after adding your feature line(s))

Q: Please enter a feature line: FEATURE mpi scald 1.900 1-jul-2000 uncounted
66FF5D54010A HOSTID=DEMO ck=166
Q: Please enter a feature line: FEATURE confsd scald 1.000 1-jul-2000 uncounted
830AF6D76B6B HOSTID=DEMO ck=119
Q: Please enter a feature line:

You have entered these feature lines:

FEATURE mpi scald 1.900 1-jul-2000 uncounted 66FF5D54010A HOSTID=DEMO ck=166
FEATURE confsd scald 1.000 1-jul-2000 uncounted 830AF6D76B6B HOSTID=DEMO
ck=119

Q: Do you want to keep them as is? [y]

Modifying ScaLM ...

-- Checking access to nodes      ----- OK      --
-- Testing for used applications ----- OK      --
-- Testing for glibc memory leak ----- NO      --
-- Checking AT daemon           ----- OK      --
-- Checking bigphysarea patch   ----- NO      --
-- Checking SCI boards          ----- OK      --
-- Checking MPI service         ----- OK      --

== Software installation      =====

-- Installing packages          ----- Wait ... --
Checking availability on hosts chosen to be updated, please wait ...
This program installs software on specified (or default) hosts
version      : 1.41
repository   : /opt/scali/repository

Checking configuration
Cleaning up from previous runs on all hosts ...
Find platforms on hosts ...

Find out on each host which packages are needed

```

```

Find out which packages we have for the platforms detected
Removing invalid platforms
Removing invalid hosts due to invalidated platforms
Find out which packages which go to each node (based on configuration data)
  Find out which packages are valid for each category
  Find out which packages are valid for each host
Match valid package names with available packages in repository
Removing hosts not allowed for updates or no packages available
Starting remote package polling ...
Copy host dependent package list to respective hosts ...

Waiting on feedback from nodes which packages they need
Poll for list of needed packages on remote nodes ...
Remove hosts not in need of any updates

Copy packages to nodes
Copy files to nodes ...
  Checking platform: Linux2.i86pc
    Package ScaComd      is now copied to      2 nodes ...
    Package ScaDesk     is now copied to      1 nodes ...
    Package ScaEnv      is now copied to      3 nodes ...
    Package ScaExecd    is now copied to      1 nodes ...
    Package ScaFPDisp   is now copied to      2 nodes ...
    Package ScaLM       is now copied to      1 nodes ...
    Package ScaLMnode   is now copied to      3 nodes ...
    Package ScaSCI      is now copied to      3 nodes ...
    Package ScaSH       is now copied to      2 nodes ...
    Package ScaSISCI    is now copied to      3 nodes ...
    Package ScaSNMPd    is now copied to      2 nodes ...
    Package ScaSNMPt    is now copied to      1 nodes ...
    Package ScaConfNd   is now copied to      2 nodes ...
    Package ScaConfSd   is now copied to      1 nodes ...
    Package ScaMond     is now copied to      1 nodes ...
    Package ScaMPI      is now copied to      3 nodes ...
    Package ScaMPItst   is now copied to      3 nodes ...
    Package ScaConfC    is now copied to      1 nodes ...
    Package ScaMPE      is now copied to      3 nodes ...
  Tell all nodes that all packages has been copied ...

Get installation results from nodes
Poll for post installation log files on remote nodes ...
Checking log files
  pc-5      : No errors
  scali4-11 : No errors
  scali4-12 : No errors

Cleaning up after us and exiting

== Post installation fix      =====
-- Configuring Interconnect  ----- !      --
  Configuring interconnect, please wait some seconds ...
  Requesting server to reroute with default routing algorithm for ring.

-- Create license request    ----- ?      --
  We will now create a permanent license request which is to be
  sent to license@scali.com.

```

(If you wish to send it as an attachment or resend it later use this file:  
 /tmp/SSPinstall.licrequest)  
 Gathering necessary information to create request file:

Q: Please type email address to recipient of license file:  
 administrator@company.com  
 Q: Please type company name : Company Inc.  
 Q: Please type order reference number : S01234

-----  
 SSP\_2\_0 (version: 1.50 ) license request

Date : Mon Jan 17 18:37:11 CET 2000  
 Recipient : administrator@company.com  
 Company : Company Inc.  
 Order refno : S01234

Frontend : pc-5  
 Domainname : company.com  
 OS : Linux 2.2.12-20

-----  
 pc-5 : The FLEXlm host ID of this machine is "00a02457386d"  
 scali4-11 : The FLEXlm host ID of this machine is "00a0c9e9e9bc1"  
 scali4-12 : The FLEXlm host ID of this machine is "00a0c9d4cacb"  
 -----

Q: Do you want to send the request now: [y]

### A file with permanent licenses will be returned shortly which you have to  
 ### copy to pc-5:/opt/scali/license/license.dat and run the following:  
 ### pc-5# /opt/scali/init.d/scald restart

== Test and verification =====

```
-- Checking SCI driver ----- OK --
-- Checking SCI board jumpers ----- OK --
-- Checking SCI links ----- OK --
-- Checking SCI communication ----- OK --
-- Checking MPI communication ----- OK --
-- Checking MPI performance ----- ! --
```

We will now test the installation by running a benchmark:

Alltoall size 2 [2 procs] 131072k in chunks of 0k4 - 512k0 transfered

Size	Total time	Iter	Sample	time	Bandwidth	[Resolution 2.51us]
4	0.02	512	30.03	0.13		
8	0.01	512	25.12	0.30		
16	0.01	512	26.20	0.58		
32	0.01	512	27.29	1.12		
64	0.02	512	33.08	1.84		
128	0.02	512	37.04	3.30		
256	0.02	512	43.52	5.61		
512	0.03	512	58.27	8.38		
1k	0.03	512	63.73	15.32		
2k	0.05	512	91.71	21.30		
4k	0.08	512	165.70	23.57		
6k	0.12	512	239.85	24.43		
8k	0.16	512	309.05	25.28		

12k	0.22	512	421.93	27.77
16k	0.28	512	540.92	28.89
24k	0.41	512	801.49	29.24
32k	0.54	512	1050.99	29.73
48k	0.80	512	1558.88	30.07
64k	1.07	512	2083.94	29.99
96k	1.64	512	3209.56	29.21
128k	2.31	512	4510.60	27.71
192k	3.88	512	7578.91	24.74
256k	4.94	512	9654.29	25.90
384k	4.97	342	14539.27	25.79
512k	4.94	256	19285.71	25.93

```
== Install complete! =====
```

```
-- Epilogue ----- ! --
```

```
Scali Software Getting Started Guide
```

```
-----
Most of the Scali software is accessible through the graphical
desktop; ScaDesktop.
To start the desktop issue this command on the command line:
```

```
# /opt/scali/bin/scadesktop
```

```
Please note that the desktop will be different for root than
for ordinary users.
For help on specific issues regarding the installation and
operation of the Scali software please refer to the "Scali
System Guide".
```

```
---
```

```
(document version: $Id: GettingStarted,v 1.1 1999/12/13 02:45:40 rws Exp $)
```

```
== Clean up and exit ... =====
```

```
The information you have specified has been recorded in:
/tmp/SSPinstall.savestate
and can be restored in later invocations of this program.
```

### 4.3.2 Uninstall session example

```
[root@pc-5 SSP_2_0]# ./uninstall

== Checking installation =====

Welcome to the Scali Software Platform (SSP) uninstallation program

This program will guide you through the uninstallation procedure.

Whenever your input is required you will see a line starting with 'Q: ...'.
The default option is given in square braces and is chosen if you press return.

If you experience problems using this program or in general have comments
please feel free to contact us at:
    support@scali.com

-- Checking permissions ----- OK --
-- Checking Frontend ----- OK --
-- Finding existing frontend ----- OK --
    The following node will be used:
        pc-5

-- Finding existing nodes ----- OK --
    The following nodes will be used:
        scali4-11
        scali4-12

-- Finding existing repository ----- OK --
    The following repository path will be used:
        /opt/scali/repository

-- Checking access to nodes ----- OK --

== Remove installation =====

-- Confirmation ----- ? --
    You will now remove the SSP from all involved nodes.

    Q: Please press "yes" to confirm, "no" to decline: [no] yes
-- Remove packages and files ----- ! --
    Removing packages on nodes ...
scali4-12 : Sending signal to (scacomd) process(es) (pid: 593 )
scali4-11 : Sending signal to (scacomd) process(es) (pid: 692 )
scali4-12 : Sending signal to (mpid) process(es) (pid: 4257 )
scali4-11 : Sending signal to (scasnmpd) process(es) (pid: 714 715 716 )
scali4-12 : Sending signal to (scasnmpd) process(es) (pid: 615 616 617 )
scali4-11 : Sending signal to (scaconfnd) process(es) (pid: 702 703 704 )
scali4-11 : Sending signal to (scaconfnd) process(es) (pid: 702 703 )
scali4-12 : Sending signal to (scaconfnd) process(es) (pid: 603 604 605 )
scali4-11 : Sending signal to (scaconfnd) process(es) (pid: 702 703 )
scali4-12 : Sending signal to (scaconfnd) process(es) (pid: 603 604 )
scali4-11 : Sending signal to (scaconfnd) process(es) (pid: 702 703 )
scali4-12 : Sending signal to (scaconfnd) process(es) (pid: 603 604 )
scali4-11 : Trying to kill (scaconfnd) process(es) (pid: 702 703 )
scali4-12 : Sending signal to (scaconfnd) process(es) (pid: 603 604 )
scali4-11 : Sending signal to (mpid) process(es) (pid: 4838 )
```

```
scali4-12      : Trying to kill  (scaconfnd) process(es) (pid: 603 604 )
scali4-12      : Sending signal to  (scid) process(es) (pid: 8210 )
scali4-11      : Sending signal to  (scid) process(es) (pid: 627 )
```

Removing files on nodes ...

Removing packages on frontend

```
Sending signal to  (lmgrd) process(es) (pid: 25229 )
Sending signal to  (scaconfsd) process(es) (pid: 28021 28025 28027 )
Sending signal to  (scaconfsd) process(es) (pid: 28021 28025 )
Sending signal to  (scaconfsd) process(es) (pid: 28021 28025 )
Sending signal to  (scaconfsd) process(es) (pid: 28021 28025 )
Trying to kill  (scaconfsd) process(es) (pid: 28021 28025 )
Sending signal to  (mpid) process(es) (pid: 19661 )
```

Removing files on frontend

Removing repository on frontend

```
== Uninstall complete!      =====
```

This chapter explains how to use a Scali system through a single point of control, the **Scali Desktop** graphical user interface.

### 5.1 Overview

The Scali Desktop enables the use, monitoring and configuration of a Scali system as a single entity, rather than a collection of autonomous nodes. The GUIs for all Scali software modules are dynamically integrated in the Scali Desktop framework as plugins. As new tools become available (or purchased) they will show up as new menus or menu items in the Scali Desktop main window after installation.

The Scali Desktop tools are sold as a complete set or as separate modules. Hence some of the modules described in this chapter may not be available with all installations, in such cases the options will not show up in the menus. Extra modules may be purchased from Scali (contact [sales@scali.com](mailto:sales@scali.com)).

### 5.2 Getting started with the Scali Desktop

The only prerequisites for running the Scali Desktop is that the machine you are using has network access to the front-ends of the Scali Systems you want to use. If you want to use the 3D monitoring options you must also have an OpenGL 1.2 compatible library installed. on Linux systems this is provided by the Mesa-3.2

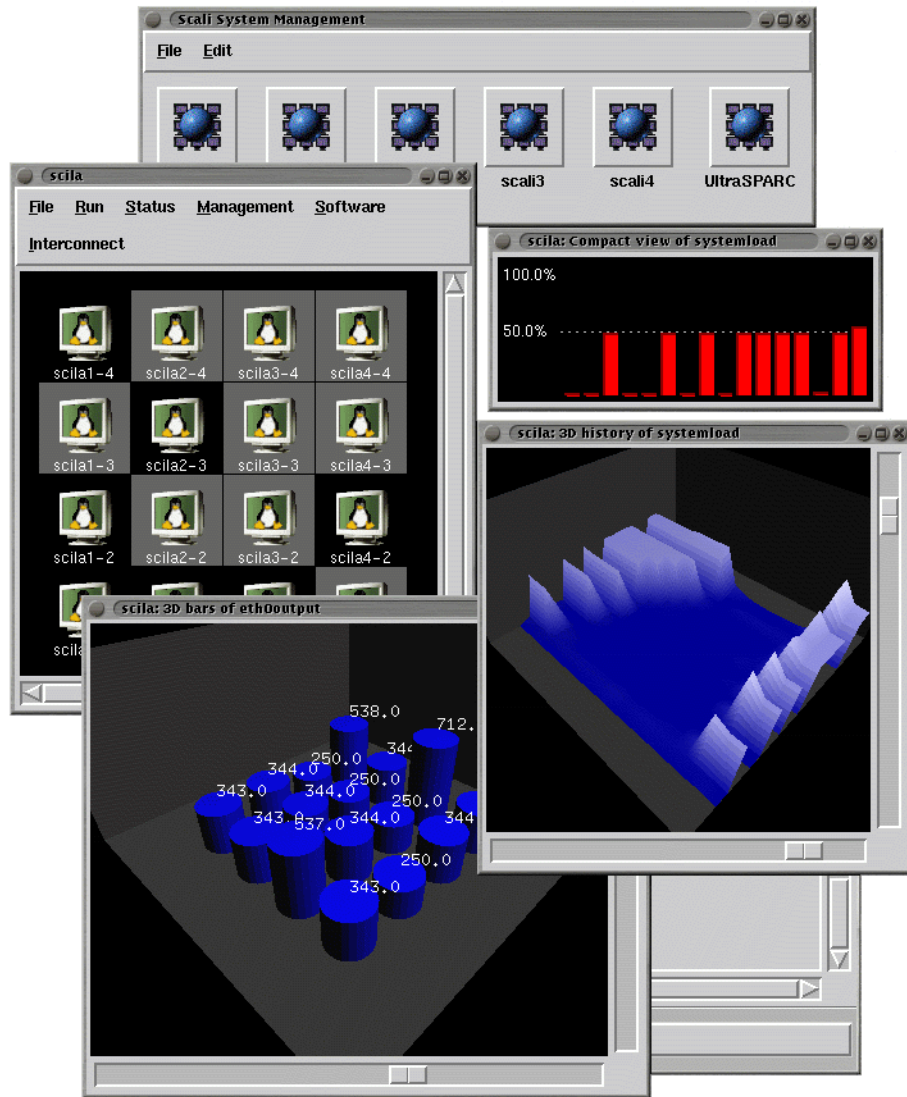
Due to the heavy use of graphic presentations, we also recommend that Scali Desktop is run on your local workstation.

To start Scali Desktop on Unix systems, type:

```
/opt/scali/bin/scadesktop
```

On Windows systems use the “Run” option from the start menu, or double-click it and the program will start.

The Scali System Management window now appears. Scali Desktop can control multiple Scali Systems each consisting of multiple nodes. The Scali System Management window is used for selecting which Scali System(s) to manage



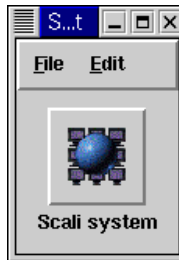
**Figure 5-1:** An example ScaDesktop session with multiple monitoring views



### 5.2.1 Configuration of the Scali Desktop

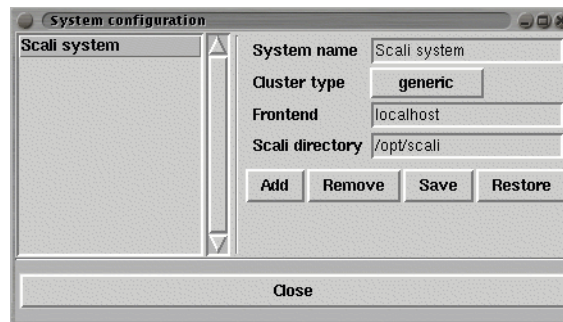
Before you can start using the Scali Desktop, you must tell it a few things about the systems you want to manage. This is known as configuring the Scali Desktop. As a result, you and every other user will have a personalized desktop configuration (user profile). The user profiles are stored in the file: `$HOME/.scali/ScalDesk.conf`.

When you start Scali Desktop for the first time you will see a dialogue box with the following warning: "Could not read configfile, using system defaults". This is because no configuration file was found in `$HOME/.scali`. Just press "OK" to get on with the configuration. The Scali System Management window then appears.



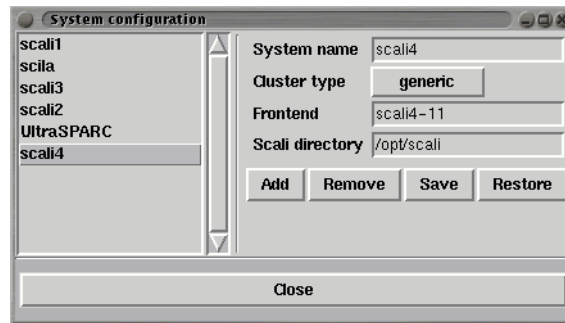
**Figure 5-2:** Initial unconfigured Scali System Management window

Initially, only a default Scali System with system default values is available, These default values have very limited use, since they only work if you run the desktop locally on the frontend of a Scali system. Hence you should configure Scali Desktop to enable remote access to this and all other available Scali systems. Select **Edit**→**Configuration...** from the main menu to open the system configuration window.:



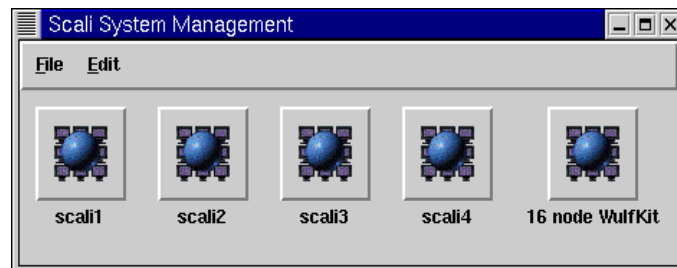
**Figure 5-3:** Configuration window showing system defaults.

In the System Configuration window select the “*Scali system*” and edit System name and Frontend to the correct values for your system. Usually default values for Cluster type and Scali directory are correct, and can be left unchanged. To make the changes take effect and save the newly edited configuration press Save. More Scali Systems may be added to your configuration like this: Press the Add button and you will be prompted for a new system name. Enter a unique system name, edit the Frontend name and press Save again. .



**Figure 5-4:** Example System configuring

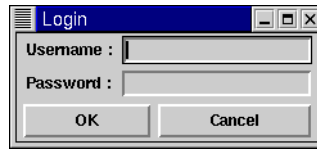
Removal of undesired systems is done by selecting a system and then pressing the Remove button. The Restore button will restore the contents of the configuration window from the last saved profile. Finally to finish the configuration, Close the System Configuration window and the System Management window will be updated to show a “cluster icon” and name for each of the configured systems.:



**Figure 5-5:** Scali System Management window with five Scali Systems

### 5.2.2 Logging in to a system

Much like any normal networked computer you must also log into the Scali System before you can start using any of the tools in ScaDesktop. To login to a scali system you simply click on the corresponding system icon in the Scali System Management window. You will then be presented with a "login" dialogue box. Enter your user name and password and press OK.



**Figure 5-6:** Login dialogue box

The username and password will then be encrypted and sent for verification locally on the frontend for this Scali system. Please note that it is therefore your access rights on the Scali system that counts, not your access rights on the workstation running the Scali Desktop. Depending on whether you log in as a regular user or as "root" the Scali Desktop will operate in either:

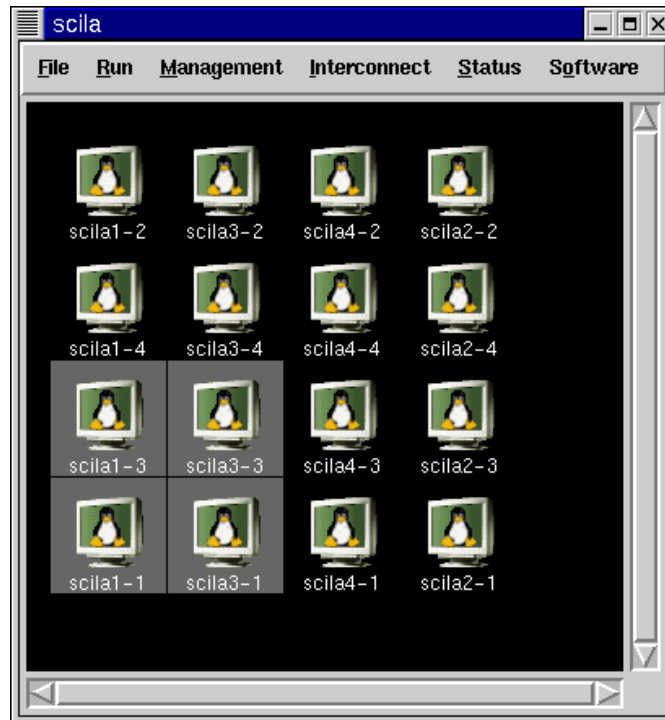
- regular user mode
- system administrator mode

The difference being that some tools, notably configuration and maintenance modules are not available to regular users. Maintenance and configuration modules gives access to remote power control, interconnect configuration and software installation.

## 5.3 Using the Scali Desktop





A user or administrator may log into a Scali System by selecting a Scali System icon in the Scali System Management window (see Table 5-1 on page 40). The Scali Desktop prompts for username and password, these are encrypted and checked at the frontend node. Note that maintenance modules are only available when logged in as system administrator.

When logged in to a Scali System, the main window of Scali Desktop appears.



**Figure 5-7:** Scali Desktop main window

The main window of the Scali Desktop shows all nodes in the network. A symbol indicates the operating system and status of each node:

	An operational node running the Linux operating system
	An operational node running the Solaris operating system.
	Node for which the status and operating system has not yet been determined
	Node unreachable.

**Table 5-1:** Node symbols in the main window of the Scali Desktop

A node may be selected by dragging the mouse and pressing the left mouse button. Additional nodes may be added to a selection by pressing **<Ctrl>** and left mouse button.

Options available may be selected from the menu bar. The following options are always available to all:

<b>File</b> ⇒Close	Exit program
<b>Run</b> ⇒MPI Program...	Run MPI program, see “Executing MPI programs” on page 48.
<b>Run</b> ⇒Parallel shell...	Run shell commands at selected nodes, see “Running Parallel Shell commands” on page 51
<b>Run</b> ⇒X terminal	Start an X terminal at selected nodes, see “Executing programs - Run menu” on page 48
<b>Run</b> ⇒X terminal with command...	Start an X terminal with a command at selected nodes, see “Executing programs - Run menu” on page 48
<b>Status</b> ⇒Process list...	List processes for user at selected nodes, see “System monitoring - Status menu” on page 55
<b>Status</b> ⇒User list...	List users at selected nodes, see “System monitoring - Status menu” on page 55
<b>Status</b> ⇒Uptime and load...	Show the uptime and load for the selected nodes, see “System monitoring - Status menu” on page 55
<b>Status</b> ⇒Link status...	Show the status of the interconnect links, see “System monitoring - Status menu” on page 55
<b>Status</b> ⇒System load...	Show the CPU load for the nodes in a Scali System, see “System monitoring - Status menu” on page 55

**Table 5-2:** Menu selections always available

When running in system administrator mode, the following additional menu options are available:

<b>Management → Console</b>	Open a console for the selected node(s), see "Power and Console control - Management menu" on page 52
<b>Management → Reboot</b>	Reboot the selected node(s), see "Power and Console control - Management menu" on page 52
<b>Management → Shutdown</b>	Shutdown the selected node(s), see "Power and Console control - Management menu" on page 52
<b>Management → Power On/Off/Cycle</b>	Turn power on/off/cycle for selected node(s), see "Power and Console control - Management menu" on page 52
<b>Interconnect → Set Routing Default/Force XY/Force YX</b>	Select routing for interconnect, see "High speed Interconnect configuration - Interconnect menu" on page 53
<b>Interconnect → Auto Routing On/Off</b>	Enable/Disable automatic rerouting, see "High speed Interconnect configuration - Interconnect menu" on page 53
<b>Software → Install...</b>	Install software at multiple nodes, see "Software installation - Software menu" on page 57
<b>Software → Configure...</b>	Configure software installation, see "Software installation - Software menu" on page 57

**Table 5-3:** Additional menu selections in administrator mode

### 5.3.1 Executing programs - Run menu

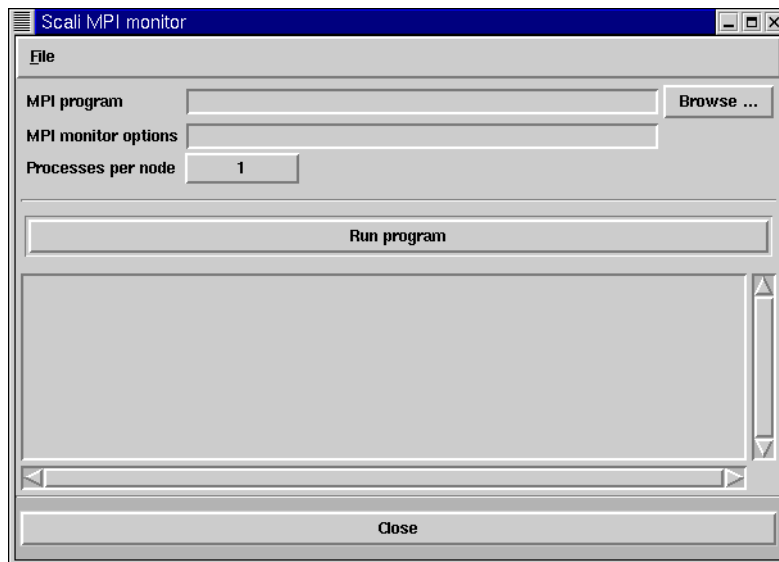
The Run menu is used for execution of programs at multiple nodes in a Scali System. The table below gives a summary of the Run menu options, the **MPI Program...** and **Parallel Shell...** options are described in more detail later in this section..

<b>Run→MPI Program...</b>	Run MPI programs at selected nodes.
<b>Run→Parallel Shell...</b>	Run shell commands in parallel at selected nodes.
<b>Run→Terminal</b>	Run X terminals at selected nodes.
<b>Run→Terminal with command...</b>	Run X terminals with a given command at selected nodes.

**Table 5-4:** Run menu options

#### 5.3.1.1 Executing MPI programs

From the Scali MPI monitor window, execution of MPI programs at selected nodes may be controlled. To invoke the Scali MPI monitor window, select **Run→MPI Program...** at the main window.

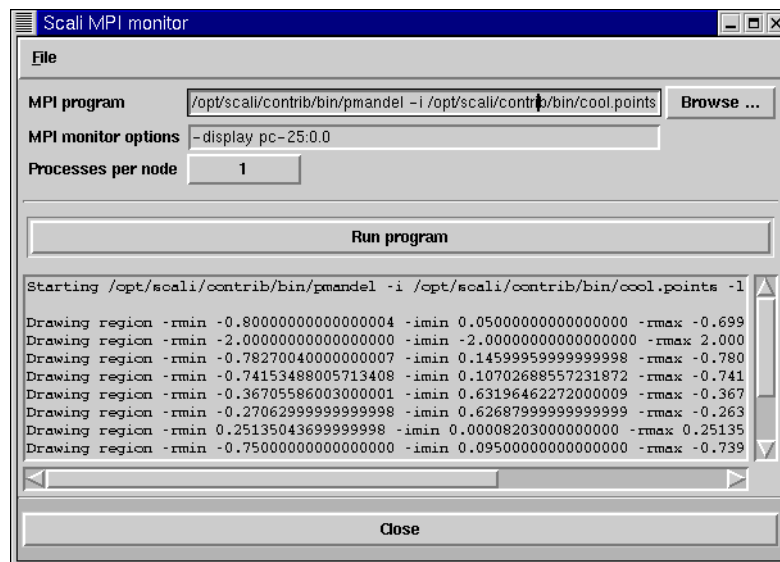


**Figure 5-8:** Scali MPI monitor window.



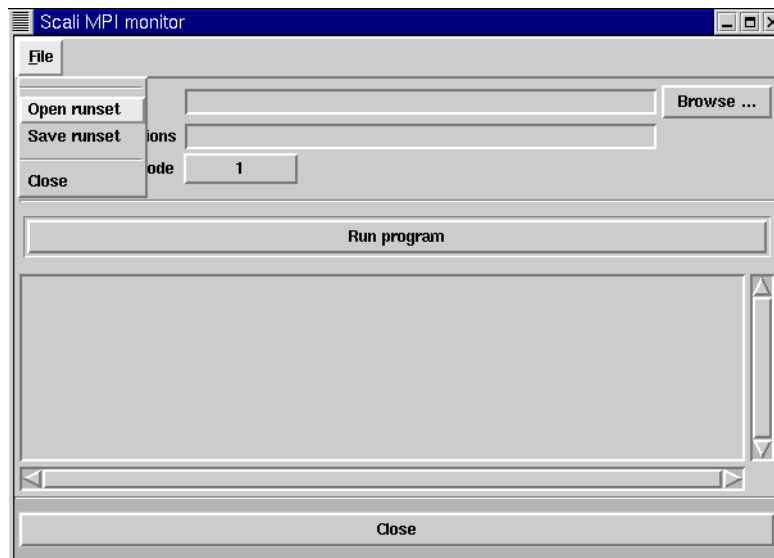
From this window a MPI program may be started or stopped. The following options are available:

- **Run→MPI Program□ MPI Program =?**  
Specify MPI program to be started. Any options to the program is also specified here. The Browse... button may be used for selecting program.
- **Run→MPI Program□ MPI monitor options =?**  
Specify options for `mpimon`, see ScaMPI User's Guide for more information.
- **Run→MPI Program□ Processes per node =?**  
The number of instances of the program to be started on each node.
- **Run→MPI Program□ Run program**  
Start the selected program by pressing this button. The program will be started at the nodes selected in the main desktop window. Output to "stdout" and "stderr" is printed in the centre window.  
*Note: only programs linked with ScaMPI may be run, other MPI implementations have different start-up procedures.*
- **Run→MPI Program□ Stop program**  
By pressing this button while a program is running the program will be aborted. All instances of the program at all nodes will be stopped. Only MPI programs started from the Scali MPI monitor window may be stopped.



**Figure 5-9:** Scali MPI monitor window with output from an MPI program.

From the File menu in Scali MPI monitor you may open and save runsets.



**Figure 5-10:** The Scali MPI monitor window with runset selection.

A runset is the collection of information needed to run an MPI program: program name, program options and MPI Monitor options. A runset can be saved to make it easier to re-run a program since you do not have to re-type the program name, program options, MPI Monitor options etc. Runsets are usually stored together with the MPI program or in your home directory.

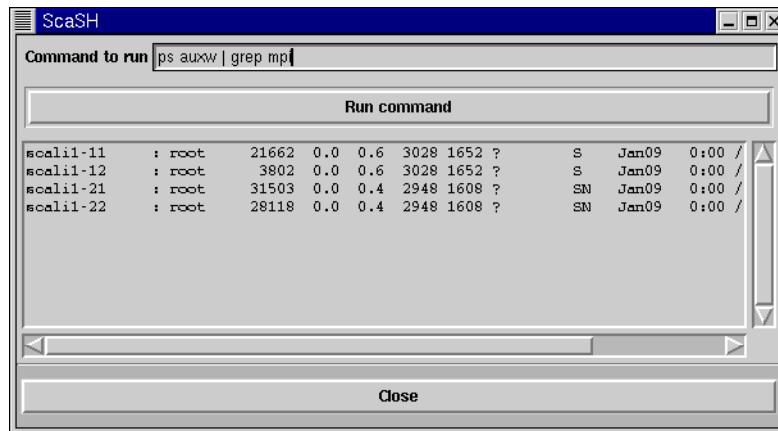
- **Run→MPI Program□File→Open runset**  
Open a stored runset
- **Run→MPI Program□File→Save runset**  
Save the current runset

Note, execution information, i.e. processes per node and node selection in the Scali Desktop main window, is not stored.

### 5.3.1.2 Running Parallel Shell commands

- **Run→Parallel Shell..**

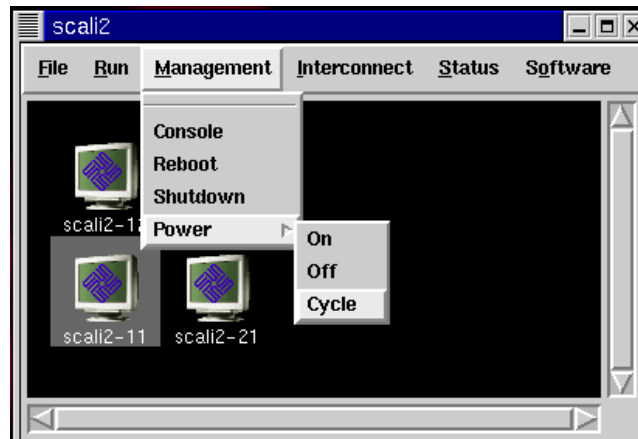
The parallel shell window allows you to run shell commands on selected nodes in parallel. The following example shows a **ps** command run on 4 nodes in parallel.



**Figure 5-11:** Example of a Parallel shell command

### 5.3.2 Power and Console control - Management menu

The Management menu is used for controlling power and console of a Scali System.



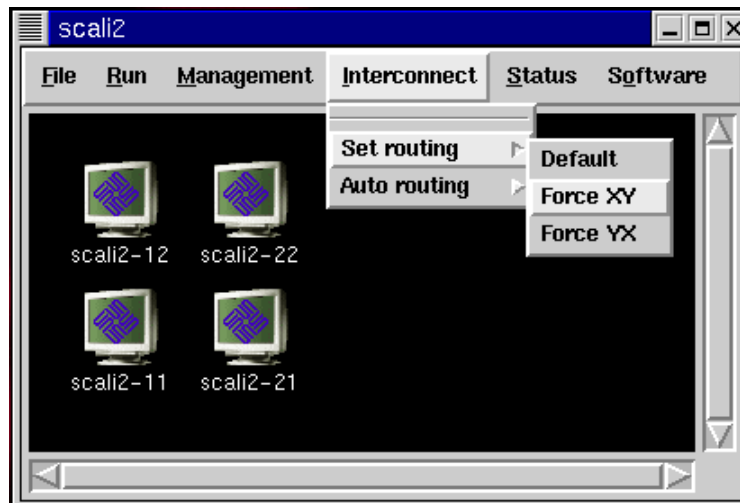
**Figure 5-12:** The Management menu

The following options are available in the Management menu.

- **Management → Console**  
By selecting Console, an xterm is started with a console connection to the selected nodes. Note that the console connection requires special hardware, e.g. a serial port switch, connected to the serial port of the nodes.
- **Management → Reboot**  
Initiates software reboot at all selected nodes.
- **Management → Shutdown**  
Initiates software shutdown at all selected nodes.
- **Management → Power**  
Control power on, power off, or power cycle at the selected nodes. Note that the power control requires special hardware, e.g. a power switch.

### 5.3.3 High speed Interconnect configuration - Interconnect menu

The Interconnect menu lets the administrator set routing for the high speed SCI network.



**Figure 5-13:** Interconnect menu

The Interconnect menu has the following option:

- **Interconnect → Set routing**  
The Set routing option is only available if the ScaConf module is installed. Using this menu, the routing of the high speed SCI interconnect may be modified to omit faulty nodes in the network.
- **Interconnect → Auto routing**  
This selection enables/disables automatic rerouting.

ScaConf provides two different approaches to routing:

- *Scali routing (Named Default in the menu for SSP 2.0, Scali in newer releases)*  
Scali routing is a fault tolerant routing algorithm, capable of maintaining full connectivity between the remaining nodes even if more than one node have failed. When all nodes are working, the Scali-algorithm is equal to dimensional routing (XY or YX).

- *XY and YX routing*  
XY and YX routing is ordinary dimensional routing which means that all motion in the first dimension must be done before routing in the next dimension. When using XY routing, packets are routed in the X (horizontal) dimension first, then in the Y (vertical) dimension. With YX routing, packets are routed in the Y dimension first, then in the X dimension.  
XY routing is used as default routing for ScaConf.

*Fault tolerance of Scali routing vs. dimensional routing algorithms*

In the case of a node failure, this will break the two SCI-rings that it is connected to. This in turn means that the other nodes on these two broken (failed) rings will only be connected to one operational SCI-ring. Still, the Scali routing algorithm will be able to connect all remaining nodes, as long as the node is connected to at least one working SCI-ring. In comparison a dimensional routing algorithm will not be able to connect any of the nodes on the broken rings. It will only be able to maintain connectivity between nodes where both horizontal and vertical ring are working.

There may also arise situations where all nodes are ok but an SCI-ring is down. (An SCI-ring is considered down if one of the links in the ring has failed). Hence if a link controller on an SCI card reports a link as "DOWN" to ScaConf this will cause the entire ring to be considered as down. Again the Scali routing algorithm will be able to connect all nodes, whereas a dimensional routing algorithm will not be able to connect the nodes on faulty rings.

### 5.3.4 System monitoring - Status menu

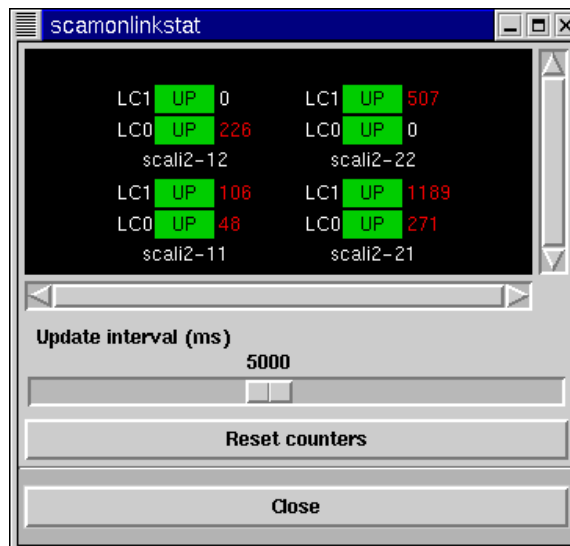
The System Monitoring menu is used for monitoring node resources. Table 5-5 on page 55 gives a summary of the monitoring menu options. The **Link status...** and **System load...** options are described in more detail later in this section, while **Process list...**, **User list...** and **Uptime and Load...** are only shown in the table.

<b>Status</b> ⇒ <b>Process list...</b>	Show processes at selected nodes.
<b>Status</b> ⇒ <b>User list...</b>	List users at selected nodes.
<b>Status</b> ⇒ <b>Uptime &amp; load</b>	Show uptime and load at selected nodes (rup command).
<b>Status</b> ⇒ <b>Link status...</b>	Show link status (see below).
<b>Status</b> ⇒ <b>System load...</b>	Show the CPU load for the nodes in a Scali System (see below).

**Table 5-5:** System Monitoring options

- **Status**⇒**Link status...**

The link status window shows the status of the high speed SCI network:

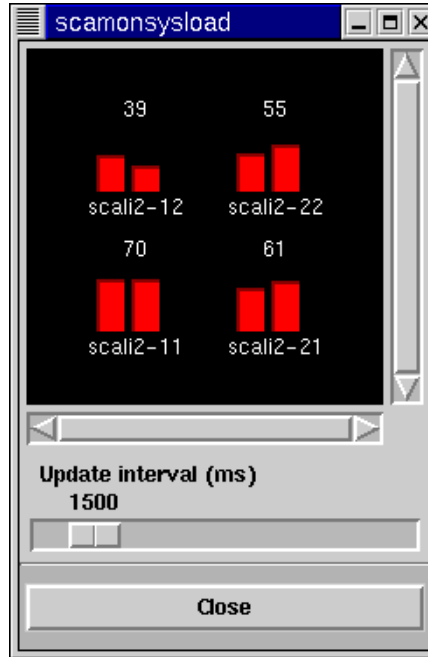


**Figure 5-14:** High speed SCI network link status

Each link controller is shown with state (UP or DOWN), and the Link controllers error counter.

- **Status=System load...**

The system load window shows the load of each CPU in the Scali System:



**Figure 5-15:** CPU Load for nodes in a Scali System

For each node in the Scali System, the overall CPU load is displayed as a numeric value, and the load for each processor is displayed as a bar.



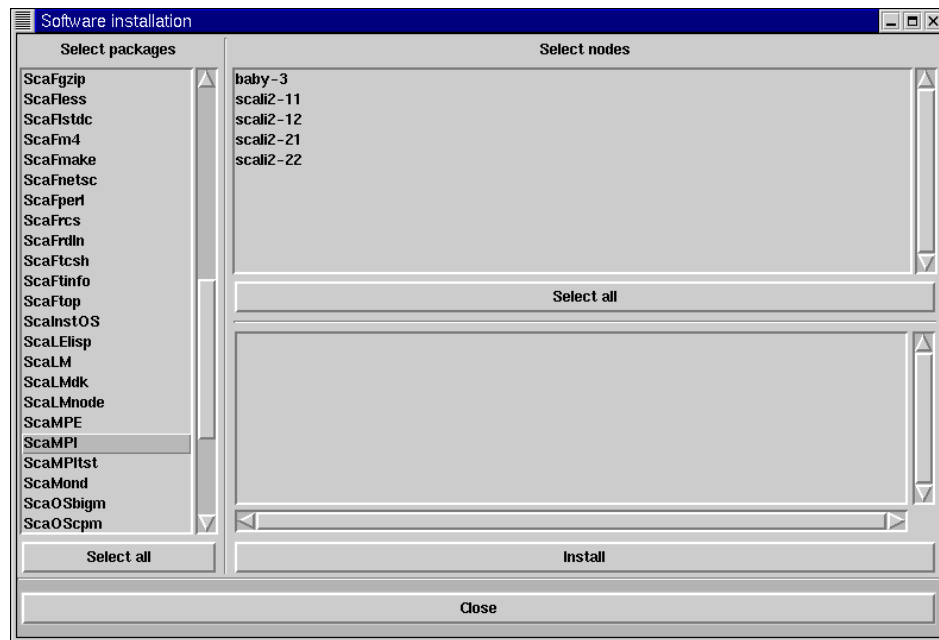
### 5.3.5 Software installation - Software menu

To install software packages at multiple nodes in a Scali System, the system administrator may use options in the Software menu to simplify the operation. The table 5-6 gives a summary of the Software menu options.

<b>Software⇒Install...</b>	Instal software at multiple nodes
<b>Software⇒Configure⇒Edit Package Configuration</b>	Edit the package categories
<b>Software⇒Install⇒Edit Node Configuration</b>	Edit the node categories

**Table 5-6:** Software menu options

- **Software⇒Install....**



**Figure 5-16:** Software installation window

The **Select packages** field at the left edge of the window lists possible packages to install on a Scali System. Multiple packages may be selected by pressing **<Ctrl>** and left mouse button. All packages may be selected with the **Select All** button.

Similarly, the **Select nodes** field specifies which nodes to participate in the software installation.

Installation is started with the **Install** button.

Nodes in a Scali System are grouped into categories describing the use of the node. Similarly software packages are grouped into the same categories describing what kind of nodes the packages are meant to be installed at. The following table describes the defined categories.:

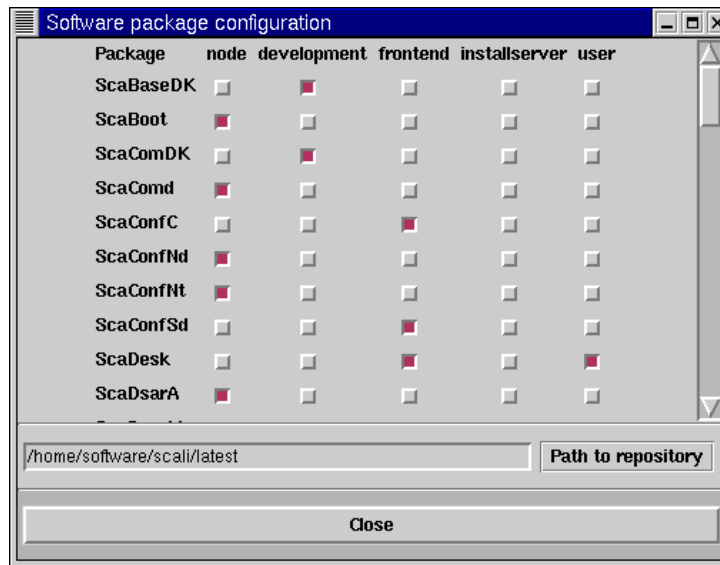
Category	Description
node	processing node, basic software
development	compilers, libraries, etc.
frontend	license managers, configuration tools, monitoring servers etc.
installserver	OS installation frontend
user	user node, limited number of software packages shall be installed

**Table 5-7:** Node and software package categories

By using this scheme, software can be installed or updated for the entire Scali System in a single operation. A software package will only be installed on nodes defined in a corresponding category.

- **Software→Configure→Edit Package Configuration**

This menu option will bring up a package configuration window:

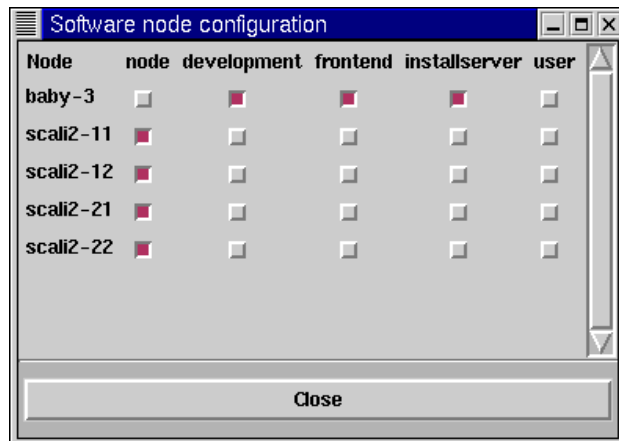


**Figure 5-17:** Software Package Configuration window

Here categories for each Software package may be selected, note that a software package may belong to multiple categories. The **Path to repository** gives the location of where the software packages can be found.

- **Software→Configure→Edit Node Configuration**

This menu option will bring up the node configuration window.



**Figure 5-18:** Node category window

Here categories for each node may be selected. Note that a node may belong to multiple categories.

## Chapter 6 ScaSH - parallel shell tools

---

The ScaSH parallel shell tool suite is a collection of scripts enabling the execution of commands in parallel on a large number of target nodes:

- **scash** : Execute command on nodes in a Scali cluster.
- **scahosts** : Checks nodes for availability.
- **scacp** : Copies file(s) locally on nodes in a Scali cluster.
- **scarcp** : Copies file(s) from/to local machine to/from nodes in a Scali cluster.
- **scaps** : Prints processes on nodes in a Scali cluster.
- **scakill** : Kill processes on nodes in a Scali cluster.
- **scarup** : Prints status on nodes in a Scali cluster.
- **scawho** : Prints user names and number of processes on nodes in a Scali cluster.

The file `/opt/scali/etc/scash.conf` contains configuration parameters to the ScaSH suite of tools. It will be sourced by the tools and must adhere to bourne shell script syntax. Table 6-1 on page 66 describes the options in the configuration file and lists their default values. All programs in the ScaSH suite will use the nodes reported by the **scahosts** program unless nodenames are specified on the command line.

### 6.1 scash - the parallel shell command

The **scash** utility is a UNIX script that can run the same shell command on a set of Scali system nodes. The target nodes may be specified in a configuration file or on the command line (see description of **scahosts** program in 6.2). The command options are listed below.

Usage:

```
scash [options] ([-n "nodenames"] <command>)|(-c "command" [<nodenames>])
```

[options]:

- |           |  |
|-----------|--|
| <b>-?</b> | Print help text  |
| <b>-h</b> | Print help text  |
| <b>-v</b> | Print machine name and command before each output block  |
| <b>-V</b> | Print version  |
| <b>-p</b> | Print machine name before each line in each output block |
| <b>-d</b> | Dryrun: only print command, do not execute               |
| <b>-b</b> | Run in background  |
| <b>-w</b> | Wait until command has finished on all hosts             |

- a** Use 'at now' to execute command
- s** Which remote execution command to use
- r key** Replace key with machinename if key is found in command
- l** Limit on number of parallel processes to run in background mode
- n "n1 n2 .."** Nodenames separated by space characters
- f nodefile** Use node names from the file *nodefile*.
- z** Do NOT check access to nodes (the default is to ping and test remote shell access)
- c "cmd"** The shell command to be run

## 6.2 scahosts - nodes availability check

The **scahosts** program located in `/opt/scali/bin`, will check a number of hosts for availability. An available host is one that answers to a ping request and is open for remote shell access. The program will print the name of the available hosts. Which hosts the program will check may be specified in several ways. One may specify hosts on the command line when running the program, either with the **-f** option which gives the path to a file containing host names or with a list of hosts at the end of the command. If neither the **-f** option nor the hostlist is present, **scahosts** will look for hosts in the file `$HOME/.scahosts`. If this file is not available, it will look for hosts in the file `/opt/scali/etc/.scahosts`. The file containing hostnames should have one name per line. The option `RSH_TESTACCESS` in the configuration file `scash.conf` (see table 6-1) may be set to disable checking for remote shell access to validate a node.

Usage:

```
scahosts [options] [host ...]
```

[options]:

- f nodefile** Use node names from the file: *nodefile*
- v** Print verbose information
- z** Don't check access to nodes
- V** Print version
- ?/-h** Print help message.

## 6.3 scacp - local file copy

The **scacp** program copies file(s) locally on nodes in a Scali cluster

Usage:

```
scacp [options] <from> [<from>] <to>
```

[options]:

- ?** Print help text
- h** Print help text
- R** Copy recursively
- v** Print machine name and command before each output block
- V** Print version
- p** Print machine name before each line in each output block
- d** Dryrun: only print command, do not execute
- b** Run in background
- w** Wait until command has finished on all hosts
- n "n1 n2 .."** Nodenames separated by space characters
- f nodefile** Use node names from file: *nodefile*
- z** Do NOT check access to nodes (the default is to ping and test remote shell access)

## 6.4 scarcp - remote file copy

The program **scarcp** copies file(s) between local machine and nodes in a Scali cluster. One may specify the command to be used when copying files with the **-c** option.

Usage:

```
scarcp [options] <from> [<to>]
```

[options]:

- ?** Print this text
- h** Print this text
- g** Copy from remote nodes to local node (default is local to remote) (use in combination with **-r** option to avoid overwriting file)
- R** Copy recursively
- v** Print machine name and command before each output block
- V** Print version
- p** Print machine name before each line in each output block
- d** Dryrun: only print command, do not execute
- b** Run in background
- w** Wait until command has finished on all hosts
- r key** Replace key with machinename if key is found in command
- c "cmd"** The copy command to use
- n "n1 n2 .."** Nodenames separated by space characters
- f nodefile** Use node names from file: *nodefile*
- z** Do NOT check access to nodes (the default is to ping and test remote shell access)

## 6.5 scaps - process information

The command **scaps** prints processes on nodes in a Scali cluster

Usage:

```
scaps [options]
```

[options]:

```
-?          Print help text
-h          Print help text
-u user    Only list those processes that match the given username
-s string  Only list those processes that contains the given string
-v          Print machine name and command before each output block
-V          Print version
-p          Print machine name before each line in each output block
-d          Dryrun: only print command, do not execute
-b          Run in background
-w          Wait until command has finished on all hosts
-n "n1 n2 ..." Nodenames separated by space characters
-f nodefile Use node names from file: nodefile
-z          Do NOT check access to nodes (the default is to ping and test remote shell access)
```

## 6.6 scakill - parallel kill command

The **scakill** program allow you to kill or send a specific signal to processes running on nodes in a Scali cluster.

Usage:

```
scakill [options]
```

[options]:

```
-?          Print help text
-h          Print help text
-v          Print machine name and command before each output block
-V          Print version
-p          Print machine name before each line in each output block
-d          Dryrun: only print command, do not execute
-b          Run in background
-w          Wait until command has finished on all hosts
-n "n1 n2 ..." Nodenames separated by space characters
```



- f nodefile** Use node names from file: *nodefile*
- z** Do NOT check access to nodes (the default is to ping and test rsh access)

One or more of the options following *must* be specified:

- i startpid-stopid** Kill those processes that lie within the given process id range
- u user** Kill those processes that match the given username
- s txt** Kill those processes that match the given string
- l level/name** Kill level or name (e.g. 9, HUP, ...)
- level** Kill level (only numeric value allowed)

:

## 6.7 scarup - node status

The **scarup** command prints status on nodes in a cluster.

Usage:

```
scarup [options]
```

[options]:

- ?** Print help text
- h** Print help text
- v** Print machine name and command before each output block
- V** Print version
- p** Print machine name before each line in each output block
- d** Dryrun: only print command, do not execute
- b** Run in background
- w** Wait until command has finished on all hosts
- n "n1 n2 ..."** Nodenames separated by space characters
- f nodefile** Use node names from file: *nodefile*
- z** Do NOT check access to nodes (the default is to ping and test rsh access)

## 6.8 scawho - user information

The **scawho** program prints usernames and number of processes on nodes in a Scali cluster.

Usage:

```
scawho [options]
```

[options]:

```
-?          Print help text
-h          Print help text
-v          Print machine name and command before each output block
-V          Print version
-p          Print machine name before each line in each output block
-d          Dryrun: only print command, do not execute
-b          Run in background
-w          Wait until command has finished on all hosts
-n "n1 n2 ..." Nodenames separated by space characters
-f nodefile Use node names from file: nodefile
-z          Do NOT check access to nodes (the default is to ping and test remote shell access)
```

## 6.9 ScaSH configuration file

The file `/opt/scali/etc/ScaSH.conf` contains configuration parameters for the ScaSH parallel shell tools suite. It will be sourced by the tools and must adhere to bourne shell script syntax. Table 6-1 describes the options in the configuration file and lists their default values.

Option	Default value	Description
RSH_CMD	rsh -n	Which program should be used to execute a command remotely. The program specified must support the following calling convention: <code>\$RSH_CMD &lt;host&gt; &lt;command&gt;</code>
RCP_CMD	rcp -p	Which program should be used to copy files from/to a remote node. The program specified must support the following calling convention: <code>\$RCP_CMD &lt;host&gt;:&lt;fromfile&gt; &lt;tofile&gt;</code> or <code>\$RCP_CMD &lt;fromfile&gt; &lt;host&gt;:&lt;tofile&gt;</code>

**Table 6-1:** Options in the file ScaSH.conf

Option	Default value	Description
<b>LCP_CMD</b>	<code>cp -p</code>	Which program should be used to copy files on the local node. The program specified must support the following calling convention: <code>\$LCP_CMD &lt;fromfile&gt; &lt;tofile&gt;</code>
<b>PARLIMIT</b>	<code>expr \`ulimit - n\` / 3`</code>	When using the background option with one of the ScaSH tools, this parameter will control the number of commands that are actually run in parallel. ScaSH tools will wait until the below number of processes are finished before issuing the next batch.
<b>RATESLEEP</b>	2	Heavy use of rsh/rcp can lead the inet daemon to enter a non responsive state. This is related to the fact that rsh uses socket ports below the 1024 limit and exhausts the number of free ports OR that the number of successive connections to the shell service are too many to be handled. Both problems arise because the sockets enter a <code>TIME_WAIT</code> state which uses ~120 seconds to terminate. To limit the number of successive rsh accesses to each host use <b>RATESLEEP</b> to adjust interval between each access to that node. (If you are using ssh or other remote execution programs which do not have the above mentioned problem set the value of <b>RATESLEEP</b> to 0).
<b>RSH_TESTACCESS</b>	<code>true</code>	Define if the <code>scahosts</code> program should test the nodes for remote shell access to verify access to nodes. The default is true.

**Table 6-1:** Options in the file ScaSH.conf



## Chapter 7 Scali system configuration

---

The system administrator can manage a Scali system with Scali system configuration software - ScaConf. ScaConf can query the status of the interconnect hardware and software, set routing algorithms, bypass troublesome nodes, connect or disconnect nodes, switch power on/off and access the console. ScaConf is necessary on all Scali systems but the ordinary user should never need to know about the details in this chapter. ScaConf is by default configured to manage a cluster automatically without user interference. Failed nodes or links is automatically detected and the cluster is rerouted automatically. ScaConf is accessible from the ScaDesktop GUI (see “The Scali Desktop GUI” on page 39) or through the command line tool: **scaconftool** which will be described in “ScaConf - command line interface” on page 70.

### 7.1 The configuration server

The configuration server is available in the ScaConfSd package. The server must run on the frontend and will start automatically when installed and when the frontend is re-booted. The server can be stopped, started or restarted with the `/opt/scali/init.d/scaconfsd` script on the frontend:

```
frontend# /opt/scali/init.d/scaconfsd [start|stop|restart]
```

It may also be started directly from the binary located in `/opt/scali/sbin/scaconfsd`. The latter assumes that the environment variable `LM_LICENSE_FILE` is set correctly.

When the server starts, it will read a configuration-file to get the name of all nodes and the configuration of the machine. Default configuration file is `/opt/scali/scaConf.conf`. Another file may be specified with the `-f` option. The `-f` option is only available if the server is started from `/opt/scali/sbin/scaconfsd` binary. The server maintains a database with the current state of the system. The database contains (among others):

- The configuration of the machine, meaning which nodes are connected through which controllers.

For each node:

- Node name and SCI nodeID.
- Routing tables.
- Number of adapters and controllers in the system.
- Link status. This status can be either UP or DOWN indicating if the connection

- is ok or not.
- Status for each controller. A controller can be either enabled or disabled.
- Status for each node regarding the communication between configuration server and configuration daemon.
- Whether a SCI driver is available or not on each node.
- Routing partition number for each node.

When started, the server will poll every node for status information. If the `AUTO_REROUTE` server-option is enabled, it will reroute the cluster upon start and at every event that demands rerouting. Such events are node or link failure or change of SCI nodeID. For further information about server-options, read section 7.3.13.

## 7.2 The ScaConf daemons

At this point one should observe that to enable configuration through ScaConf, a node must run two daemons, namely the Scali communication daemon **sacomd** and Scali configuration daemon **saconfnd**. The first provides communication services and is shared by other Scali tools, the latter provides all ScaConf services. If either daemon is not running, the node is rendered useless for ScaConf. Hence the status of these daemons is very much a part of the total node status. For the configuration server to collect information from the nodes, a configuration daemon must run on each operational node in the machine. The configuration daemon will be started upon reboot on each node when installed. It may also be started manually with the `/opt/scali/init.d/saconfnd` script on each node or with the `daemon` command in **saconftool**.

## 7.3 ScaConf - command line interface

This is a description on how to use ScaConf with the ASCII based configuration tool, **saconftool**. This program provides a fast and efficient way to access the ScaConf functionality. Further information about **saconftool** may be found in the file `/opt/scali/doc/ScaConfC/saconf.users.instruction.txt`.

### 7.3.1 Starting saconftool

When installed, **saconftool** is located in the `sbin` directory of the Scali installation and can be started on the frontend by:

```
frontend%/opt/scali/sbin/saconftool
```

Starting the tool on another machine than the frontend:

```
bash% /opt/scali/sbin/saconftool -H <frontend>
```

If **scaconftool** fails to start or exits abruptly during run, chances are that it has lost the connection to the server. In that case, you need to check that the server is running, and if not, restart the server and start a new tool.

### 7.3.2 Using scaconftool

The following command line options are available with **scaconftool**:

Usage:

```

/opt/scali/sbin/scaconftool [Options]
[Options]:
-l d/e          enable (e) or disable (d) log from server
-e "command1;...;commandn;"  execute these commands in batch mode.
-h             help
-H <host>     set server host
-p <port>     set server port, (default is 32015)
-q            make scaconftool go as quiet as possible
-s <number>  tune the sleep constant, usually
              number is between 0 and 1, default 0.7
-V            print version
-?           help

```

When the tool has started, it will prompt you with **ConfTool>** indicating that it is ready to accept commands. An online reference to commands and their syntax can be found in the file `/opt/scali/doc/ScaConfC/scaconftool.txt`. Command line editing may be performed in **readline** style, where the arrow-key gives access to a history of previous commands.

#### 7.3.2.1 scaconftool - node selection

Some of the commands allow you to perform a command on a set of nodes. This may be one node, all nodes or only some of the nodes. There are five different ways to state a list of nodes. The *list* command will be used as an example to illustrate these possibilities:

- **Explicit.**  
You may explicitly specify the node names on the command line partitioned by one or more blanks. Example: *list node1 node3 node14* will list information about node1, node3 and node14.
- **All nodes.**  
The keyword 'all' is an alias for all nodes in the system. The command *list all* will apply to all nodes known to the configuration server. It is equivalent to typing *list node1 node2 ... noden* but more time saving.

- By node state.  
It is possible to select a set of all nodes that is in one particular state. Use the 'list status all' command to see which nodes are in what state. *list NO\_SCI\_DEV* will list information on all nodes with status *NO\_SCI\_DEV*.
- Built-in list.  
scaconftool maintains an internal list of nodes. This list is empty when the tool is started. To manipulate the internal list of nodes, use the *select* and *unselect* command. If the node list on the command-line is left empty, the nodes in the internal list are used. *list* with no parameters will list information about nodes in the internal list. To get a list of all nodes except those in state *OK*, type *select all* followed by *unselect OK*. *list* will now list all nodes except those in state *OK*.
- By routing partition number. One may select all nodes part of a particular routing partition. '*list status #1*' will show status for all nodes in partition 1. Partition number is prefixed by #.

#### 7.3.2.2 scaconftool - verbose mode

You may get some extended feedback from the server by enabling the *log* parameter. This is done with the *log enable* command. If log is enabled the tool will print a message, each time it receive some information from the server. Log is enabled by default. It may be turned off with *log disable*. You may enable or disable log from start with the **-l** option to **scaconftool** at start-up time.

#### 7.3.2.3 scaconftool - truncation

The commands and some of the keywords are also recognized if you truncate them, as long as enough of the command is present to be unambiguous. Node states, node names and name of routing algorithms may not be truncated. *lis no st a* is equal to typing *list nodeid status all*.

#### 7.3.2.4 ScaConf user modes.

The configuration tool is available in two different modes, one user mode and one administrative (admin) mode. Admin mode is available to users with administrative rights (i.e. user id = 0). User identity is checked upon start and the mode for the configuration tool is set according to this information. Admin mode gives full access to ScaConf, user mode only gives access to status information.

- User mode.  
To ordinary users only a restricted part of the commands are available. Ordinary users may use *getopt*, *list*, *log*, *reconnect*, *select*, *status*, *unselect*, *update*, *version*, *help* and *quit*.
- Admin mode.  
Admin mode is only available to root, and gives full access to ScaConf operations



like rerouting, starting/stopping daemons etc.

### 7.3.2.5 scaconftool - batch mode

To run **scaconftool** in batch mode, use the `-e` option when starting the tool. The commands you want **scaconftool** to run must be enclosed by `" "` and separated by `;`. The following batch-job will restart the configuration daemon on nodes in state `NO_DAEMON`:

```
frontend% scaconftool -e "update all;daemon scaconfnd restart NO_DAEMON"
```

**scaconftool** will first execute `update all` causing the database to be updated with new information for all nodes. Then `daemon scaconfnd restart NO_DAEMON` will launch a **scash** job to restart **scaconfnd** on all nodes in state `NO_DAEMON`.

### 7.3.3 scaconftool - status check of nodes and daemons

Normally the first thing you want to do is to check the status of all the nodes in your system, normally with the `list` or `status` commands. Details of their usage can be found by using the `help` command: i.e. `help list` or `help status` or in the file: `/opt/scali/doc/ScaConf/scaconftool.txt`. (Details about all **scaconftool** commands may be found there). Here is an example using `list` with the options `nodeid status all` which will show SCI node identification and status for all nodes:

```
ConfTool> list nodeid status all
Name      NodeId  Status
scali2-24 0x2400  OK
scali2-23 0x2300  OK
scali2-22 0x2200  OK
scali2-21 0x2100  OK
scali2-14 0x1400  OK
scali2-13 0x1300  OK
scali2-12 0x1200  OK
scali2-11 0x1100  UNREACHABLE
```

Status may be one of the states listed in Table 7-1. If some nodes are not OK, you

Node state	Description	How to resolve
OK	SCI-driver is loaded, and communication with daemons on this node is ok	This is the state of a healthy node and need not be resolved.
NO_SCI_DEV	The SCI-driver on the node is not loaded	You can reload the driver with the <i>reload</i> command, or use the <i>fix</i> command.
NO_DAEMON	The configuration server has lost contact with the node daemon <b>scaconfnd</b> on the node. This usually indicates that the daemon is not running or not connected to the server.	You may use the <i>fix</i> command to resolve this state, or do the following: *type <i>reconnect NO_DAEMON</i> . *If state is still <i>NO_DAEMON</i> for one or more nodes, try restarting the configuration node daemon <i>scaconfnd</i> on these nodes with the <i>daemon</i> command *reconnect the same nodes with the <i>reconnect</i> command.
UNREACHABLE	The configuration server has lost (or never gained) contact with this node. This may indicate that the communication daemon ( <i>sacomd</i> ) on this node is not running or that the node itself is erroneous.	If it is a communication failure, this will be resolved with the <i>fix &lt;node&gt;</i> command. If the node is still <i>UNREACHABLE</i> after running the <i>fix</i> -command, the node most likely has problems beyond reach of <i>scaconftool</i> .
UNKNOWN	The configuration server has not been able to determine the state of this node.	You may try the <i>update</i> command or the <i>restart</i> command to resolve this state.

**Table 7-1:** ScaConf node states

should investigate further. It might be that some daemons are not running on some nodes. Try running the *fix* command. See below for an explanation on how to use the *fix* command.

#### 7.3.4 The *fix* command

The *fix* command will try to re-establish connection with the nodes specified on the command line that are currently in an other state than OK. The following algorithm is used:

- For all of the nodes (specified on the command line) that are in state UNREACHABLE:

```

*Attempt to reconnect nodes.
*For all nodes still in state UNREACHABLE:
    *If the node answer to ping, restart the communication daemon.
    *Wait for some time.
    *Attempt to reconnect the nodes again.
Then, for all of the nodes (specified on the command line) that now in state
NO_DAEMON:
    *Attempt to reconnect nodes.
    *For all nodes still in state NO_DAEMON: Restart configuration daemon
    on the node.
    *Wait for some time.
    *Attempt to reconnect the nodes again.
• Then, for all of the nodes (specified on the command line) that are in state
NO_SCI_DEV:

    *Attempt to reconnect nodes.
    *For all nodes still in state NO_SCI_DEV: Reload SCI-driver on the
    node.
    *Wait for some time.
    *Attempt to reconnect the node.

```

It may be necessary to run the *fix* command maybe more than once, if the process of restarting daemons take long time due to heavy load on the system. When a daemon is started (this is done with the **scash** functionality) **scaconftool** will wait for a given amount of time. By default this is (Number\_of\_nodes \* WaitConst) seconds. WaitConst is 0.7 by default, causing a system with 8 nodes to wait for approx. 6 seconds before it will try to reconnect a node. WaitConst may be tuned when **scaconftool** is started with the **-s** option:

```
frontend> scaconftool -s0.5
```

On a machine with 8 nodes will cause the tool to wait 4 seconds between restart of a daemon on a node and an attempt to reconnect the node.

### 7.3.5 Setting SCI nodeID

The *nodeid* command will set the SCI node identification for a node in the SCI-hardware. The nodeID normally should be set once for each node at the beginning of a machines lifetime. It may also be set later if the cluster is reconfigured, or if a SCI card is replaced. With no parameters, the configuration server will set the nodeID to default as coded in the file `/opt/scali/etc/ScaConf.nodeidmap`. If this file is not available and name of nodes are on the format *nodeprefix-XY*, nodeID is computed from the name. X and Y indicate the x and y coordinates for the node in the torus. For configurations where coordinates greater than 9 are needed, this algorithm is not

usable. If neither the `nodeidmap` file is available, nor the node name algorithm is useable, `nodeID` is generated sequentially starting on 0x1100 for the first node listed in the configuration file `/opt/scali/etc/ScaConf.conf`.

One may also specify a node and a `nodeID` on the command line, but make sure this `nodeID` is a legal one. The `nodeID` must be unique, no two nodes in the same cluster may have the same `nodeID` and the `nodeID` should be in the range 0x1100 to 0xff00, and the two least significant bytes must always be 00. Do not use this command if you do not know what you are doing. To keep the number of writes to a minimum, a new `nodeID` is only written if it is different from the `nodeID` already set. The `nodeid` command is only available to root.

### 7.3.6 Reloading SCI-driver

The SCI-driver is loaded upon reboot of a node. If there arises a situations where the SCI-driver must be reloaded, this can be done with the `reload` command. All programs blocking the SCI device must be terminated before the SCI-driver may be reloaded successfully. The `reload` command is only available to root. Reloading the SCI-driver on a node will clear the routing table on that node. One needs to reroute the cluster if the SCI-driver is reloaded on one of the nodes.

### 7.3.7 Powerswitching with scaconftool

You may use the `power` command to turn power on, off or on/off if the system is installed with a power switch. The power switch configuration is read from the file `/opt/scali/etc/ScaConf.conf`.

### 7.3.8 Access to console on one node

The `console` command may be used to connect to the console on one node. This command will start a telnet session, where one can log on to the console on a node specified on the command line. The console command may fail if someone else has access to the console. To exit from the telnet session, you must type the escape character sequence `Ctrl-a`, and then type `quit` when the telnet prompt (`telnet>`) appear.

### 7.3.9 On-line restart of the configuration server

The configuration server may be restarted on-line. This is done with the `restart` command. This will cause the server to reread the configuration file, build a new database, and contact the configuration daemon on each node for new information.

### 7.3.10 Failing nodes

One may fail a node, regardless of whether the node is alive or not. This is done with the *fail* command. This is a way of forcing a node into the *UNREACHABLE* state, and it is of limited interest for the ordinary user. Failing a node only implies a change of state in the configuration server database, and it will not affect the node, apart from the fact that this node will be considered failed if the cluster is rerouted. The *fail* command is useful for generating and setting routing patterns that normally only will occur if the node actually has failed. A node failed with the *fail* command, may be brought back to its correct state with the *reconnect* command or *restart* command. In order for the fail command to take effect, the *AUTO\_REPAIR* option in the server must be turned off before the node is failed.

### 7.3.11 Getting updated information

The *update* command may be used to poll each node for updated information. The configuration server will query each node for the most recent information and update its database. For each query about states in the cluster scaconftool will request information from the server and display it to the user.

### 7.3.12 Daemon control with scaconftool

The two daemons scaconfnd and scacomd located in the directory `/opt/scali/sbin/` on the nodes may be started, stopped or restarted on all nodes from the tool. This is done with the *daemon* command. If you want to stop and start the communication daemon on all nodes, type this in the tool:

```
ConfTool>daemon scacomd restart all
```

If you want to stop and start the configuration daemon on all nodes in state *NO\_DAEMON* (no configuration daemon), you type this in the tool:

```
ConfTool>daemon scaconfnd restart NO_DAEMON
```

The daemon command uses scash to access all nodes, and the ScaSH package must be installed on the machine running the tool in order for the daemon command to work from the tool. Starting and stopping daemons from the tool is only available in admin-mode.

### 7.3.13 Server options.

One may get or set options in the server from **scaconftool**. The *getopt* command with no arguments will list all the available options and their value. The available options and their default value are specified in table Table 7-3 on page 83. The server options *SAVE* and *RESTORE* are in fact commands to the server. These options do not have a value and when set, they will cause the server to perform an action, rather than setting a parameter. The command *setopt SAVE* will save all server options to the file `/opt/`

`sca/i/etc/ScaConf.opt`. When the server is restarted this file will be read and all options will be set with the value from this file. The *RESTORE* option, will set the value of all options in the server back to its factory default.

## 7.4 Routing

In order for the nodes to communicate over the SCI-network, there must be a routing path between the nodes. Which route a packet shall use is decided by a routing algorithm. Scali support different routing algorithms for different topologies, which will be described here. For the ordinary user it is recommended to use the *reroute* command in `scaconftool` without specifying routing algorithm. Then the system will use the routing algorithm recommended by Scali.

When the machine is re-booted, routing is set automatically by the server unless automatic rerouting is on. One may turn automatic reroute on and off with the *AUTO\_REROUTE* server option. Read section 7.3.13 for further details on how to set and read configuration server options. The machine may also be routed manually via **scaconftool** with the *reroute* command. If automatic rerouting is enabled, the configuration server will reroute the machine whenever it discover a situation that demand a reroute action to be taken. Such a situation may be that one or more nodes or link is erroneous or the SCI nodeID has changed for a node.

### 7.4.1 Ring topology

If the machine is configured with all nodes on one SCI ring, only one routing algorithm is available. Ring topology is detected by the flag `topology TOP_RING` in configuration file `/opt/sca/i/etc/ScaConf.conf`. Use *reroute* command with no arguments to route on a single ring from **scaconftool**.

### 7.4.2 Torus Topology

Torus topology is detected by the *topology TOP\_TORUS\_2D* flag in configuration file `/opt/sca/i/etc/ScaConf.conf`. In a torus each node is connected to two SCI ringlets, one in each dimension of the torus. There may arise situations where one or more nodes or one or more links is erroneous. Nodes are considered erroneous by ScaConf if they are in a different state than *OK*. SCI-links are considered erroneous if they are reported as *DOWN*. You may inspect the status of links and nodes with the *list status c0 c1 all* command from **scaconftool**. Routing is computed according to the state of nodes and SCI-ringlets. A SCI-ringlet is considered faulty if one of the links in the ring has failed. This means that controllers reporting their links as *DOWN* will cause the entire ring to be considered broken. You may also check the status on links with the *status link* command. A ring is also considered erroneous if one or more of the nodes

part of this ring is erroneous. If one or more rings are erroneous, maxcy routing algorithm will be able to connect all nodes with at least one working ring, dimensional routing algorithm will be able to connect all but the nodes on the faulty rings.

### 7.4.3 Dimensional routing

In dimensional routing are all routing in one dimension completed, before packets are routed in the next dimension and so on. Dimensional routing is selected with the *reroute xy* or *reroute yx*. *xy* means that packets are routed in the x dimension first. x-dimension is equal to the horizontal dimension drawn for the *list configuration* command. This is also equal to the dimension connected through controller number 1 on the SCI adapter. *yx* means that packets are travelling in the vertical dimension first. Vertical refer to the printout from the *list configuration* command in *scaconftool*. The dimensional routing will not be able to connect any of the nodes on erroneous rings, and will only be able to connect nodes where both horizontal and vertical ring are working.

There is two different versions of dimensional routing. The one that is described so far, consider which nodes and links that are operational and which that is failed, and enables and disables links according to this. This algorithm will be used if you execute the *reroute xy* or *reroute yx* command as described above. The other version, called raw *xy* or *yx* routing, does not consider which nodes or links that are failed. With this algorithm one may force routing to be set as if all nodes and rings are ok. This does not make all rings and nodes ok, but it will set the routing tables on each node as if they where. If raw *xy* routing is set on a system where nodes or rings has failed, one will not get full connectivity. The raw *xy* routing is mostly useful for debugging purposes. Use the command *reroute <xy / yx> raw* from **scaconftool** to set raw *xy* routing.

### 7.4.4 Maxcy routing algorithm

The *maxcy-algorithm* is a fault tolerant routing algorithm, capable of connecting the maximum number of nodes if one or more nodes are failing. When all nodes are working, the *maxcy-algorithm* is equal to dimensional routing. *Maxcy-routing* is selected with the *reroute maxcy* command. Rings and nodes which are not *OK*, as far as the configuration server is concerned, will be considered failed when routing is computed, and these nodes and rings will be routed around.

When one node fail, this will make the two SCI-rings that it is attached to useless for SCI communication. This means that other nodes on these two failed rings will only have one operational SCI-ring left. When this arises, the *scali-routing* algorithm will still be able to connect all remaining nodes, as long as the node is part of at least one working SCI-ring. The routing might result in some disabled links. Operational nodes, where one or more rings are broken, will have their link disabled in the direction of the broken ring.

### 7.4.5 Routing partitions

A routing partition (partition from here) is a collection of nodes which has a route to all other nodes in the same partition via the SCI network. This result in that all nodes in the same partition can communicate via the SCI interconnect. Nodes not part of the same partition does not have a route between them, and can not communicate via the SCI-network.

Partitions are reported as a positive integer, called partition number for each node. If a negative number or 0 is reported this have special meanings:

- 0:  
The node is unreachable.(Power off or considered dead in some sense). No MPI programs or anything else may run on nodes with partition number 0.
- -1:  
Partition number for this node is not known (initial state). Before routing is set, all nodes will have this partition number.

A partition can contain from one to all nodes in the system. The routing algorithms decide which node is part of which partition. If a node is *OK* but due to routing strategy it is not possible to connect this node to any other node on the SCI-network, it will receive an exclusive partition number. Nodes with exclusive partition numbers can not communicate via SCI-network with any other nodes, but may operate separately. Exclusive partition numbers are those that are held by only one node. You may list the partitions with the *list partition all* command in **scaconftool**.

### 7.4.6 Testing the routing

Once routing is set, the routing may be tested with the *sciping* command. This command will send an sciping between all nodes listed. The *sciping* command is a combination of **scash** and the command `/opt/scali/bin/sciping`. The response is a list of possibly not responding nodes for each node in the list. Remember that in a system where one or more nodes or rings are down, there may be nodes that will not respond. This apply to faulty nodes or nodes where both SCI rings are unusable. All nodes with the same partition number should be able to reach each other with *sciping*. (E.g *sciping #1*)



## 7.5 Installation and package dependencies

Before you can use **scaconftool**, the ScaConfNd and ScaComd package must be installed on each node. ScaConfSd and ScaConfC must be installed on the frontend. In order to get daemon control and sciping functionality from scaconftool, ScaSH package must be installed on the frontend. These packets will be installed together with the rest of Scali software when complete Scali SSP is installed.

Name	Description
ScaComd	Communication daemon (node)
ScaConfNd	Configuration daemon (node)
ScaConfSd	Configuration server (frontend)
ScaConfC	Scaconftool, ASCII based configuration tool (frontend)
ScaSH	Parallel remote shell interface to nodes (frontend)

**Table 7-2:** ScaConf packages

## 7.6 scaconftool - command reference

This section describes all available **scaconftool** commands and their syntax. The `<list>` parameter is essential. `<list>` is to be replaced with a set of nodes. For commands where a list of nodes are required, it is at least five different ways to state these nodes. For further information about node selection, read section 7.3.2.1. `<list>` is to be replaced by:

- keyword **all**, meaning all nodes.
- node state, meaning all nodes in this state (e.g OK, NO\_DAEMON etc.)
- list of node names separated by blanks, meaning all nodes explicitly listed.
- left blank, meaning all nodes in internal list.
- partition number, e.g. #1, meaning all nodes with this partition number.

### 7.6.1 ConfTool> console

*console* is used to attach to the console on a specified node.

Usage:

```
console <node name>
```

Example :

```
console node1
```

scaconftool will try to connect to console on node1.

### 7.6.2 ConfTool> daemon

*daemon* is used to start or stop daemons on nodes.

Usage:

```
daemon <daemon name> <action> [<list>]
```

Example :

```
daemon scacomd restart UNREACHABLE
```

Restart the communication daemon on all nodes with state UNREACHABLE

Example :

```
daemon scaconfnd restart NO_DAEMON
```

Restart the configuration daemon on all nodes with state NO\_DAEMON

### 7.6.3 ConfTool> fix

The **scaconftool** command *fix* try to bring the nodes in the list into state OPERATIONAL. This is achieved by restarting daemons that does not run, reloading essential drivers and reconnecting the nodes to configuration server.

Usage:

```
fix [<list>]
```

Example :

```
fix UNREACHABLE
```

**scaconftool** will attempt fix all nodes with status UNREACHABLE.

### 7.6.4 ConfTool> getopt

The **scaconftool** command *getopt* query the server for the value of the server options specified and list their values. If the list of options is empty, all available options are listed.

Usage:

```
getopt [<option> .. <option>]
```

<option> :

Can be replaced by any legal options.

Example :

```
getopt
```

```
getopt AUTO_REPAIR
```

### 7.6.5 ConfTool> setopt

The *setopt* command set the value of the specified server options.

Usage:

```
setopt <option=value> [<option=value>... ]
```

<option=value>:

Can be replaced by any legal options. Use *getopt* with no parameters to get a list of all legal options. Options described in Table 7-3:

Example :

```
setopt AUTO_REPAIR=OFF AUTO_REROUTE=OFF
```

OPTION	VALUE	DESCRIPTION
AUTO_REPAIR	ON*/OFF	Turn auto repair of machine on or off. Server will automatically poll erroneous nodes to check if they are ok, and reconnect them if they are.
REPAIR_INTERVAL	[3,300]	Server will wait for this many seconds before it will try to reconnect nodes that is reported as <i>UNREACHABLE</i> . Recommended value: 10*
AUTO_REROUTE	ON*/OFF	Server will automatically reroute cluster if change of state in one or more nodes is detected.
AUTO_RECONNECT	ON*/OFF	Nodes will be reconnected to the cluster automatically when they come up if this option is ON.
AUTO_NODE_ID	ON*/OFF	Automatic checking and setting of SCI node id on or off.
ROUTING_TYPE	SCA_ROUTE_DEFAULT*	Default routing type for the topology. Use 'info routalg' to get a list of all routing type available for your topology.
REROUTE_DELAY	[0,..,3*..,10]	Seconds to wait after a reroute is issued before new status is sampled.
SAVE	<none>	Save the options to file. This will cause server to start with the saved settings if restarted.
RESTORE	<none>	Restore factory defaults for all options

**Table 7-3:** Available server options. Default values are marked with \*

#### 7.6.6 ConfTool> select

The *select* command add nodes to the list of active nodes

Usage:

```
select <list>
```

Example :

```
select scali2-11 scali2-12
select all
select NO_DAEMON
```

### 7.6.7 ConfTool> unselect

The *unselect* command remove nodes from the list of active nodes.

Usage:

```
unselect <list>
```

Example :

```
unselect scali2-13 scali2-12
unselect all
unselect UNREACHABLE
```

### 7.6.8 ConfTool> list

The *list* command gives information about nodes

Usage:

```
list [<format>] [<list>]
```

<format>::

```
[ [status] [nodeid] [c0] [c1] [partition] [position] ] | configuration
```

status: Status for the node.

nodeid: Node identification for SCI devices

c0: Status for controller 0

c1: Status for controller 1

partition: Routing partition number

position: pair of X/Y coordinates

configuration: List the configuration of the cluster in a matrix representation, reflecting the topology in the cluster.

Node name is always printed. Default is that only nodeid and node name is printed. The format is stored, and will be used if the next list-command does not specify any format.

Display formats:

Name: Node name

Nodeid : Hex number

Status: OK indicates that everything is ok

UNREACHABLE indicates the node is down or unreachable to the tool for some reason. The reason might be no power or scacomd does not run on node.

NO\_DAEMON indicates the ScaConf daemon is not running on the node.

NO\_SCI\_DEVICE indicates the SCI driver is not loaded.

UNKNOWN is an initial state, indicating that no information is received on the node yet.

Ctrl0:

Ctrl1: EN indicates that SCI link controller is enabled.

DIS indicates that SCI link controller is disabled.

UP indicates that the SCI link is up.

DOWN indicates that the SCI link is down.

Display format from the command:

```
ConfTool> list nodeid status c0 c1 all
Name      NodeId Status  Ctrl0 Ctrl1
scali2-11 0x1100 OK     EN/UP  EN/UP
scali2-12 0x1200 OK     EN/UP  EN/UP
scali2-13 0x1300 OK     EN/UP  EN/UP
scali2-14 0x1400 OK     EN/UP  EN/UP
```

Example:

```
list
```

Lists information on nodes in built-in list. Use select to add nodes to the internal list. The format from previous list command will be used:

Example:

```
list nodeid status c0 c1 scali2-11 scali2-13
```

Lists nodeid, status and controller states for scali2-11 and scali2-13

Example :

```
list status all
```

List status for all nodes

### 7.6.9 ConfTool> fail

The *fail* command turn off nodes in the sense that status become UNREACHABLE.

Usage:

```
fail [<list>]
```

Example:

```
fail
```

Fail nodes in the active list.

Example :

```
fail all
```

Fail all nodes

### 7.6.10 ConfTool> reconnect

The *reconnect* command does a reconnect to specified nodes.

Usage:

```
reconnect [<list>]
```

Example :

```
select UNREACHABLE
```

```
reconnect
```

Attempts to reconnect all nodes with state UNREACHABLE

**7.6.11 ConfTool> log**

The *log* command is used to enable or disable messages from the tool.

Usage:

```
log [enable/disable]
```

Example:

```
log disable
Make the tool work silent
```

**Example:**

```
log enable
Make the tool verbose.
```

**7.6.12 ConfTool> nodeid**

The *nodeid* command is used to set node-id for a node

Usage:

```
nodeid [<node name> <id>]
<id>: The id can be either decimal, octal (leading 0) or hex (leading 0x).
With no arguments nodeid will be set to default for all nodes.
```

Example:

```
nodeid
Set nodeid to default for all nodes.
```

Example :

```
nodeid scali2-24 0x2400
Set nodeid for node scali2-24 to 0x2400
```

**7.6.13 ConfTool> power**

The *power* command turn power on or off for the specified nodes.

Usage:

```
power <on | off | cycle> [<list>]
<action>:  on - turn power on
           off - turn power off
           cycle. - turn power off and the on again
```

Example :

```
power cycle UNREACHABLE
Turn power off and on again for all UNREACHABLE nodes
```

**7.6.14 ConfTool> reload**

The *reload* command does a reload of the SCI driver on the nodes

Usage:

```
reload [<list>]
```

Example:

```
reload scali2-22
Reload SCI-driver on node scali2-22
```

### 7.6.15 ConfTool> reroute

The *reroute* command does reroute the cluster with specified algorithm. Use 'info routalg' to get a list of available routing algorithms for your topology.

Usage:

```
reroute [routing algorithms]
```

Example :

```
reroute maxcy
Use the maxcy routing algorithm to route the 2d torus.
```

### 7.6.16 ConfTool> status

The ScaConfTool command *status* print information about nodes, controllers or links according to the cluster configuration. Looks like the *list configuration* command, but instead of node name, a three letter code is indicating the status of the node.

Usage:

```
status [nodes|controllers|links]
```

Example:

```
status nodes
Print status information for each node.
```

Example:

```
status controllers
Print information on controllers on each node.
```

Example node output which display three letter status codes.

```

1      2      3      4
UNK ---000 ---000 ---000
|      |      |      |
... ---000 --- 000 ---0..
000   : The node is OPERATIONAL
...    : The node is UNREACHABLE
0..   : The node has no node daemon (NO_DAEMON).
00.   : The SCI driver is not loaded (NO_SCI_DEV)
INV   : This is an invalid state, and should never occur.
UNK   : The node is in state UNKNOWN.
```

Example : Controller/link output with display codes.

```

?      ?      1      1
? ? --? ? --0 0 --0 0
?      ?      1      1
|      |      |      |
```

```

0      0      1      1
1 1 -- 1 1 --1 1 -- 1 1
0      0      1      1

```

**0** : disabled/down  
**1** : enabled/up  
**?** : unknown/unknown

#### 7.6.17 ConfTool> link

The *link* command can disable or enable links.

Usage:

```
link <[disable | enable]> <controller number> [<list>]
```

Example:

```
link disable 0 all
Disable controller 0 on all nodes.
```

#### 7.6.18 ConfTool> update

The *update* command ask the configuration server for updated system information.

Usage:

```
update [ <list> | system]
```

Example:

```
update scali2-13 scali2-23
Ask for updated information on nodes scali2-13 and scali2-23.
```

Example:

```
update system
Ask for new information on whole system.
```

#### 7.6.19 ConfTool> restart

The *restart* command does an on-line restart of the configuration server.

Usage:

```
restart
```

#### 7.6.20 ConfTool> sciping

The *sciping* command send ping over SCI to other nodes.

Usage:

```
sciping [<list>]
```

Example:

```
sciping node1 node2 node3
send sciping from the listed nodes to the listed nodes
```

#### 7.6.21 ConfTool> help

The *help* and *?* command print a help message.



Usage:

```
help/? [<command>]
```

Example :

```
help/? restart
```

Print information about the “restart” command

Example :

```
help/?
```

Print a list of all available commands

### 7.6.22 ConfTool> quit

The quit command exits the command tool.

Usage:

```
quit
```



## Chapter 8 Scali System Monitoring

---

The Scali system monitoring system: ScaMon provides flexible and powerful monitoring of a clustered computer system. Based on the industry standard SNMP a number of system and interconnect parameters may be monitored. The graphical monitoring plug-in for ScaDesktop provides a number of presentation options ranging from simple “xload” style 2D history graphs to accelerated 3D OpenGL graphics.

### 8.1 Overview

ScaMon consists of two main components: The Scali SNMP daemon **scasnmppd**, and the Scali monitoring daemon **scamond**. The SNMP daemon must run on every node to be monitored, and the Scali monitoring daemon must run on the workstation denoted front-end for the cluster. Scali recommends that a separate workstation is used as front-end for running configuration and monitoring software, but the SSP supports that one of the compute-nodes may double as front-end. This is of course a less fail safe configuration.

### 8.2 The monitoring daemon scamond

The Scali monitoring daemon: **scamond** provides filtered monitoring values to the clients. This means that two clients requires the same parameter only one SNMP request is sent. Hence both the network load and CPU load from the **scasnmppd** is reduced.

#### 8.2.1 Configuration file

The scamond configuration file defines a subset of monitoring variables that will be available for monitoring by other applications. It is located in

```
/opt/scali/etc/ScaMond.conf
```

A configuration file may look like this:

#### 8.2.2 Manual start/stop

Although **scamond** will be installed and started by the SSP install script it may be started, stopped or restarted with the command below should it be necessary:

```
# /opt/scali/init.d/scasnmppd [start|stop|restart|info]
```

## 8.3 The SNMP daemon scasmpd

The Scali SNMP daemon: **scasmpd** is based upon the UCD SNMP package [18]. The daemon has been extended by Scali with monitoring and configuration variables (OIDs) for Dolphin PCI-SCI adapters and various other host specific variables.

The Scali SNMP daemon is configured to not interfere with existing SNMP daemons by using a non-standard SNMP port. The default port used by **scasmpd** is: 32016

### 8.3.1 Manual start/stop

Although **scasmpd** will be installed and started by the SSP install script it may be started, stopped or restarted with the command below should it be necessary:

```
# /opt/scali/init.d/scasmpd [start|stop|restart|info]
```

This chapter is the place to start if you are experiencing problems either using or installing a Scali system. Some of the most common problems and their solutions are described here. Remember that the FAQ (Frequently Asked Questions) lists at Scali's web site may contain information which is more up to date than the one contained in this document.

If your problems persists please refer to "Technical Support" on page 97 which explains how to get technical support.

## 9.1 Hardware installation problems

Before you investigate your problems further, make sure the hardware installation is in accordance with the guidelines in Chapter 3.

Problem	Description	Solution
No SCI adapter card jumpers installed.	The SCI adapter card jumpers described in Figure 3-3: are not installed.	Install the jumpers.
Red LEDs on the SCI card	This is quite normal. LEDs will remain red until the cabling is done and the SCI software is loaded.	Complete cabling and software installation.

**Table 9-1:** Hardware installation troubleshooting

## 9.2 Software installation problems

Before you investigate your problems, check that your software installation is according to the guidelines in Chapter 4.

Problem	Solution
Where can I find the ScaOsinstall program?	This program exists either on the Scali SSP CD-ROM or can be downloaded from: <a href="http://www.scali.com">http://www.scali.com</a> . One may also follow the manual OS installation described in section 4.2.
The software installation script reports errors.	Whenever the installation program run into unexpected situations, it will inform you about what is wrong, and make suggestions on how to solve this problems. Please follow the instruction given by the installation program very carefully.

**Table 9-2:** Software installation troubleshooting

## 9.3 License problems

Troubleshooting licensing problems is described in the appendix “Scali Software Licensing” on page 107

## 9.4 MPI problems

Common problems encountered running MPI programs with Scali’s MPI implementation ScaMPI is handled in the chapter “Getting Help” of the “ScaMPI User’s Guide” accompanying the product, or take a look at the on-line ScaMPI FAQ at <http://www.scali.com/support>.

## 9.5 SCI Interconnect problems

Scali uses its own highly optimized driver **ScaSCI** when utilising the ultra high speed SCI interconnect in a Scali system. Hence error messages regarding SCI origins from ScaSCI.

### 9.5.1 SCI error messages

The SCI status error messages are explained by the program `/opt/scali/bin/scierrmsg`. Taking as input a SCI status error message number or a symbolic name, **scierrmsg** (see D-1.3) gives a description of the cause of the error. Some common SCI error messages are explained here:

SCI error type	Description
ICMS_NO_RESPONSE	Reporting of this message type when attempting communication with an existing node is an indication that something is wrong because the remote node did not respond. Either the physical connection is broken, there is no route back from the destination or the node is down or unable to respond.
ICMS_CONNECTION_REFUSED	This message type is returned when the connection request was refused by the memory owner either because the resource (remote memory) does not exist or it is not currently offered to new connections.
ICMS_OUTBOUND_MAP_FULL	This error type indicates that not enough SCI map resources are available to satisfy the application need. (That is the local SCI adapter address translation table (ATT) is full and there is no room for more remote memory mappings). This is unrelated to the state of the interconnect (except of course that processes may hang and block resources because of some interconnect problem). See "wrong jumper settings" in Table 9-4:.
ICMS_FAILURE	This error type indicates that an operation failed due to an implementation detail which does not map to the abstract level. This may indicate a server memory alloc situation or to many open SCI driver connections.
ICMS_OUT_OF_MEMORY	This error type indicates that it was not possible to allocate the requested amount of pinned memory. Add more memory to the system or change configuration options to allow a more aggressive SCI memory allocation strategy.

**Table 9-3:** Some SCI error types

## 9.5.2 SCI Troubleshooting

Problem	Description	Solution
<b>Interrupt conflict</b>	During ScaSCI install the SCI hardware have an interrupt conflict with other hardware cards.	Please consult the operating system install documentation about how to solve the problem.
<b>Wrong jumper settings</b>	After installation of the ScaSCI software package, run: <code>/opt/scali/sbin/scinfo -m</code> to check that jumper settings are correct. Used/available ATT entries should typically be: RMAP_IO of 1/256 and RMAP_GATHERING of 1/256 when ATT size is 512k (default), i.e., the SCI driver is able to map a maximum of 256*512k=128M remote memory.	Check the ScaSCI release notes if your values are different and correct the SCI board jumper settings or the driver configuration values.
<b>No adapters found</b>	Use the <code>/opt/scali/sbin/scicards</code> script to find number of installed SCI adapters on the node. If the answer does not match your hardware installation of SCI adapters or you get the error message: <i>SciInitialize failed because no adapters found</i> , check if the SCI hardware are recognised by the OS.	Are your mother board supported by Scali tests? Check the ScaSCI release notes.  Check if the SCI hardware are recognised by the OS: <b>Solaris:</b> <code>% prtconf -pv   grep 'PCI: 11c8'</code> model:'PCI:11c8,658-class:Bridge dev.' model:'PCI: 11c8,0 - class: Bridge device' model:'PCI: 11c8,658 - class:Bridge dev.' <b>Linux:</b> <code>cat /proc/pci   grep 'Vendor id=11c8'</code> Vendor id=11c8. Device id=658.
<b>LEDs on SCI cards are red.</b>	One or more of the LEDs on the SCI card is red.	Check that cabling is according to the guidelines in section 3.3.2. Check that all cables run from IN connector to OUT connector, and that no SCI main boards are connected via cables to SCI daughter boards and visa versa
<b>Red LEDs on SCI cards.</b>	One or more of the LEDs on the SCI card is red, and the cabling is checked and found ok.	Check that all the cables is properly inserted into the connector, and tightening all screws.

Table 9-4: ScaSCI trouble shooting



Problem	Description	Solution
<b>Red LEDs on SCI card</b>	Checking cabling and tightening screws didn't help.	Check that the SCI driver is loaded on all nodes. You may check this with the <i>list status all</i> command in <b>scaconftool</b> , see section 7.6
<b>OS does not recognizes the SCI card</b>	Some PCI slots does not fully support the specification. Interference with AGP or other hardware devices may happen.	Try another PCI slot for the SCI card. If changing PCI slot for the SCI cards does not help, please refer to Dolphin. <a href="http://www.dolphinics.com">http://www.dolphinics.com</a> .
<b>sciping reports "not responding" nodes</b>	The SCI utility <i>/opt/scali/bin/sciping</i> or the sciping command in the ScaConf utility <b>scaconftool</b> reports not responding nodes.	Reroute the cluster with <b>scaconftool</b> or ScaDesktop. Make sure sciping is run only between nodes with same routing partition. See 7.4.
<b>ScaSCI log message RMAP_OVERFLOW</b>	The ScaSCI log contains RMAP_OVERFLOW message of lost pages: Unix: WARNING: kernel map: rmap overflow, lost [39250, 39254] Unix: WARNING: kernel map: rmap overflow, lost [39238, 39242] ... error messages repeated multiple times in the log.	The problem is the MEM_BLOCK_SIZE setting in the ScaSCI.response file that are reflected to the mem_block_size variable setting in the ssci.conf file on all nodes at ScaSCI package installation time. This variable regulates the size of each page aligned block of memory allocated from the OS (e.g. "granularity" of allocation).

**Table 9-4:** ScaSCI trouble shooting

## 9.6 Technical Support

Scali is proud to provide the highest level of technical support. Normal first-response time for a support request is less than 8 working hours. You will then be appointed a personal contact and a serial number for your particular problem.

### 9.6.1 Support Contracts

The duration and level of your support contract or warranty should be obvious from the invoice that came with the product. If in doubt just contact Scali. Extended support contracts or pay-per-request is also available. When requesting support please state your *name* and *company*.

### 9.6.2 How to report the problem

It will save both your time and ours if you do the following before requesting support

- Read the documentation again: Quite often the problems can be resolved from the relevant chapters in the manuals or FAQs.
- Collect version information: This includes version information for *all* involved

software; Scali SW, compilers operating system. Versions of HW: SCI-cards, computer architecture and chipset versions (BX, LX, NX ...).

- Isolate your problem to a small test case - if applicable. The shorter the better.
- Record the sequence of events causing the problem.

You can reach Scali's technical support using either:

**E-mail: support@scali.com (Preferred)**

**Phone: (+47) 2262 8950**

**Fax: (+47) 2262 8951**

### **9.6.3 Other Feedback**

Scali welcomes any suggestions, improvements, feedback or bug reports concerning this Scali System Guide or the hardware and software described herein. Such comments should also be sent to **support@scali.com**.

### **9.6.4 Keeping informed**

Apart from regular visits to the Scali web site Scali also recommends subscription to the two mailing lists maintained by Scali. These are scali-announce which is where Scali posts the latest news about Scali products, releases, patches and so on, and scali-user which is a discussion group for Scali users. Instructions on how to join the appropriate mailing-list can be found on **<http://www.scali.com/support>**.

---

## A-1 General package information

### A-1.1 Scali software platform CD-ROM

The Scali Software Platform (SSP) CD-ROM contains all Scali software packages needed to get a Scali system installed and operative. Below is a description of the top level file system layout on the SSP CD-ROM.

- `README` : Start by reading this!
- `doc` : documentation.
- `install` : The Scali software installation program.
- `uninstall` : The Scali software removal program.
- `pkg` : All the software distribution files.
- `bin` : binaries used during installation.

Before starting the installation program you should first install the SCI boards (see `doc/HW` and Chapter 3) and configure the operating system (see `doc/OS`) to suit the SSP.

#### A-1.1.1 Install

When you are ready to start the installation type the following (as root):

```
# ./install
```

The installation program will guide you through the installation process which installs the SSP on the whole cluster. The installation program will detect if this is a clean installation or an upgrade. The installation process is done in three different stages: configuration, package installation and test, one may restart the installation at any of these stages without the need to run the previous stage again. If the configuration went OK but there were problems with the package installation you may run the package installation again using the `-i` option.

```
# ./install -i
```

Likewise to run the test stage again do a:

```
# ./install -t
```

### A-1.1.2 Uninstall

If you wish to remove the SSP at a later stage you can run the uninstall program (as root):

```
# ./uninstall
```

This will remove all traces of the SSP on all nodes.

### A-1.2 Scali software directory structure

All Scali software is installed under the directory `/opt/scali`. The top level directory structure is as follows:

- `/bin` : all normal executables to be run by users.
- `/sbin` : all daemons and executables to be run by administrators.
- `/libexec` : executables which are hidden from normal invocation (e.g. used by applications under `/bin`).
- `/lib` : libraries used by our applications and by end users.
- `/include` : include files for libraries under `lib`.
- `/doc` : all documentation.
- `/etc` : configuration files for Scali software.
- `/examples` : example MPI applications and source.
- `/contrib` : 3rd party software adapted to Scali software.
- `/init.d` : boot/startup scripts. Soft links are made from system startup catalogues (`rc.d` and `init.d` under `/etc`) by install script.
- `/kernel` : kernel related files.
- `/man` : manual pages
- `/plugins` : plugins for the Scali desktop GUI
- `/share` : share directory used by SNMP clients/agents

## A-2 Scali package file naming convention

Every software unit is distributed as a single package file:

```
module.os.arch-x.y.z.package
```

where:

- `module` is the software name e.g. ScaMPI, ScaSCI, ScaPkg.
- `x.y.z` is the release number, e.g. 1.0.2,
- `os` is the operating system, e.g. SunOS5 or Linux2
- `arch` is the architecture, e.g. sparc-u, i86pc or alpha
- `package` is e.g. pkg, rpm or exe depending on the operating system.

## A-3 Scali software platform contents

Product name	Description
ScaDesktop	This is graphical user interface which unifies access to several Scali tools and applications. It is modular by design and makes use of plug-ins to extend it's functionality.
ScaMPI	This is Scali's implementation of the MPI standard 1.2 and tools for application loading. A number of benchmarks and test applications are bundled together with an adoption of the mpich "upshot" MPI call event tracer. A ScaDesktop plug-in and standalone application loader is supplied.
ScaSCI	The Scali SCI drivers used by ScaMPI.
ScaMon	ScaMon is the Scali monitoring system, consisting of the Scali SNMP daemon and the Scali Monitoring server; ScaMond. A rich set of ScaDesktop plugins are provided enabling different presentations of monitoring data.
ScaConf	ScaConf provides cluster administration with e.g. power switching, console access and full connectivity routing. A ScaDesktop plug-in is supplied.
ScaSH	ScaSH contains tools for executing commands in parallel on a selected set of nodes. A ScaDesktop plug-in is supplied.
ScaPkg	ScaPkg provides tools for installing, updating or removing software packages on selected categories of nodes. Package files are kept in a repository on the installation server, distributed and installed with automatic configuration. A ScaDesktop plug-in is supplied.
SSPinstall	The SSP install program provides installation of all Scali software onto nodes in the cluster. The user is assisted through the installation step by step until the cluster is fully operational.
ScaOSinstall	The Scali Operating System installation tool simplifies OS installation on a cluster by creating and configuring an installserver on one node and subsequently use this node to install the OS in parallel on all the remaining nodes in the cluster.
ScaBoot	ScaBoot provides an x86 boot loader enabling OS selection from console and/or keyboard. Includes tools for selecting which OS to boot next.

**Table 9-5:** Overview of Scali software products distributed with the SSP.

Section:

# Appendix B      ScaPkg - Scali Software installation program

---

The Scali software installation program: **scapkg** enables the maintenance of software packages on several nodes across different platforms in parallel. It will automatically compare versions of installed packages with versions on packages in a repository and install, remove or upgrade the software packages on a selected set of nodes accordingly.

## B-1 Using scapkg

Before installing packages you may use the dryrun option **-D** to check what the command actually does. Use **-h** to list all **scapkg** install options. As an example we will show how to install a ScaMPI package from the file `scaMPI.Linux2.i86pc-1.8.5.rpm` located in the repository `/home/software`. Here we recognize the format `module.os.arch-x.y.z.package` from section A-2. Note that for Scali packages it is only the `module` part of the package file name that is used to specify the package. It is important that no two package files with same `module` prefix is present in the repository when **scapkg** is used.

```
# scapkg -p scaMPI -r /home/software nodeX nodeY nodeZ
```

If no nodes are specified on the command line, **scapkg** will use nodenames from the `nodelist` in the configuration file `scaPkg.conf` as default. For further information about the configuration file, see section B-2.

Usage:

```
scapkg [-dvVDSfeh?] [-p "<package> ..."] [-r "<path to repository> ..."]  
[<host> ...]
```

Options:

- p** Specify packages to install. The default is to use the packages found in the default repository. See configuration file `/opt/scali/etc/ScaPkg.conf` for path to repository.
- c** Specify which categories nodes belong to. This switch can only be used in combination with explicit nodes (given as arguments) when **-p** is not used and will only influence nodes not found in the configuration file.
- r** Specify the path to the repository. See configuration file `/opt/scali/etc/ScaPkg.conf` for default path to repository.

- s Enable "sync" mode. scapkg will then try to install and remove packages in order to reflect the setup in the config file (`/opt/scali/etc/ScaPkg.conf`). The need to remove packages can occur if one has manually installed packages.
- f Enable "force" mode. This will skip the version test and install it anyway.
- e Enable "erase" mode. This will remove packages. This switch may not be used in combination with "sync" mode.
- D Enable dryrun mode. No action taken, but everything is checked.
- d Enable debugging mode. This will generate logfiles under /tmp.
- S Enable sequential mode. The default is to run as much as possible in parallel.
- v Enable verbose mode.
- V Show version.
- h/? Usage information.

It is possible to use **scapkg** for handling general packages (i.e not Scali specific packages). A general package may not have the string "Sca" as prefix in its filename. Another restriction for general packages is that the first part of the filename should be identical with the name of the package

## B-2 Configuration

The ScaPkg configuration file is located in:

```
/opt/scali/etc/ScaPkg.conf
```

The configuration file will be created as part of the SSP installation process and contains enough information to tell **scapkg** which packages would be installed on which nodes. Changes to the configuration file can be made either manually with a text editor, by using the "Software" menus in Scali Desktop (see "Software installation - Software menu" on page 57) or as a result of running the SSP installation program. The ScaPkg configuration file is divided into four logical sections which are described in sections B-2.1 through B-2.4.

### B-2.1 ScaPkg.conf - path to package files (repository)

The path-to-repository section defines a path to the directory holding the software packages (repository). Note, an architecture identification string will be appended at the end of the path corresponding to the output from the **systype -c** script from the ScaEnv package when searching for packages made by Scali. For example, if `/usr/local/src/pkg` is the repository, a Linux i86pc system would expect to find package files in `/usr/local/src/pkg/Linux2.i86pc` and an UltraSPARC system would expect to find package files in `/usr/local/src/pkg/SunOS5.sparc-u`. When searching for



general packages, (not Scali packages), the search order would be `/usr/local/src/pkg/Linux2.i86pc` and then `/usr/local/src/pkg` if the package was not found at the first search location. It is possible to specify more than one `path-to-repository` locations. The search order for each package will be as listed in the `path-to-repository` section of the configuration file. The `path-to-repository` may be overridden with the `-r` option.

### B-2.2 ScaPkg.conf - package list

The package list section defines which packages are associated with which categories. Each line in the package list is on the format:

```
package <name> <categories>
```

where `<name>` is the package module part, and `<categories>` is one or more categories separated by space. The categories are freely defined in the node list section, see B-2.4. Each package can belong to any number of categories. Packages with no category are not to be installed (only kept in repository). Packages will only be installed on nodes with corresponding categories. No user changes should normally be necessary. Here is an example package list:

```
# Format: package <name> <categories>
package ScaBoot      node
package ScaConfC     frontend
package ScaEnv       development frontend node
```

### B-2.3 ScaPkg.conf - dependency list

The dependency list section defines dependencies between packages. Each package may have several dependencies. A line in the dependency list is on format:

```
dependency <pkgname> <deppkg>
```

where `<pkgname>` is the package module and `<deppkg>` is a list of all packages that this package depend upon. Note that no user changes should be necessary in this section. Here is an example package dependency list entry, defining dependencies for the package `ScaConfC`:

```
# Format: dependency <pkgname> <deppkg>
dependency ScaConfC      ScaConfSd ScaConfNd ScaSH
```

#### B-2.4 ScaPkg.conf - nodelist

The nodelist section defines which nodes to install packages on and which categories they belong to. The following categories are defined:

- **node** Node running parallel applications.
- **frontend** Node holding license servers, installation tools etc.
- **development** Node holding development tools like compilers etc.
- **installserver** Os installation frontend.
- **user** User node, limited number of software packages shall be installed.

Each host may belong to any number of categories. The format for a node list entry is

```
node <name> <categories>
```

where <name> is the name of the node and <categories> is a list of all categories this node is part of.

Here is an example nodelist defining four nodes `nodeW`, `nodeX`, `nodeY`, `nodeZ` and a frontend machine, `boss1`:

```
# Format: node <name> <categories> ...
node nodeW node
node nodeX node
node nodeY node
node nodeZ node
node boss1 frontend
```

#### B-2.5 Package response files

If a file with the name <package>.response is found in the package repository (without architecture extension), this file will be read when installing <package>. Examples are `ScaLM.response` and `ScaSCI.response`. No user changes should be necessary for these files.

## Appendix C Scali Software Licensing

---

### C-1 Introduction

Normally all necessary components of the Scali software licensing system will be installed and configured by the SSP **install** program, but if you for some reason should need to alter or update it manually this appendix explains the most common procedures.

**Note:** Prior to release 2.1 of the SSP Scali were using the FLEXlm license manager system from Globetrotter Software Inc. If you are experiencing problems regarding FLEXlm please refer to this section of a version 2.0 or older version of the "Scali System Guide". Older versions of the System Guide are available from the download section on Scali's web-site.

### C-2 Licensing Software

The Scali licensing software is distributed in the package; ScaLM which can be found on your Scali SSP distribution CD-ROM or downloaded freely from the download section on Scali's web-site:

`http://www.scali.com/download`

The ScaLM package should be installed on *every* node in the system including the front-end. The most efficient way of doing this is to use ScaPkg. For help on how to install a software package using the **scapkg** utility, please refer to "Appendix B ScaPkg - Scali Software installation program". Alternatively the text file `INSTALL`, distributed with every Scali package, contains instructions on how to install a package on every platform supported by the SSP.

### C-3 The license file

#### C-3.1 Default location

The default license file for Scali software is:

`/opt/scali/etc/license.dat`

This file holds the licenses for all Scali software and a copy must be distributed to the same location on *all* nodes of the system.

### C-3.2 SCALM\_LICENSE\_FILE environment variable

The SCALM\_LICENSE\_FILE environment variable may be used to alter the name and location of the license file. Simply set it to the full path of the license file. If SCALM\_LICENSE\_FILE contains a valid file name, this file will be tried before the default location.

### C-3.3 Installation (distribution)

After the license file has been altered it must be installed (distributed) to the same location on all nodes in the system. The most efficient way of doing this is by using the **scarcp** utility from ScaSH. Assuming you are on the front-end with an updated license file in the default location, the command:

```
scarcp /opt/scali/etc/license.dat
```

will copy the license file to the same location on all nodes - simple!

### C-3.4 License file example

After a successful installation by the SSP **install** program or after manual editing, the license file should look something like this:

```
#
# Scali AS - license file
#
#
# ScaMPI package
#
FEATURE mpi scald 1.900 1-aug-2000 uncounted 03BEF3779ADF9C9847D7C2435C24\
        HOSTID=DEMO
#
# ScaConf package
#
FEATURE confsd scald 1.000 1-aug-2000 uncounted 1B335F1429BB9C876A5645FE92\
        HOSTID=DEMO
```

Lines starting with # are comments. You may have several FEATURE lines, one for each licensed software package.

### C-3.5 Adding or updating features (licenses)

In the license file, each individually licensed software package will have its own FEATURE line. When you receive a software license from Scali, be it a demo or a permanent license, this will be in the form of a new FEATURE line. The FEATURE lines should be inserted in the license file, possibly replacing an older one. When receiving a multi-line FEATURE it is important to keep the format exactly as received from Scali, including whitespaces, as it otherwise might not work.

## C-4 Requesting licenses

Requests for permanent or demo licenses should be sent to: **license@scali.com**.

### C-4.1 Demo licenses

Scali will provide demo licenses for evaluation purposes. Since demo licenses are valid for all hosts until the expiration date, no system information is required. You may send a simple E-mail request or use the "License Request" form in the support section of the Scali web-site: <http://www.scali.com/support>

### C-4.2 Permanent licenses

Permanent licenses for Scali software are node-locked. Therefore, in order to receive a permanent license, you will need to supply Scali with the hostid for all nodes in the system, including the hostname and hostid of the frontend. Fortunately, the SSP **install** program now does this automatically as part of the SSP installation process. A file containing all relevant information for a license request will be generated and placed in: `/tmp/SSPinstall.licrequest`.

If you wish to generate a new license request at a later time, simply rerun the **install** program with the **-l** option and follow the instructions.

```
install -l
```

Alternatively, you can collect the same information manually by running a small ScaLM utility program: **lmhostid** on every node. The **lmhostid** is included in the ScaLM package.

```
% /opt/scali/bin/lmhostid
The host ID of this machine is "80c661b6"
%
```

## C-5 Troubleshooting

If your program fails to start due to a license problem this will be clearly indicated by an error message logged to the system logfile. The log files are: `/var/log/messages` for Linux and `/var/adm/messages` for Solaris.

### C-5.1 Error: Invalid FEATURE line

This means that the license file found by the application does not contain any valid FEATURE lines entries, or that the encryption key of the FEATURE is invalid.

### C-5.2 Error: Invalid version x.x > y.y

This means that the version of the FEATURE you're trying to check out is newer than the one you have in your license file. The most common reason for this is that you are trying to use an outdated copy of the license file. This may happen if you upgraded your software and forgot to update the license file.

### C-5.3 Error: FEATURE has expired

This error occurs only with time limited licenses and tells you that the license for this FEATURE has expired. The only time limited license used by Scali software are DEMO licenses

### C-5.4 Error: Host ID is not found!

This error occurs with host bound licenses only and means that the requested FEATURE is not valid for this host. Most Scali software licenses are host-bound.

## D-1 SCI hardware status programs

This chapter will describe some SCI utility programs. These programs are located in the `/bin` and `/sbin` directories of the Scali installation. They are useful for testing and monitoring the SCI hardware and of limited interest to the ordinary user, *and should not be used under normal circumstances.*

### D-1.1 sciping

The **sciping** program checks the reachability of SCI nodes through an adapter. Default adapter number is 0. The most important options are listed below. The sciping command will send an SCI packet via the SCI interconnect from the node it is run on to the nodes with the nodeid specified on the command line. This utility is useful for testing connectivity between nodes.

Usage:

```
/opt/scali/bin/sciping [-L][[-x|-t|-s]<tag>] <nodeid0> <nodeid1> ..
```

Options:

<b>-a &lt;adapter&gt;</b>	Use adapter number 'adapter' instead of adapter 0
<b>-m &lt;count&gt;</b>	Send count requests for each node on command line
<b>-c &lt;sec&gt;</b>	Continuous mode. Repeat requested pings every sec seconds
<b>-p</b>	Print performance statistics (t/ping)
<b>-u</b>	Allow loop back pings (potentially unsafe)
<b>-h</b>	Print this text
<b>-v</b>	Only print nodes that fails to respond/links that are down
<b>-L</b>	Write a log message to the system log at both nodes (requires all involved nodes to run a similar sciping)

**D-1.2 scimonitor**

The **scimonitor** program watches link status changes for an SCI adapter. The default is adapter number 0.

Usage:

```
/opt/scali/bin/scimonitor [-a<adapter>]
```

[Options]:

**-a <adapter>** Adapter number to monitor.

**D-1.3 sciemsg**

The **sciemsg** script maps a return value from a call to the SCI API or a symbolic error type to the corresponding error message. See Table Table 9-3 on page 95 for some examples of symbolic SCI error types.

Usage:

```
/opt/scali/bin/sciemsg <SCI status error message number>
```

**D-1.4 scidbx**

The **scidbx** program is a text based command line tool for inspection and manipulation of the SCI hardware through the SCI driver. When started, scidbx will prompt you with **#scidbx>** ready to accept commands. Type **help** for further information about available commands in **scidbx**. The **scidbx** program is only available to root.

Usage:

```
/opt/scali/sbin/scidbx [-a<adapter>][-f]
```

[Options]:

**-c** or **-e** <string> Execute <string> as scidbx command in batch mode.  
**-f** Force start even when adapter sanity checks fails.  
**-h** print this text.

**D-1.5 scideb**

The **scideb** program will set the debug level of the SCI driver. This setting can change the amount and character of SCI driver information printed in the system log files.

Usage:

```
/opt/scali/sbin/scideb [-V] <new debug level>
```

[Options]:

**-V** Enable verbose mode

Possible levels (in any combination):

SSCI\_INFO      0x1      /\*Various infrequent info \*/



SSCI_ICM	0x2	/* Use of KalLog in OS independent code */
SSCI_CFG	0x4	/* Configuration info */
SSCI_UE	0x8	/* Details about user errors */
SSCI_CR	0x20	/* Connect and release details */
SSCI_ENT	0x40	/* Entry and exit of entry points */
SSCI_CB	0x80	/* Entry and exit of callbacks */
SSCI_FABRIC	0x100	/* Interconnect fabric related conditions */
SSCI_PKT	0x200	/* SCI software packet interface */
SSCI_AEH	0x400	/* Asynchronous error handling */
SSCI_MSEG	0x800	/* Memory segment details (not too verbose) */
SSCI_MMAP	0x1000	/* Entry in mmap */
SSCI_CH	0x2000	/* Channel interface */
SSCI_KAL	0x4000	/* Enter and exit of Kal funs except mutexes */
SSCI_SAL	0x8000	/* Suballocator */
SSCI_ICME	0x10000	/* Use of KdPrint in OS independent code */
SSCI_INTR	0x20000	/* User interrupt processing */
SSCI_SOFT	0x40000	/* DDI_soft */
SSCI_MEM	0x80000	/* Memory allocation and export process */
SSCI_MCTX	0x100000	/* Mmap context adm */
SSCI_PG	0x200000	/* Page pool adm.info */
SSCI_CH_V	0x800000	/* Verbose printout for channel interface */
SSCI_RCHUNK	0x2000000	/* Remote connect/disconnect details */
SSCI_LCHUNK	0x4000000	/* Local chunk maintenance details */

#### D-1.6 scinode

The **scinode** program prints/sets nodeid of the specified adapter. Default is to print the nodeid of adapter number 0 on standard output. Only root may set nodeid. The **scinode** options are listed below.

Usage:

```
/opt/scali/sbin/scinode [-h][-v][-s <new nodeid>][-a <adapter no>]
```

Options:

- h Print this help string
- v Print version information
- a<n> Use adapter number <n>
- s<id> Sets new nodeid for the adapter

#### D-1.7 scinfo

The **scinfo** utility gives global information and statistics for all SCI instances, and are particularly useful as an aid in error diagnostics.

Usage:

```
/opt/scali/sbin/scinfo [-v][-a <adapter number>]
```

[Options]:

```
-m    show memory and att usage statistics
-n    show processes with active SCI connections
-l    show SCI link statistics
-p    show pkt statistics (driver-to-driver comm.)
-c    show driver configuration and version
-i    show general interrupt statistics
-v    all of the above (equal to -mlpci)
-h    print help text
```

### D-1.8 scireconf

The **scireconf** program may be used to reconfigure the Scali SCI driver. This utility is only to be used upon instructions from Scali.

Usage:

```
/opt/scali/sbin/scireconf [-d] parameter [value]
```

[Options]:

```
-d parameter    Delete (comment out) the parameter setting
parameter      Display current setting if any
parameter value Change (and enable) parameter with this value
```

### D-1.9 scireload

The **scireload** command will reload the Scali SCI driver on the machine the command is run on. This command is only available to root, and it should not be necessary to use this command under normal circumstances.

Usage:

```
/opt/scali/sbin/scireload
```

#### D-1.10 scidle

The **scidle** utility returns status code 0 if SCI links are ok and idle.

Usage:

```
/opt/scali/sbin/scidle [-a <adapter no>]
```

[Options]:

**-a n** Use adapter number *n* (default 0)

#### D-1.11 scicards

The **scicards** script prints the number of SCI adapter cards installed and recognised by a node.

Usage:

```
/opt/scali/sbin/scicards
```

Section:

## E-1 Introduction

This appendix contains the “*Scali System Cabinet Assembly Guide*”. You will be shown in detail how to assemble the different hardware components of a Scali system cabinet.

A Scali system cabinet is a very practical way of arranging a number of computing nodes (PC's or workstations), interconnect, cabling, and other infrastructure hardware units like HUBs, remote power switches or console switches to form a single system.

The assembly guide is divided into three sections:

Section E-2 describes the cabinet assembly.

Section E-3 describes the insertion of the processing nodes into the cabinet.

Section E-4 describes the interconnect setup.

## E-2 Cabinet assembly

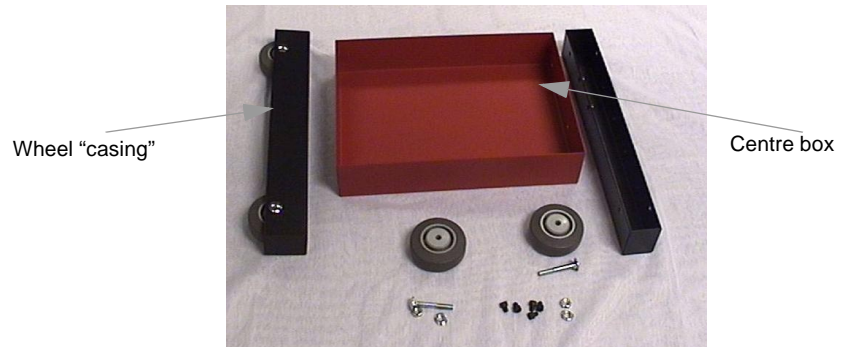
A Scali system cabinet consists of 1 to 9 node-enclosures. Each node enclosures have room for a standard midi-tower desktide PC-cabinet. Alternatively using the built-in 19” fittings each enclosure will accept up to 5U rackmount units. Rackmount units are typically used for HUBs, switches, remote power switches and so on.

In a full size (9 enclosure) cabinet, infrastructure usually occupies one enclosure leaving 8 for processing nodes. Multiple cabinets are clustered to form larger machines. The 24 node system in Figure E-1 uses 3 full size cabinets and two extra enclosures for a total of 24 enclosures for processing nodes and five for infrastructure.



**Figure E-1:** The Scali system cabinet

1. Base assembly, all major parts are shown.




**Figure E-2:** All major parts of the base assembly

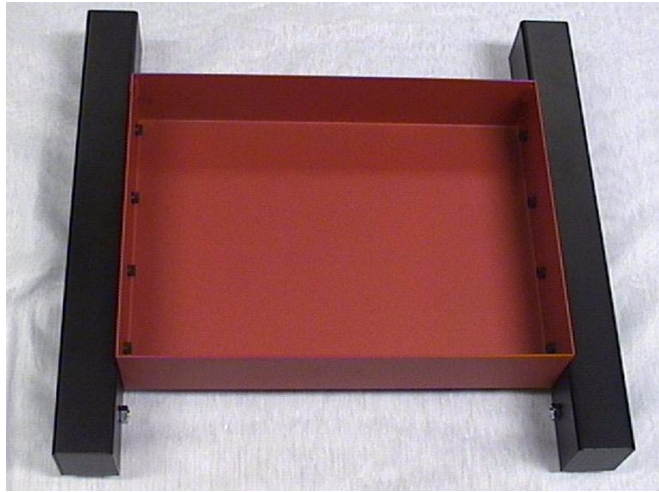
2. Assembly of wheels to the casing, note 2 + 1 washers on each wheel.



**Figure E-3:** Assembly of the casing wheels


3. Complete base assembly. Fasten the “wheel casings” to the center box using 4 unbraco bolts on each side.

8 x   
6mm



Completing the base assembly

4. Fasten the backbone to the base.

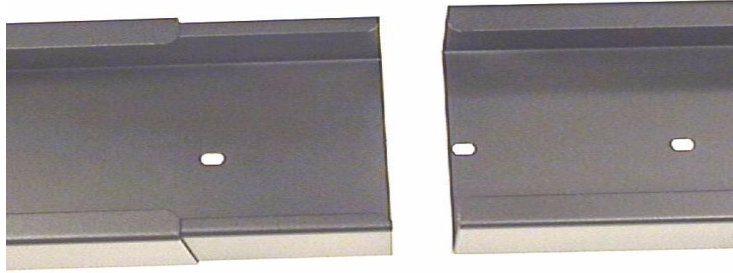
4 x   
6mm



**Figure E-4:** Assembling backbone to the base



5. For a cabinet with more than 5 enclosures, the backbone extension must be added to the backbone.



**Figure E-5:** Assembly of backbone extension

6. Fasten the node enclosures to the backbone:

2 x  
  
6mm  
pr. enclosure

Remove film from  
friction-tape under  
here!



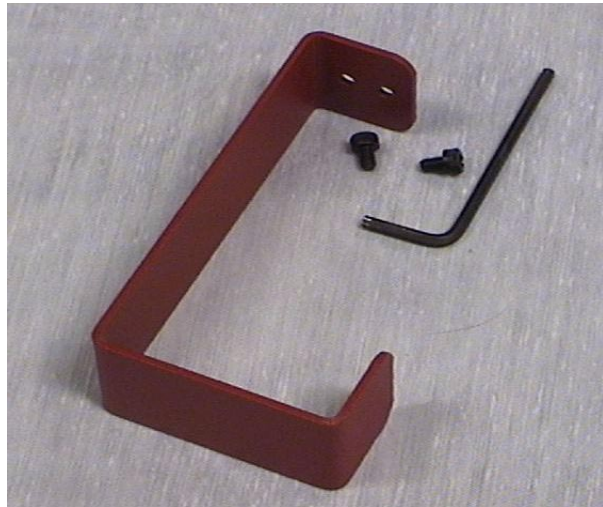
**Figure E-6:** Fasten node enclosures to the backbone

**IMPORTANT**

To increase the structural stability of the cabinet, small pads of high-friction tape have been fitted underneath the red distance bars under each enclosure. The pads are covered by a thin protective film which **must be removed** before the enclosures are stacked. Otherwise a less stable cabinet will result.

7. Cable hook (please note, these are slightly modified on new systems).

2 x   
3 mm



**Figure E-7:** Cable hook parts

8. Cable hook fastened to the backbone. 5 (or 9) cable hooks are needed.



**Figure E-8:** Fasten cable hooks to the backbone

Please note, the cable hooks are fastened to the node enclosures for new cabinets.

## E-3 Insertion of the processing nodes

9. The nodes fit into the node enclosures from the front.



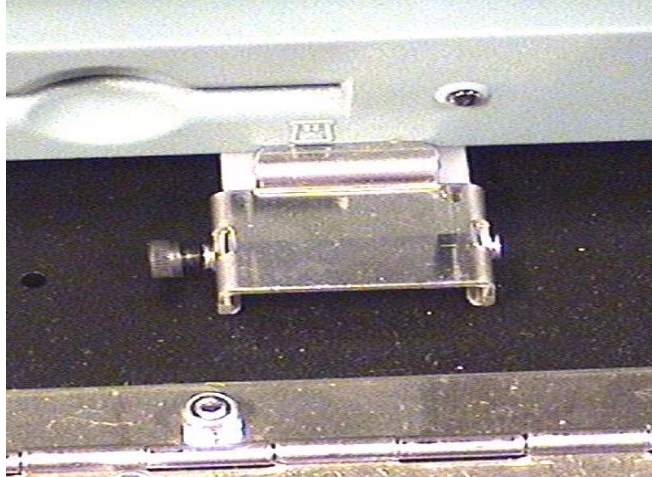
**Figure E-9:** Insertion of node in enclosure

10. If necessary adjust the guides at the back of the node enclosure to fit node width.



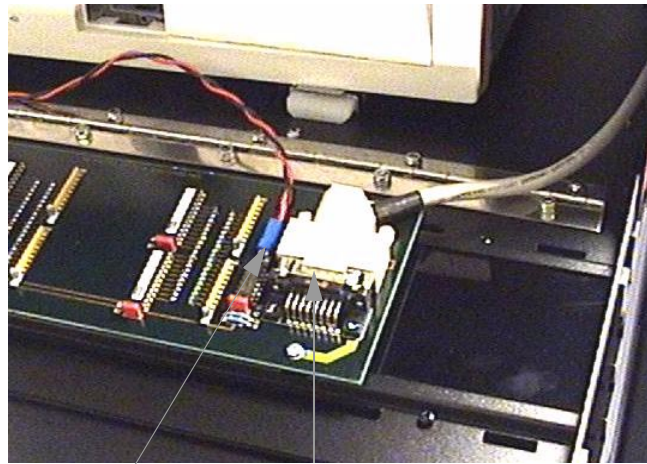
**Figure E-10:** Node guide adjustment

11. Fasten the node in front using the rail stoppers. (For large nodes these must be mounted with the bent part in front and with the node resting at the rail stoppers.)



**Figure E-11:** Railstopper in front of node

12. Connect the front panel with the small power cable and a parallel cable.

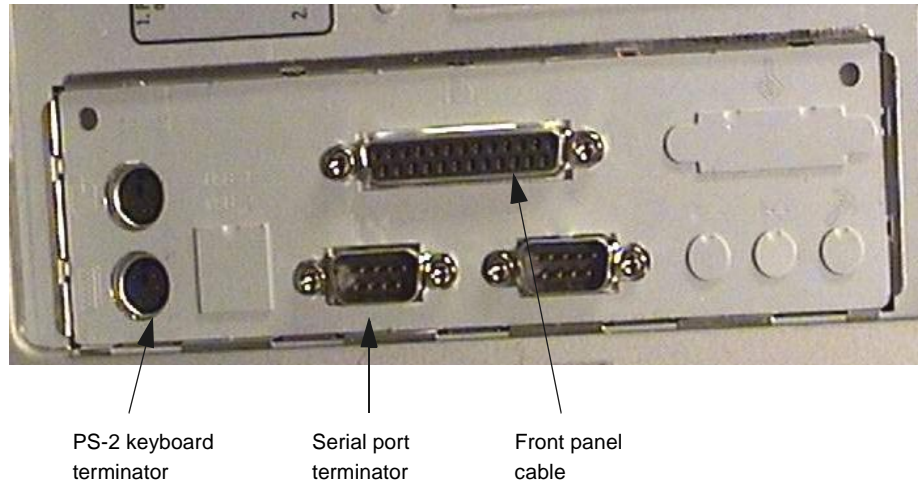


Power cable      Parallel cable

**Figure E-12:** Power and parallel cable connections to front panel



- 13.** Fit the front panel cable, PS2 keyboard terminator and the serial port terminator in the ports shown.



**Figure E-13:** Set front panel cable and keyboard/serial terminators

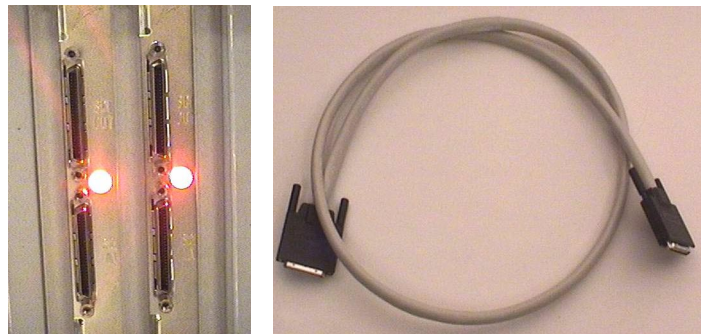
## E-4 Interconnection of nodes

14. Fasten the HUB in an enclosure reserved for infrastructure. Usually for cabinets with 5 enclosures or less it is most practical to use the top enclosure for infrastructure. For cabinets with more than 5 enclosures it has proven practical to continue to use the number 5 enclosure for infrastructure units.



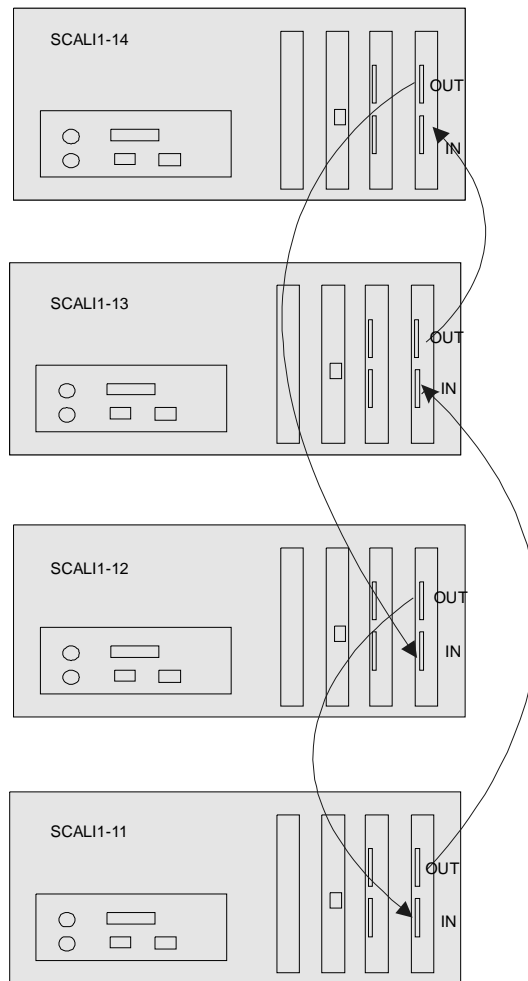
**Figure E-14:** Fasten the HUB

15. Connect the SCI cables to the SCI adapter. The adapter shown below is an adapter with a SCI daughter card used for 2D topologies.



**Figure E-15:** Connect SCI cables

- 16.** The SCI cables must be connected to form the specified topology. The cable must always go from the port marked *out* to another port marked with *in*. Figures E-16 and E-17 shows a schematic drawing and a photo of a real system connected in a ring topology. Note the interleaving to avoid long cables.:



**Figure E-16:** SCI ring topology drawing



**Figure E-17:** SCI ring topology cabling on a real system



## Appendix F                      Related Documentation

---

### F-1 References

- [1] **“MPI: A Message-Passing Interface Standard”**, The Message-Passing Interface Forum, Version 1.1, June 12, 1995, <http://www.mpi-forum.org>.
- [2] **“MPI: The complete Reference: Volume 1, The MPI Core”**, Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker, Jack Dongarra. 2e, 1998. The MIT Press.
- [3] **“MPI: The complete Reference: Volume 2, The MPI Extension”**, William Grop, Steven Huss-Lederman, Ewing Lusk, Bill Nitzberg, William Saphir, Marc Snir. 1998. The MIT Press.
- [4] **“ScaMPI User’s Guide”**. Scali AS, <http://www.scali.com>.
- [5] **“ScaMPI release notes”**. Scali AS, <http://www.scali.com>.
- [6] **“ScaConf User’s Guide”**. Scali AS, <http://www.scali.com>.
- [7] **“The Scali parallel tools environment”**, Draft 1999, Scali AS, <http://www.scali.com>.
- [8] **“CCS: Computing Center Software resource management for networked high-performance computers”**. Paderborn Center of Parallel Computing, <http://www.uni-paderborn.de/pc2>
- [9] **“TotalView Multiprocessor Debugger User’s Guide”**, Etnus Inc., Version 4.1, 2000, [http://www.etnus.com/support/online\\_doc/user\\_guide/index.html](http://www.etnus.com/support/online_doc/user_guide/index.html).
- [10] **“TotalView Multiprocessor Debugger Installation Guide”**, Etnus Inc., Version 4.1, 2000, [http://www.etnus.com/support/online\\_doc/install\\_guide/index.html](http://www.etnus.com/support/online_doc/install_guide/index.html)
- [11] **“VAMPIRtrace for Solarisx86/ScaMPI Installation and User’s Guide”**, Pallas GmbH, Release 1.0 for VAMPIRtrace version 1.5, 1998, <http://www.pallas.de>.
- [12] **“Review of Performance Analysis Tools for MPI Parallel Programs”**, <http://www.cs.utk.edu/~browne/perf-tools-review/>.
- [13] High Performance Debugger Forum, <http://www.ptools.org/hpdf/>.

- [14] NHSE, National HPC Software Exchange - Parallel Tools Library,  
*<http://www.nhse.org/ptlib>*.
- [15] TFCC, IEEE CS Task Force on Cluster Computing,  
*<http://www.dgs.monash.edu.au/~raj कुमार/tfcc>*.
- [16] The Extreme Linux Organisation, *<http://www.extremelinux.org>*.
- [17] **“IEEE Standard for Scalable Coherent Interface(SCI)”**, IEEE Std 1596-1992. Additional information on SCI is available from: <http://www.scizzl.com/>
- [18] **“UCD-SNMP Home page”**, *<http://ucd-snmp.ucdavis.edu/>*

## List of figures

---

2-1	Basic structure of a 4x4 Scali system.....	14
2-2	A 4x8 Scali system boxed in Scali cabinets.....	15
2-3	Open front and rear side of a Scali system in a Scali cabinet .....	15
3-1	A 4x4 cluster with XY coordinates: physical view.....	17
3-2	SCI hardware from Dolphin: PCI adapter, daughter card and cable.....	18
3-3	SCI adapter card jumpers.....	19
3-4	Node with SCI adapter card and daughter card inserted.....	19
3-5	Avoid long cables by interleaving interconnect in each ring .....	20
3-6	Example of 1x8 single ring cabling in a 1x8 cabinet.....	21
3-7	Example of 2x4 two-dimensional system cabling in a 2x4 cabinet.....	22
3-8	Example of 2x4 two-dimensional system cabling in a 1x8 cabinet.....	23
E-1	The Scali system cabinet .....	118
E-2	All major parts of the base assembly .....	119
E-3	Assembly of the casing wheels .....	119
E-4	Assembling backbone to the base.....	120
E-5	Assembly of backbone extension .....	121
E-6	Fasten node enclosures to the backbone.....	121
E-7	Cable hook parts.....	122
E-8	Fasten cable hooks to the backbone .....	122
E-9	Insertion of node in enclosure.....	123
E-10	Node guide adjustment .....	123
E-11	Railstopper in front of node .....	124
E-12	Power and parallel cable connections to front panel.....	124
E-13	Set front panel cable and keyboard/serial terminators.....	125
E-14	Fasten the HUB .....	126
E-15	Connect SCI cables.....	126
E-16	SCI ring topology drawing.....	127
E-17	SCI ring topology cabling on a real system.....	128

Section:

## List of tables

---

1-1	Acronyms and abbreviations .....	10
1-2	Basic terms .....	11
1-3	Typographic conventions .....	11
3-1	Example of cabling a 8 node SCI ring.....	21
3-2	Example cabling of one vertical ring in a 4x4 system (x=[1-4]).....	22
3-3	Example cabling of one horizontal ring in a 4x4 system (y=[1-4]) .....	22
5-1	Node symbols in the main window of the Scali Desktop.....	45
5-2	Menu selections always available .....	46
5-3	Additional menu selections in administrator mode .....	47
5-4	Run menu options .....	48
5-5	System Monitoring options.....	55
5-6	Software menu options .....	57
5-7	Node and software package categories .....	58
6-1	Options in the file ScaSH.conf.....	66
7-1	ScaConf node states .....	74
7-2	ScaConf packages.....	81
7-3	Available server options. Default values are marked with * .....	83
9-1	Hardware installation troubleshooting.....	93
9-2	Software installation troubleshooting.....	94
9-3	Some SCI error types .....	95
9-4	ScaSCI trouble shooting.....	96
9-5	Overview of Scali software products distributed with the SSP.....	101

Section:

<b>C</b>	
CCS resource management software .....	129
<b>F</b>	
Feedback .....	98
<b>P</b>	
Package response files .....	106
<b>R</b>	
References .....	129
<b>S</b>	
ScaConfTool commands .....	81
Scali software platform .....	61
Scali system cabinet .....	117, 118
Scali system install guide .....	9
Scali system overview .....	13
ScaSCI error messages .....	94
ScaSCI trouble shooting .....	96
ScaSH .....	61
SciCards .....	115
SciErrorMessage .....	112
SciIdle .....	115
SciInfo .....	112
SciMonitor .....	112
SciNode .....	113
SciPing .....	111
SSP .....	10
SSP cdrom .....	99
SSP file system layout .....	99
SSP installation program .....	25
<b>T</b>	
Task Force on Cluster Computing .....	130
TotalView Users's Guide .....	129
trouble shooting .....	93
<b>V</b>	
VAMPIRtrace Installation and User's Guide .....	129