



CLE XE™ and XK™ System Administration Guide S-2393-5204xx

Contents

About CLE System Administration Guide.....	13
Cray System Components.....	15
System Management Workstation (SMW).....	16
Cray Linux Environment (CLE).....	16
Service Nodes.....	17
Service Node Options and Services.....	19
Compute Nodes.....	22
Hardware Supervisory System (HSS).....	23
Hardware Supervisory System (HSS) Daemons.....	26
The xtdiscover Command.....	28
Storage Options.....	28
Logical Machines.....	29
Hardware Component Identification.....	29
Physical ID for Cray XE/XK Systems.....	30
Node ID (NID) on Cray XE/XK Systems.....	33
Class Name.....	33
Topology Class.....	34
Manage the System.....	35
Connect the SMW to the Console of a Service Node.....	35
Log on to the Boot Node.....	35
Generic RPM Usage.....	36
Service Node and Compute Node Boot Image Preparation.....	37
Prepare Compute and Service Node Boot Images.....	37
Customize Existing Boot Images.....	39
Create a Boot Image from Existing File System Images.....	40
Change Boot Parameters.....	40
The xtbootsys Command.....	41
Manually Boot the Boot Node and Service Nodes.....	41
Manually Boot the Compute Nodes.....	42
Reboot a Single Compute Node.....	43
Reboot Login or Network Nodes.....	43
Boot a Node or Set of Nodes Using the xtcli boot Command.....	44
Increase the Boot Manager Timeout Value.....	44
Request and Display System Routing.....	45
Bounce Blades Repeatedly Until All Blades Succeed.....	45

Shut Down the System Using the auto.xtshutdown File.....	46
The xtshutdown Command.....	46
Shut Down Service Nodes.....	47
Shut Down the System or Part of the System Using the xtcli shutdown Command.....	47
Stop System Components.....	48
Restart a Blade or Cabinet.....	49
Abort Active Sessions on the HSS Boot Manager.....	50
Display and Change Software System Status.....	50
View and Change the Status of Nodes.....	50
Mark a Compute Node as a Service Node.....	52
Find Node Information.....	52
Display and Change Hardware System Status.....	53
Generate HSS Physical IDs.....	53
Disable Hardware Components.....	54
Enable Hardware Components.....	55
Set Hardware Components to <code>EMPTY</code>	56
Lock Hardware Components.....	56
Unlock Hardware Components.....	56
Perform Parallel Operations on Service Nodes.....	57
Perform Parallel Operations on Compute Nodes.....	57
xtbounce Error Message Indicates Cabinet Controller and Its Blade Controllers Not in Sync.....	58
Power-cycle a Component to Handle Bus Errors.....	58
When a Component Fails.....	58
Capture and Analyze System-level and Node-level Dumps.....	59
cdump and crash Utilities for Node Memory Dump and Analysis.....	59
Dump and Reboot Nodes Automatically.....	60
Enable dumpd.....	61
The <code>/etc/opt/cray-xt-dumpd/dumpd.conf</code> Configuration File.....	61
The dumpd-dbadmin Tool.....	62
The dumpd-request Tool.....	63
Collect Debug Information From Hung Nodes Using the xtnmi Command.....	63
Manage System Access.....	64
Password Management.....	64
Change the Default SMW Passwords.....	64
Change root and crayadm Passwords on Boot and Service Nodes.....	64
Change the root Password on CNL Compute Nodes.....	65
Change the HSS Data Store (MySQL) Password.....	65
Change Default MySQL Passwords on the SDB.....	65

Assign and Change User Passwords.....	68
Logins That Do Not Require Passwords.....	69
User Account Administration.....	69
Add or Modify a Group.....	69
Add or Modify a User Account.....	70
Remove a User or Group Account.....	71
Assign Groups of Compute Nodes to a User Group.....	71
Associate Users with Projects.....	71
Enable LDAP Support for User Authentication.....	72
Load Balancing Across Login Nodes.....	73
Configure the Load Balancer.....	73
Configure lbname on the SMW.....	74
Install the Load Balancer on an External "White Box" Server.....	74
Prevent Login Node Hangs.....	75
Set Disk Quotas for a User on the Cray Local, Non-Lustre File System.....	76
About Modules and Modulefiles.....	77
Configure the Default Programming Environment (PE).....	77
Deny or Allow Services Using the pam_listfile Module in the Shared Root Environment.....	78
Stop a Job Running in Interactive Mode.....	79
Monitor the System.....	80
Display Release Levels and Configuration Information.....	80
Display Boot Configuration Information.....	81
Manage Log Files Using CLE and HSS Commands.....	81
SEC Software for Log Monitoring and Event Processing.....	83
Use cray_pam to Log Failed Login Attempts.....	83
Configure cray_pam to Log Failed Login Attempts.....	84
Check the Status of System Components.....	86
Check the Status of Compute Processors.....	86
Check CNL Compute Node Connection.....	87
Check Link Control Block and Router Errors.....	88
Display System Network Congestion Protection Information.....	89
Monitor the System with the System Environmental Data Collector (SEDC).....	89
HSN Network Metrics Monitoring.....	89
Access Lustre File System Activity Data with the Lustre Monitoring Tool.....	90
Examine Activity on the HSS Boot Manager.....	91
Poll a Response from an HSS Daemon, Manager, or the Event Router.....	91
Validate the Health of the HSS.....	92
Monitor Event Router Daemon (erd) Events.....	92

Monitor Node Console Messages.....	93
View Component Alert, Warning, and Location History.....	93
Display Component Information.....	93
Display Alerts and Warnings.....	96
Clear Component Flags.....	97
Display Error Codes.....	97
Modify an Installed System.....	99
About the Shared-root File System on Service Nodes.....	99
File Specialization.....	99
Visible Shared-root File System Layout.....	100
How Specialization Is Implemented.....	101
Control and Monitor the Shared-root File System.....	102
Manage the System Configuration with the xtopview Tool.....	103
Run xtopview from the SMW while the System is not Booted.....	105
Update Specialized Files From Within the xtopview Shell.....	105
How to Specialize Files.....	106
Specialize a File by Class login.....	106
Specialize a File by Node.....	107
Specialize a File by Node Without Entering xtopview.....	107
Determine Which Files Are Specialized.....	107
Check the Shared-root Configuration.....	109
Verify the Coherency of /etc/init.d Files Across all Shared Root Views.....	109
Clone a Shared-root Hierarchy.....	110
Change the Class of a Node.....	110
Remove Specialization.....	110
Display RCS Log Information for Shared Root Files.....	111
Check Out an RCS Version of Shared Root Files.....	112
List Shared Root File Specification and Version Information.....	112
Perform Archive Operations on Shared Root Files.....	113
Disable Secure Shell (SSH) on Compute Nodes.....	114
Modify SSH Keys for Compute Nodes.....	115
Configure Optional RPMs in the CNL Boot Image.....	116
Configure Memory Control Groups.....	116
Adjust the Memory Control Group Limit.....	117
Disable Memory Control Groups.....	117
Configure cron Services.....	117
Configure cron for the SMW and the Boot Node.....	118
Configure cron for the Shared Root with Persistent /var.....	118

Configure cron for the Shared Root without Persistent /var.....	119
Configure the Node Health Checker (NHC).....	120
Disable the NHC.....	122
Configure NHC to Use SSL.....	122
Configure Node Health Checker Tests.....	122
Guidance for the Accelerator Test.....	125
Guidance for the Application Exited Check and Apinit Ping Tests.....	125
Guidance for the Filesystem Test.....	126
Guidance for the Hugepages Test.....	127
Guidance for the NHC Lustre File System Test.....	127
NHC Control Variables.....	127
Global Configuration Variables that Affect all NHC Tests.....	128
Standard Variables that Affect Individual NHC Tests.....	130
NHC Suspect Mode.....	131
NHC Messages.....	132
If a Login Node Crashes While xtcheckhealth Binaries are Monitoring Nodes.....	133
Recover from a Login Node Crash when a Login Node will not be Rebooted.....	133
Configure Fast Compute Node Reboot.....	134
Boot-node Failover.....	135
Configure Boot-node Failover.....	136
Disable Boot-node Failover.....	137
Compute Node Failover Manager.....	138
Configure SDB Node Failover.....	138
Create Logical Machines for Cray XE/XK Systems.....	139
Create Routable Logical Machines.....	139
Configure a Logical Machine.....	141
Boot a Logical Machine.....	142
Update the Boot Configuration.....	142
Modify Boot Automation Files.....	142
Change the System Software Version to be Booted.....	143
Invoke a Minor Release Switch Within a System Set.....	143
Invoke a Major Release Switch Using Separate System Sets.....	144
Boot a System Set.....	144
Change the Service Database (SDB).....	144
Service Database Tables.....	145
View the SDB with MySQL Commands.....	145
Update SDB Tables.....	147
Update the SDB when Services Change.....	149

Export Lustre with NFSv3.....	150
Configure the NFS client to Mount the Exported Lustre File System.....	151
Enable File-locking for Lustre Clients.....	152
Back Up and Restore Lustre Failover Tables.....	152
Node Attributes.....	153
Initially Set Node Attributes.....	153
Display the Format of the SDB attributes Table.....	155
View and Temporarily Set Node Attributes.....	155
The XTAdmin Database segment Table.....	156
Network File System (NFS).....	157
Configure Ethernet Link Aggregation (Bonding, Channel Bonding).....	157
Configure a Virtual Local Area Network (VLAN) Interface.....	158
Configure Realm-specific IP Addressing (RSIP).....	159
Install and Configure RSIP Using CLEinstall.....	160
Install, Configure, and Start RSIP Clients and Servers.....	161
Add Isolated Service Nodes as RSIP Clients.....	162
Configure a Login Node as an RSIP Server.....	163
Configure RSIP Gateway Address Pools.....	163
RSIP Gateway Address Pool Example.....	166
Configure Multiple RSIP-IP Assign Requests.....	167
Configure RSIP to use RSA-IP Mode.....	169
IP Routes for CNL Nodes in the /etc/routes File.....	170
Update the System Configuration After a Blade Change.....	170
Update the System Configuration after Adding a Blade when the System is not Booted.....	171
Update the System Configuration after Hardware Changes when the System is not Booted.....	172
Update the System Configuration While the System Is Booted.....	172
Reuse One or More Previously-failed HSN Links.....	173
Reuse One or More Previously-failed Blades, Mezzanines, or Cabinets.....	173
Add or Remove a High-speed Network Cable from Service.....	174
Remove a Compute Blade from Service While the System is Running.....	174
Return a Compute Blade into Service.....	175
Cray Lightweight Log Management (LLM) System.....	177
Configure LLM.....	178
State Manager LLM Logging.....	178
Boot Manager LLM Logging.....	179
LLM Configuration Tips.....	179
Change the Location to Log syslog-ng Information.....	179
Manage System Services.....	180

Synchronize Time of Day on Compute Node Clocks with the Clock on the Boot Node.....	180
Add, Start, Restart, or Limit Services.....	180
Add and Start a Service Using RCA.....	181
Indicate Nodes on Which the Service Will Be Started.....	181
Create a Snapshot of /var.....	182
Prevent Login Node Hangs.....	182
Archive the Shared-root Using xtoparchive.....	183
Use Linux Utilities to Back Up Limited Shared-root Configuration Data.....	184
Back Up the Boot Root and Shared Root.....	184
Use the xthotbackup Command to Back Up Boot Root and Shared Root.....	184
Back Up the Boot Root and Shared Root Using dump and restore.....	185
Restore the HSS Database.....	186
Recover from Service Database Failure.....	187
How to Handle Single-node Failures.....	187
Increase the Boot Manager Timeout Value.....	187
The Application Level Placement Scheduler (ALPS).....	189
ALPS Architecture.....	189
ALPS Clients.....	191
ALPS Daemons.....	192
ALPS Configuration Files.....	195
The /etc/sysconfig/alps Configuration File.....	195
The alps.conf Configuration File.....	200
ALPS Global Configuration Parameters.....	203
apsched Configuration Parameters.....	203
apsys Configuration Parameters.....	206
aprun Configuration Parameters.....	207
apstat Configuration Parameters.....	207
Application and Reservation Cleanup Configuration Parameters.....	207
Suspend/Resume.....	208
Resynchronize ALPS and the SDB Command After Manually Changing the SDB.....	209
Display Reserved Resources.....	209
Release a Reserved System Service Protection Domain.....	210
Set a Compute Node to Batch or Interactive Mode.....	210
Manually Start, Stop, or Restart ALPS Daemons on a Specific Service Node.....	211
Manually Clean Up ALPS, PBS, Moab/TORQUE After a Login Node Goes Down.....	211
Verify that ALPS is Communicating with Cray System Compute Nodes.....	212
When an Application Terminates.....	212
aprun Actions.....	214

apinit Actions.....	214
apsys Actions.....	215
Node Health Checker Actions.....	215
Verify Cleanup.....	216
Comprehensive System Accounting.....	217
Interacting with Batch Entry Systems or the PAM job Module.....	217
CSA Configuration File Values.....	218
Configure CSA.....	219
Obtaining File System and Node Information.....	219
Editing the csa.conf File.....	220
Editing Other System Configuration Files.....	220
Creating a CNL Image with CSA Enabled.....	221
Configure CSA Project Accounting.....	221
Configure CSA Job Accounting for non-CCM CNOS Jobs.....	223
Creating Accounting cron Jobs.....	224
csanodeacct cron Job for Login Nodes.....	224
csarun cron Job.....	224
csaperiod cron Job.....	224
Enabling CSA.....	225
Using LDAP with CSA.....	225
Resource Utilization Reporting.....	226
Enable Per-Application RUR.....	227
Enable Per-Job RUR.....	227
Enable Performance-Enhanced RUR.....	228
Disable RUR.....	229
Enable/Disable Plugins.....	229
Enable Plugin Options.....	229
Define Multiple Plugin Versions Using <code>config_sets</code>	230
Limit the Report Generated by an Output Plugin.....	231
RUR Debugging Level.....	231
The energy Data Plugin (Cray XC Series only).....	231
The gpustat Data Plugin.....	233
The kncstats Data Plugin.....	234
The memory Data Plugin.....	234
The taskstats Data Plugin.....	236
The timestamp Data Plugin.....	239
The file Output Plugin.....	239
The llm Output Plugin.....	239

The user Output Plugin.....	240
The database Example Output Plugin.....	240
Create Custom RUR Data Plugins.....	241
Create Custom RUR Output Plugins.....	243
Implement a Site-Written RUR Plugin.....	243
Additional Plugin Examples.....	245
Application Completion Reporting (ACR) to RUR Migration Tips.....	248
Application Resource Utilization (ARU) to RUR Migration Tips.....	249
CSA to RUR Migration Tips.....	250
Dynamic Shared Objects and Cluster Compatibility Mode.....	252
Configure the Compute Node Root Runtime Environment (CNRTE).....	252
Cluster Compatibility Mode.....	254
CCM Preconditions and Configuration Options.....	255
Use DVS to Mount Home Directories on the Compute Nodes for CCM.....	255
Modify CCM and Platform-MPI System Configurations.....	256
Set up Files for the cnos Class.....	257
Link the CCM Prologue/Epilogue Scripts for use with PBS and Moab TORQUE on Login Nodes... ..	257
Create a General CCM Queue and Queues for Separate ISV Applications.....	258
Configure Platform LSF for use with CCM.....	259
Create Custom Resources with PBS.....	260
Create Custom Resources with Moab.....	260
InfiniBand and OpenFabrics Interconnect Drivers.....	262
InfiniBand Uses.....	263
Upper Layer InfiniBand I/O Protocols.....	264
Configure InfiniBand on Service Nodes.....	265
Subnet Manager (OpenSM) Configuration.....	266
Start OpenSM at Boot Time.....	266
Configure IP Over InfiniBand (IPoIB) on Cray Systems.....	267
Configure SCSI RDMA Protocol (SRP) on Cray Systems.....	267
Lustre Networking (LNET) Router.....	268
Configure the LNET Routers.....	269
Specify Service Node LNET routes and ip2nets Directives with Files.....	271
Manually Control LNET Routers.....	271
Configure the InfiniBand Lustre Server.....	272
Configure the LNET Compute Node Clients.....	272
Remove an LNET Router from a Live System.....	273
Tune Lustre Networking.....	276
Router Node Recommended LNET Parameters.....	279

External Server Node Recommended LNET Parameters.....	280
Internal Server (DAL) Recommended LNET Parameters.....	281
DVS Server Node Recommended LNET Parameters.....	281
External Client (CDL) Recommended LNET Parameters.....	281
Additional Information About PTLRPC Parameters.....	281
LNET Feature: Peer Health.....	282
Configure Fine-grained Routing with clcvt.....	282
clcvt Prerequisite Files.....	283
The info.file-system-identifier File.....	283
The client-system.hosts File.....	284
The client-system.ib File.....	286
The cluster-name.ib File.....	286
The client-system.rtrlm File.....	287
Generate ip2nets and routes Information.....	288
Create the persistent-storage File.....	288
Create ip2nets and routes Information for the Compute Nodes.....	288
Create ip2nets and routes information for service node Lustre clients (MOM and internal login nodes).....	289
Create ip2nets and routes information for the LNET router nodes.....	289
Create ip2nets and routes Information for the Lustre Server Nodes.....	290
Access and Manage SMW Services.....	292
Change the Default iDRAC Password.....	292
Configure the SMW to Synchronize to a Site NTP Server.....	292
Enable an Integrated Dell Remote Access Controller (iDRAC6) on a rack-mount SMW.....	293
R815 SMW: Change the BIOS and iDRAC Settings for an iDRAC.....	294
R630 SMW: Create an SMW Bootable Backup Drive.....	299
R815 SMW: Create an SMW Bootable Backup Drive.....	306
Desk-side SMW: Create an SMW Bootable Backup Drive.....	314
(Optional) R630 SMW: Set Up the Bootable Backup Drive as an Alternate Boot Device.....	321
(Optional) R815 SMW: Set Up the Bootable Backup Drive as an Alternate Boot Device.....	323
R630 SMW: Replace a Failed LOGDISK.....	324
R815 SMW: Replace a Failed LOGDISK.....	326
Reboot a Stopped SMW.....	328
SMW Primary Disk Failure Recovery.....	328
Remote Access to SMW.....	329
Start the VNC server.....	329
For Workstation or Laptop Running Linux: Connecting to the VNC Server through an SSH Tunnel, using the vncviewer -via Option.....	330

For Workstation or Laptop Running Linux: Connecting to the VNC Server through an SSH Tunnel.....	330
For Workstation or Laptop Running Mac OS X: Connecting to the VNC Server through an SSH Tunnel.....	331
For Workstation or Laptop Running Windows: Connecting to the VNC Server through an SSH Tunnel.....	331
SMW and CLE System Administration Commands.....	333
System Component States.....	340
Update the Time Zone.....	342
Change the time zone for the SMW and the blade and cabinet controllers on XE systems.....	342
Change the Time Zone on the Boot Root and Shared Root.....	344
Change the Time Zone for Compute Nodes.....	346
PBS Professional Licensing for Cray Systems.....	347
Migrate the PBS Professional Server and Scheduler.....	348
Configure RSIP to the SDB Node.....	349
Add the SDB Node as an RSIP Client to an Existing RSIP Configuration.....	351
Configure Network Address Translation (NAT) IP Forwarding.....	352
Install and Configure a NIC.....	354

About CLE System Administration Guide

CLE XE™ and XK™ System Administration is the replacement document for *Manage System Software for the Cray Linux Environment*. It has a new format that allows it to be easily published as a PDF and also on the new Cray Publication Portal: <http://pubs.cray.com>.

Scope and Audience

This publication covers a wide range of system management topics and is intended for experienced Cray system administrators.

Release Information

This publication includes information, guidance, and procedures for Cray software release CLE5.2.UP04 and supports Cray XE and XK systems. Changes included in this document include:

Added Information

- Cray's Realm Specific Internet Protocol (RSIP) implementation now supports configuring the RSIP client to use Realm Specific Address IP (RSA-IP) mode; see [Configure RSIP to use RSA-IP Mode](#) on page 169
- (Cray XC series systems only) A new Node Health Checker test, DataWarp, has been added; see [Configure Node Health Checker Tests](#) on page 122.
- (Cray XC series systems only) A new command (`xtbiosconf`) to change BIOS settings has been added; see [Modify BIOS Parameters](#).
- (Cray XC series systems only) The RUR `energy` data plugin reports two new data points: `nodes_cpu_throttled` and `nodes_memory_throttled`; see [The energy Data Plugin \(Cray XC Series only\)](#) on page 231.

Revised Information

- (Cray XC series systems only) The calculation method of `nodes_throttled` reported by the RUR `energy` data plug has changed; see [The energy Data Plugin \(Cray XC Series only\)](#) on page 231.

Typographic Conventions

Monospace	Monospaced text indicates program code, reserved words, library functions, command-line prompts, screen output, file names, path names, and other software constructs.
Monospaced Bold	Bold monospaced text indicates commands that must be entered on a command line or in response to an interactive prompt.
<i>Oblique or Italics</i>	<i>Oblique or italicized</i> text indicates user-supplied values in commands or syntax definitions.

Proportional Bold

\ (backslash)

Proportional bold text indicates a graphical user interface window or element.

A backslash at the end of a command line is the Linux[®] shell line continuation character; the shell parses lines joined by a backslash as though they were a single line. Do not type anything after the backslash or the continuation feature will not work correctly.

Alt-Ctrl-f

Monospaced hyphenated text typically indicates a keyboard combination.

Feedback

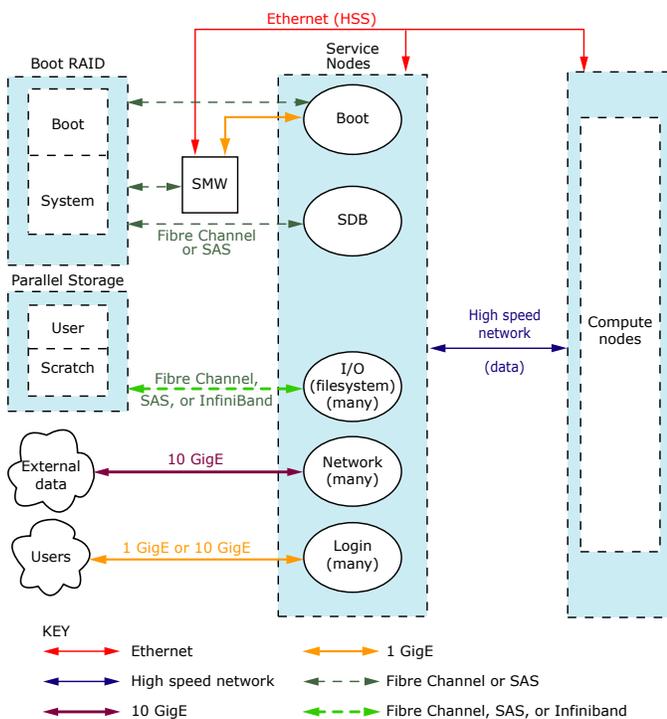
Please provide feedback by visiting <http://pubs.cray.com> and clicking the *Contact Us* button in the upper-right corner, or by sending email to pubs@cray.com.

Cray System Components

Throughout this guide, references to Cray systems mean Cray XE and Cray XK systems unless otherwise noted.

A Cray system is a massively parallel processing (MPP) system comprised of commodity and open-source components combined with custom-designed components to create a system that can operate efficiently at immense scale. Cray systems separate calculation and monitoring functions.

Figure 1. Administrative Components of a Cray System



A Cray system contains operational components plus storage:

- The System Management Workstation (SMW) is the single point of control for system administration.
- The Cray Linux Environment (CLE) operating system is the operating system for Cray systems.
- Service nodes perform the management functions that enable the computations to occur.
- Compute nodes are primarily dedicated to computation.
- The Hardware Supervisory Service (HSS) is an integrated system of hardware and software that monitors components, manages hardware and software failures, controls system startup and shutdown, manages the system interconnection network, and maintains system states.
- RAID is partitioned for a variety of storage functions such as boot RAID, database storage, and parallel and user-file system storage.

A Cray system has the following network components:

- The 10-GigE network is a high-speed Ethernet pipe that provides external NFS access. It connects to the network nodes and is specifically configured to transfer large amounts of data in and out of the system.
- Users access a 1-GigE network server connection to the login nodes. Logins are distributed among the login nodes by a load-leveling service through the Domain Name Service (DNS) that directs them to the least loaded login node.
- Fibre Channel, SAS, or InfinBand networks connect storage to the system components.
- The RAID controllers connect to the SMW through the HSS network. This storage sends log messages to the SMW when a failure affects the ability of the storage system to reliably store and retrieve data.
- The system interconnection network includes custom Cray components that provide high-bandwidth, low-latency communication between all the service nodes and compute nodes in the system. The system interconnection network is often referred to as the *high-speed network (HSN)*.
- The HSS network performs the reliability, accessibility, and serviceability functions. The HSS consists of an internet protocol (IP) address and associated control platforms that monitor all nodes.
- Optional: InfiniBand network connecting LNET service nodes with external Lustre server nodes.

System Management Workstation (SMW)

The System Management Workstation (SMW) is the system administrator's console for managing a Cray system. The SMW is a server that runs a combination of the SUSE Linux Enterprise Server version 11 operating system, Cray developed software, and third-party software. The SMW is also a point of control for the Hardware Supervisory System (HSS). The HSS data is stored on an internal hard drive of the SMW. For information about installing the SMW release software, see *Installing Cray System Management Workstation (SMW) Software (S-2480)*.

An administrator logs on to an SMW window on the console to perform SMW functions. From the SMW, an administrator can log on to a disk controller or use a web-browser-based interface from the SMW to configure a RAID controller, Fibre Channel switch, or SAS switch. An administrator can log on to the boot node from the SMW as well. From the SMW, it is not possible to log on directly (ssh) to any service node except the boot node.

Most system logs are collected and stored on the SMW. The SMW plays no role in computation after the system is booted. From the SMW an administrator can initiate the boot process, access the database that keeps track of system hardware, analyze log messages, and perform standard administrative tasks.

Cray Linux Environment (CLE)

The Cray Linux Environment (CLE) operating system includes Cray's customized version of the SUSE Linux Enterprise Server (SLES) 11 Service Pack 3 (SP3) operating system, with a Linux kernel. This full-featured operating system runs on the Cray system's service nodes. The Cray compute nodes run a kernel developed to provide support for application execution without the overhead of a full operating-system image. In the compute node root runtime environment (CNRTE), compute nodes have access to the service node shared root (via chroot) such that compute nodes can access the full features of a Linux environment.

CLE commands enable administrators to perform administrative functions on the service nodes to control processing. The majority of CLE commands are launched from the boot node, making the boot node the focal point for CLE administration.

Service Nodes

Service nodes perform the functions needed to support users, administrators, and applications running on compute nodes. Service nodes run the CLE operating system. The administrator commands for these nodes are standard Linux commands and Cray system-specific commands. Service node login is accomplished by first logging on to the boot node through the SMW console and then from the boot node to any service node.

Service nodes can be specialized and categorized by the services that run on them, as determined by the system administrator. Configuration information in the service database on the SDB node determines the functions of the other nodes and services, such as where a batch subsystem runs. In small configurations, some services can be combined on the same node: for example, the `sdb` and `syslog` services can both run on the same node.

Services are started either system-wide or on specific nodes. They can be started during the boot process or later on specific nodes of a running system. How services are started depends on the type of service.

Service nodes, unlike compute nodes, are generally equipped with Peripheral Component Interconnect (PCI) protocol card slots to support external devices.

Service nodes run a full-featured version of the Linux operating system. Service node kernels are configured to enable Non-Uniform Memory Access (NUMA), which minimizes traffic between sockets by using socket-local memory whenever possible.

System management tools are a combination of Linux commands and Cray system commands that are analogous to standard Linux commands but operate on more than one node. After the system is up, an administrator possessing the correct permissions can access any service node from any other service node.

Boot Node

The boot node is used to manage files, add users, and mount and export the shared-root file system to the rest of the service nodes. These shared-root files are mounted from the boot node as read-only. An administrator logs on to the boot node through the SMW console.

The boot node is the first node to be booted, which is done through the boot node's blade control processor (see [Blade Controllers and Cabinet Controllers](#)). The boot node is typically located on a system blade. System blades have only one node with two PCIe slots (node 1). Of the remaining three nodes on the blade, node 0 has no PCIe I/O connectivity and nodes 2 and 3 have the typical configuration of one PCIe slot per node. There can be only one dual-slot node per blade.

Two boot nodes can be configured per system or per partition, one primary and one for backup (secondary). The two boot nodes must be located on different blades. When the primary boot node is booted, the backup boot node also begins to boot. But the backup boot node boot process is suspended until a primary boot-node failure event is detected. For information about configuring boot-node failover, see [Configure Boot-node Failover](#) on page 136.

Boot Root File System

The boot node has its own root file system, `bootroot`, which is created on the boot RAID during installation. The boot RAID is installed and configured from the SMW before booting the boot node. The boot node mounts `bootroot` from the boot RAID.

Shared Root File System

Cray systems have a root file system that is distributed as a read-only shared file system among all the service nodes except the boot node. Each service node has the same directory structure, which is made up of a set of

symbolic links to the shared-root file system. For most files, only one version of the file exists on the system; therefore, modifying the single copy affects all service nodes. This makes the administration process similar to that of a single system. The shared-root file system is managed from the boot node through the `xtopview` command (see [Manage the System Configuration with the xtopview Tool](#) on page 103).

The shared-root file system has a modified directory structure for `/etc` that enables unique files for a specific node or class of nodes (i.e., nodes of a certain type). When a service node boots, the directory hierarchy from the specialized `/etc` overlays the `/etc` that the service node mounted from the rest of the shared root. The specialized `/etc` has symbolic links that point to the files whether they were created as default (for all service nodes), class specific (for a class of nodes), or node specific (for a single node). A service node uses a node-specialized file unless there is a class-specialized file, yet if there are neither node-specialized nor class-specialized files, it will use the default file. For further information, see [About the Shared-root File System on Service Nodes](#) on page 99.

Service Database (SDB) Node

The SDB node hosts the service database, a MySQL database that resides on a separate file system on the boot RAID. The SDB is accessible to every service node. The SDB provides a central location for storing information so that it does not need to be stored on each node. The SDB is accessible, with correct authorizations, from any service node after the system is booted.

The SDB stores the following information:

- Global state information of compute processors. This information is used by the Application Level Placement Scheduler (ALPS), which allocates compute processing elements for compute nodes running Compute Node Linux (CNL).
- System configuration tables that list and describe processor attribute and service information.

The SDB node is the second node that is started during the boot process.

Two SDB nodes can be configured per system or per partition, one primary and one for backup (secondary). The two SDB nodes must be located on different system blades. For more information, see [Configure SDB Node Failover](#).

Syslog Node

By default, the boot node forwards syslog traffic from the service nodes to the SMW for storage in log files. An optional syslog node may be specified (see the `SMWinstall.conf(5)` man page); however, this service node must be provisioned and configured to be able to reach the SMW directly over an attached Ethernet link.

Login Nodes

Users log on to a login node, which is the single point of control for applications that run on the compute nodes. Users do not log on to compute nodes.

The administrator can use the Linux `lbnamed` load balancer software provided to distribute user logins across login nodes (see [Configure the Load Balancer](#) on page 73). The number of login nodes depends upon the installation and user requirements. For typical interactive usage, a single login node handles 20 to 30 batch users or 20 to 40 interactive users with double this number of user processes.



CAUTION: Login nodes, as well as other service nodes, do not have swap space. If users consume too many resources, Cray service nodes can run out of memory. When an out of memory condition occurs, the node can become unstable or may crash. System administrators should take steps to manage system resources on service nodes. For example, resource limits can be configured using the `pam_limits`

module and the `/etc/security/limits.conf` file. For more information, see the `limits.conf(5)` man page.

Network Nodes

Network nodes connect to the external network with a 10-GigE card. These nodes are designed for high-speed data transfer.

I/O Service Nodes

There are three types of I/O service nodes: direct-attached Lustre (DAL) nodes, LNET nodes that connect via IB to an external Lustre file server, and Cray Data Virtualization Service (Cray DVS) nodes. (Cray DVS) is a parallel I/O forwarding service that provides for transparent use of multiple file systems on Cray systems with close-to-open coherence, much like NFS. Cray DVS servers run on an I/O node. Cray DVS servers act as external file system clients projecting the external file systems to service and compute node clients within the system.

Service Node Options and Services

Service nodes provide the following options and services to support users, administrators, networking, storage, and applications running on compute nodes.

Resiliency Communication Agent (RCA)

The RCA is the message path between the CLE operating system and the HSS. The RCA runs on all service nodes and CNL compute nodes.

The RCA consists of a kernel-mode driver and a user-mode daemon on CLE. The Gemini chip and the L0 controller on each blade provide the interface from the RCA to the HSS through application programming interfaces (APIs). The RCA driver, `rca.ko`, runs as a kernel-loadable module for the service partition. On CNL compute nodes, the RCA operates through system calls and communicates with the HSS to track the heartbeats of any registered programs and to handle event traffic between the HSS and the applications registered to receive events. The RCA driver starts as part of the kernel boot, and the RCA daemon starts as part of the initialization scripts.

Lustre File System

Cray systems running CLE support the Lustre file system, which provides a high-performance, highly scalable, POSIX-compliant shared file system. Lustre file systems are configurable to operate in the most efficient manner for the I/O needs of applications, ranging from a single metadata server (MDS) and object storage target (OST) to a single MDS with up to 128 OSTs. User directories and files are shared and are globally visible from all compute and service node Lustre clients.

For additional information, see the *Manage Lustre for the Cray Linux Environment (CLE) (S-0010)* and *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*

Cray Data Virtualization Service (Cray DVS)

The Cray Data Virtualization Service (Cray DVS) is a parallel I/O forwarding service that provides for transparent use of multiple file systems on Cray systems with close-to-open coherence, much like NFS.

For additional information, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)* and *Introduction to Cray Data Virtualization Service (S-0005)*

Application Level Placement Scheduler (ALPS) for Compute Nodes

ALPS provides application placement, launch, and management functionality and cooperates with third-party batch systems for application scheduling on compute nodes running CNL. A third-party batch system (such as PBS Professional, Moab/TORQUE, or Platform LSF) makes the policy and scheduling decisions, and ALPS provides a mechanism to place and launch the applications contained within batch jobs. ALPS also supports placement and launch functionality for interactive applications.

ALPS is automatically loaded as part of the CNL environment when booting CNL. The RCA starts the ALPS `apinit` daemon on the compute nodes.

An Extensible Markup Language (XML) interface is provided by ALPS for communication with third-party batch systems. This interface is available through use of the `apbasil` client. ALPS uses application resource reservations to guarantee resource availability to batch system schedulers.

The ALPS application placement and launch functionality is provided for applications executing on compute nodes only; ALPS does not provide placement and launch functionality for service nodes. Only one application can be placed per node; two different executables cannot be run on the same node at the same time. When a job is running, the `aprun` process interacts with ALPS to keep track of the processors that the job uses.

For more information about ALPS, see [The Application Level Placement Scheduler \(ALPS\)](#).

Job Launch Commands

Users run applications from a login node and use the `aprun` command to launch CNL applications. The `aprun` command provides options for automatic and manual application placement. With automatic job placement, `aprun` distributes the application instances on the number of processors requested, using all of the available nodes.

With manual job placement, users can control the selection of the compute nodes on which to run their applications. Users select nodes on the basis of desired characteristics (*node attributes*), allowing a placement scheduler to schedule jobs based on the node attributes. To provide the application launcher with a list of nodes that have a particular set of characteristics (attributes), the user invokes the `cselect` command to specify node-selection criteria. `cselect` uses these selection criteria to query the table of node attributes in the SDB and returns a node list to the user based on the results of the query. For an application to be run on CNL compute nodes, the nodes satisfying the requested node attributes are passed by the `aprun` utility to the ALPS placement scheduler as the set of nodes from which to make an allocation. For detailed information about ALPS, see [The Application Level Placement Scheduler \(ALPS\)](#).

For more information about the `aprun` and `cselect` commands, see the `aprun(1)` and `cselect(8)` man pages.

Node Health Checker (NHC)

NHC is automatically invoked by ALPS upon the termination of an application. ALPS passes a list of CNL compute nodes associated with the terminated application to NHC. NHC performs specified tests to determine if compute nodes allocated to the application are healthy enough to support running subsequent applications. If not, it removes any compute nodes incapable of running an application from the resource pool. The CLE installation and upgrade processes automatically install and enable NHC software; there is no need to change any installation configuration parameters or issue any commands. To configure NHC tests, see [Configure the Node Health Checker \(NHC\)](#) on page 120.

Optional Workload Management (Batch) System Software

For information about optional batch systems software products for Cray systems, see the following websites.

- Moab and TORQUE - See <http://www.adaptivecomputing.com>
- PBS Professional - See <http://www.pbsworks.com>
- Platform LSF - See <http://www.platform.com>
- Slurm Workload Manager - See <http://www.schedmd.com/slurmdocs/slurm.html>

Cluster Compatibility Mode (CCM)

Cluster Compatibility Mode (CCM) provides the services needed to run most cluster-based independent software vendor (ISVs) applications "out of the box." CCM is tightly coupled to the workload management system. It enables users to execute cluster applications alongside workload-managed jobs running in a traditional MPP batch or interactive queue. Support for dynamic shared objects and expanded services on CNL compute nodes, using the compute node root runtime environment (CNRTE), provide the services to compute nodes within the cluster queue. Essentially, CCM uses the batch system to logically designate part of the Cray system as an emulated cluster for the duration of the job. For more information about CCM, see [Cluster Compatibility Mode](#) on page 254.

IP Implementation

Ethernet interfaces handle IP connectivity to external components. Both IPv4 and IPv6 are supported; IPv4 is the default.

The IPv6 capability is limited to the Ethernet interfaces and `localhost`. Therefore, IPv6 connectivity is limited to service nodes that have Ethernet cards installed. Routing of IPv6 traffic between service nodes across the HSN is not supported.

Realm-Specific IP Addressing (RSIP)

Realm-Specific IP Addressing (RSIP) allows compute nodes and the service nodes to share the IP addresses configured on the external Gigabit and 10 Gigabit Ethernet interfaces of network nodes. By sharing the external addresses, the system administrator can rely on the system's use of private address space and does not need to configure compute nodes with addresses within the site's IP address space. The external hosts see only the external IP addresses of the Cray system. For further information, see [Configure Realm-specific IP Addressing \(RSIP\)](#) on page 159.

Repurpose Compute Nodes as Service Nodes

Some services on Cray systems have resource requirements or limitations (i.e., memory, processing power, response time) that an administrator can address by configuring a dedicated service node, such as a Cray Data Virtualization Service (Cray DVS) node or a batch system management (MOM) node. On Cray systems, service I/O node hardware (on a service blade) is equipped with Peripheral Component Interconnect (PCI) protocol card slots to support external devices. Compute node hardware (on a compute blade) does not have PCI slots. For services that do not require external connectivity, the administrator can configure the service to run on a single, dedicated compute node and avoid using traditional service I/O node hardware.

When a node on a compute blade is configured to boot a service node image and perform a service node role, that node is referred to as a *repurposed compute node*.

For additional information, see the section on repurposing compute nodes in *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*.

Persistent /var Directory

The system administrator can configure a persistent, writable /var directory on each service node served with NFS. The boot node has its own root file system and its own /var directory; the boot node /var is not part of the NFS exported /snv file system.

Because the Cray system root file system is read-only, some subdirectories of /var are mounted on tmpfs (memory) and not on disk. Persistent /var retains the contents of /var directories between system boots. If `persistent_var=yes` in `CLEinstall.conf`, `CLEinstall` configures the correct values for `VAR_SERVER`, `VAR_PATH`, and `VAR_MOUNT_OPTIONS` in the `/etc/sysconfig/xt` file during installation so the service nodes will NFS mount the proper path at boot time.

Boot scripts and the `xtopview` utility respect these configuration values and mount the correct /var directory.

For more information, see the *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*.

The /etc/hosts File

The host file on the boot node is for the HSN and external hosts accessible from login and network nodes. The hosts file on the SMW is for the HSS network.

The `/etc/hosts` file on the boot node maps IP addresses to node IDs on the system interconnection network. The file can also contain aliases for the physical ID location of the system interconnection network components and class names. The upper octets typically range from 10.128.0.0 to 10.131.255.255. Lower octets for nodes are derived from their NID. The NID is a sequential numbering of nodes from cabinet 0 up.

The installation and upgrade process modifies the `/etc/hosts` file on the boot root to include all service and compute nodes and any hostname aliases for special service nodes.

The `/etc/hosts` file on the SMW contains `physIDs` (physical IDs that map to the physical location of HSS network components), such as the blade and cabinet controllers.

The default system IP addresses are shown in the *Installing Cray System Management Workstation (SMW) Software (S-2480)*.

The service configuration table (`service_config`) in the SDB XAdmin database provides a line for each service IP address of the form, where `SERV1` and `SERV2` are the service names in the `service_config` table:

1.2.3.1	SERV1
1.2.3.2	SERV2

Each time CLE software is updated or upgraded, `CLEinstall` verifies the content of `/etc/opt/cray/sdb/node_classes` and modifies `/etc/hosts` to match the configuration specified in the `CLEinstall.conf` file. For additional detail about how `CLEinstall` modifies the `/etc/hosts` file, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*.

Compute Nodes

Cray system compute nodes run the CNL compute node operating system. CNL is a lightweight compute node operating system. It includes a run-time environment based on the SLES 11 SP3 distribution, with a Linux kernel and Cray specific modifications. Device drivers for hardware not supported on Cray systems were eliminated from the kernel. CNL features scalability; only the features required to run high-performance computing applications are available on CNL compute nodes. Other features and services are available from service nodes. Cray has configured and tuned the kernel to minimize processing delays caused by inefficient synchronization. CNL compute node kernels are configured to enable Non-Uniform Memory Access (NUMA), which minimizes traffic between sockets by using socket-local memory whenever possible. CNL also includes a set of supported system calls and standard networking.

Cray XE/XK systems also include NVIDIA GPGPU (General Purpose Graphics Processing Unit) processors. For optimal use of compute node resources in mixed Cray XE systems with Cray XK compute blades, system administrators can elect to assign Cray XK compute nodes to a batch queue, allowing users to make reservations for either scalar-only or accelerator-based compute node pools.

Several libraries and compilers are linked at the user level to support I/O and communication service. Cray, PGI, PathScale, and the GNU Compiler Collection (GCC) C, C++, and Intel compilers are supported. For information about using modulefiles and configuring the default programming environment, see [About Modules and Modulefiles](#) and [Configure the Default Programming Environment \(PE\)](#). For information about the libraries that Cray systems host, see the *Cray Application Developer's Environment User's Guide (S-2396)*.

The Resiliency Communication Agent (RCA) is the message path between the CLE operating system and the HSS. Its daemon, `rcad-svcs`, runs on all service nodes and CNL compute nodes.

The Application Level Placement Scheduler (ALPS) handles application launch, monitoring, signaling and coordination of batch job processing with third-party batch systems. If ALPS is running on the system, use the `xtnodestat` command to report job information.

User-level BusyBox commands are functional on CNL compute nodes. For information about supported commands and options, see the `busybox(1)` man page.

The following administrator-level busybox commands and associated options are functional on CNL compute nodes:

- `dmesg -c -n -s`
- `fuser -m -k -s -4 -6 -SIGNAL`
- `logger -s -t -p`
- `mount -a -f -n -o -r -t -w`
- `ping -c -s -q`
- `sysctl -n -w -p -a -A`
- `umount -a -n -r -l -f -D`

A compute-node failure affects only the job running on that node, the rest of the system continues running.

Hardware Supervisory System (HSS)

HSS is an integrated system of hardware and software that monitors the hardware components of the system and proactively manages the health of the system. HSS communicates with nodes and management processors over an internal (private) Ethernet network that operates independently of the system interconnection network.

HSS data is stored on an internal hard drive of the System Management Workstation (SMW).

HSS includes the following components:

- HSS network
- HSS command-line interface (CLI)
- Blade and cabinet control processors
- Network Time Protocol (NTP) server
- Event router daemon (`erd`)
- HSS daemons
- Cray system network routing utility (`rtr`)
- Various logs

HSS Network

The HSS Ethernet network connects the SMW to all cabinets in the system via a series of switches. The SMW performs reliability, accessibility, and serviceability tasks over this network. The SMW manages the HSS network.

For a list of SMW commands, see [SMW and CLE System Administration Commands](#) on page 333.

HSS Command Line Interface

HSS has a command-line interface (`xtcli`) to manage and monitor the system from the SMW. For complete usage information, see the `xtcli(8)man` page.

For a list of all HSS system administration commands, see [SMW and CLE System Administration Commands](#) on page 333.

Blade Controllers and Cabinet Controllers

A blade control processor (*blade controller*) is hierarchically the lowest component of the monitoring system. In some contexts, the blade controller is also referred to as a *slot*. One blade controller resides on each compute blade and service blade, monitoring only the nodes and ASICs. It provides access to status and control registers for the components of the blade. The blade controller also monitors the general health of components, including items such as voltages, temperature, and other failure indicators. A version of Linux optimized for embedded controllers runs on each blade controller.

In some contexts, the blade controller is referred to as a slot. On Cray XE and Cray XK systems, the blade controller is referred to as the *LO*.

Each cabinet has a cabinet control processor (*cabinet controller*) that monitors and controls the cabinet power and cooling equipment and communicates with all blade controllers in the cabinet. It sends a periodic heartbeat to the SMW to indicate cabinet health.

The cabinet controller connects to the chassis controller and in turn the chassis controller connects to the blade controllers (via the backplane) on each blade by Ethernet cable and routes HSS data to and from the SMW. The cabinet controller runs embedded Linux.

The monitoring system operates by periodic heartbeats. Processes send heartbeats within a time interval. If the interval is exceeded, the system monitor generates a fault event that is sent to the state manager. The fault is recorded in the event log, and the state manager sets an alert flag for the component (blade controller or cabinet controller) that spawned it.

The cabinet and blade controllers use `ntpclient` to keep accurate time with the SMW.

The administrator can dynamically configure the cabinet controller system daemon and the blade controller system daemon with the `xtdaemonconfig --daemon_name` command.

For more information, see the `xtdaemonconfig(8)` man page.



CAUTION: There is no NV write protection feature on the cabinet and blade controllers; do not assume the write protection functionality on the cabinet controller front panel display will protect the NV memory on the cabinet and blade controllers.

Network Time Protocol (NTP) Server

The SMW workstation is the primary NTP server for the Cray system. The blade controllers use the HSS network to update themselves according to the NTP protocol. To change the NTP server, see [Configure the SMW to Synchronize to a Site NTP Server](#) on page 292.

HSS Daemons

HSS daemons act as administrators for most of the HSS components. They exchange information with `erd` to subscribe to, and inject, events into the HSS system. See [Hardware Supervisory System \(HSS\) Daemons](#) on page 26 for further details, including a complete list of daemons.

Cray System Network Routing Utility

The `rtr` command performs a variety of routing-related tasks for the High Speed Network (HSN). Tasks include:

- Requests that the HSN be routed and initialized
- Verifies that routes can be generated for the current configuration
- Verifies that generated routes are free of cyclic dependencies
- Dumps out a variety of routing-related and link-related information

The `rtr` command is also invoked as part of the `xtbootsys` process. For more information, see the `rtr(8)` man page.

Log Files

Event Logs The event router records events to the event log in the `/var/opt/cray/log/event-yyyymmdd` file. When the log grows beyond a reasonable size, it turns over and its contents are stored in a numbered file in the directory.

Boot Logs The `/var/opt/cray/log/session-id` directory is a repository for files collected by commands such as `xtbootsys`, `xtconsole`, `xtconsumer`, and `xtnetwatch` for the currently booted session. To determine the current `sessionid`, see the `xtsession(8)` man page.

NOTE: A symbolic link will be created from `/var/opt/cray/log/partition-current` to the currently booted session directory.

Dump Logs The `/var/opt/cray/dump` directory is a repository for files collected by the `xtdumpsys` command. It contains time-stamped dump files.

For more information, see [Manage Log Files Using CLE and HSS Commands](#) on page 81.

Hardware Supervisory System (HSS) Daemons

HSS daemons exchange information with the event router daemon (`erd`). They are located in `/opt/cray/hss/default/etc` and are started by running the `/etc/init.d/rsms start` command. HSS daemons can be configured dynamically by executing the `xtdaemonconfig` command.

Key HSS daemons include:

- State manager
- Boot manager
- System environmental data collections (SEDC) manager
- Node ID (NID) manager
- Flash manager

For a list of all HSS daemons, see the `xtdaemonconfig(8)` man page.

State Manager

HSS maintains the state of all components that it manages. The state manager, `state_manager`, runs on the SMW and uses a relational database (also referred to as the *HSS database*) to read and write the system state. The state manager keeps the database up-to-date with the current state of components and retrieves component information from the database when needed. Thus, the dynamic system state persists between SMW boots. The state manager uses the Lightweight Log Manager (LLM). The log data from state manager is written to `/var/opt/cray/log/sm-yyyyymmdd`. The default setting for state manager is to enable LLM logging. The state manager performs the following functions:

- Updates and maintains component state information (See [System Component States](#) on page 340)
- Monitors events to update component states
- Detects and handles state notification upon failure
- Provides state and configuration information to HSS applications to prevent interference with other applications working on the same component

The state manager listens to the `erd`, records changes of states, and shares those states with other daemons.

The Event Router (`erd`)

HSS functions are event-driven. The event router daemon, `erd`, is the root of HSS. It is a system daemon that runs on the SMW, cabinet controllers, and blade controllers. The SMW runs a separate thread for each cabinet controller. The cabinet controller runs a separate thread for each blade controller. HSS managers subscribe to events and inject events into the HSS system by using the services of the `erd`. The event router starts as each of the devices (SMW, cabinet controller, blade controller) are started.

When the event router on the SMW receives an event from either a connected agent or from another event router in the hierarchy, the event is logged and then processed. The `xtcli` commands, which are primary HSS control commands, also access the event router to pass information to the managers.

The `xtconsumer` command monitors the `erd`. The `xtconsole` command displays all node console messages.

Boot Manager

The boot manager, `bootmanager`, runs on the SMW. It controls the acts of placing kernel data into node memories and requesting that they begin booting.

During the boot process, the state manager provides state information that allows the nodes to be locked for booting. After the nodes boot, the state manager removes the locks and notifies the boot manager. The boot manager logging facility includes a timestamp on log messages.

System Environmental Data Collections (SEDC) Manager

The System Environment Data Collections (SEDC) manager, `sedc_manager`, monitors the system's health and records the environmental data and status of hardware components such as power supplies, processors, temperature, and fans. SEDC can be set to run at all times or only when a client is listening. The SEDC configuration file provided by Cray has automatic data collection set as the default action.

The SEDC configuration file (`/opt/cray/hss/default/etc/sedc_srv.ini` by default) configures the SEDC server. In this file, the administrator can create sets of different configurations as groups so that the blade and cabinet controller daemons can scan components at different frequencies. The `sedc_manager` sends out the scanning configuration for specific groups to the cabinet and blade controllers and records the incoming data by group.

For information about configuring the SEDC manager, see *Using and Configuring System Environment Data Collections (SEDC) (S-2491)*.

Node ID (NID) Manager

The NID manager, `nid_mgr`, runs on the SMW and provides a NID mapping service for the rest of the HSS environment. Along with the ability to assign NIDs automatically, the `nid_mgr` supports a mechanism that allows an administrator to control the NID assignment; this is useful for handling unique configurations. Administrator-controlled NID assignment is accomplished through the NID assignment file, `nids.ini`.



CAUTION: The `nids.ini` file can have a major impact on the functionality of a Cray system and should only be used or modified at the recommendation of Cray support personnel. Setting up this file incorrectly can make the Cray system unroutable.

Typically after a NID mapping is defined for a system, this mapping is used until some major event occurs, such as a hardware configuration change (see [Update the System Configuration After a Blade Change](#) on page 170). This may require the NID mapping to change, depending on the nature of the configuration change. Adding cabinets to the ends of rows does not typically result in a new mapping; however, adding rows most likely results in a new mapping. If the configuration change is such that the topology class of the system is changed, a new NID mapping is required. Otherwise, the NID mapping remains static.

The `nid_mgr` generates a list of mappings between the physical location and Network Interface Controller ID (NIC ID) and distributes this information to the blade controllers.

Since the operating system always uses NIDs, the HSS converts these to NIC IDs when sending them onto the HSS network and converts them to NIDs when forwarding events from the HSS network to a node.

Flash Manager

The flash manager, `fm`, runs on the SMW and is used to transfer an updated L0 and L1 controller system image to a specified target to update the firmware in its L0 and L1 controllers and to program processor Programmable Intelligent Computer (PIC) firmware.



WARNING: The `fm` command is intended for use by Cray Service Personnel only; improper use of this restricted command can cause serious damage to the computer system

The `xtflash` system administrator command uses `fm` to flash memory on one or more L0 and L1 controllers. The `xtflash` command updates only out-of-date L0 and L1 controllers. For more information, see the `xtflash(8)` man page.

The `xtdiscover` Command

The `xtdiscover` command automatically discovers the hardware components on a Cray system and creates entries in the system database to reflect the current hardware configuration. The `xtdiscover status` command identifies missing or nonresponsive cabinets, empty or nonfunctioning slots, the blade type (service or compute) in each slot, and the CPU type and other attributes of each node in the system. The `xtdiscover` command and the state manager ensure that the system status represents the real state of the hardware. When `xtdiscover` has finished, a system administrator can use the `xtcli` command to display the current configuration. No previous configuration of the system is required; the hardware is discovered and made available. Modifications can be made to components after `xtdiscover` has finished creating entries in the system database.

The `xtdiscover` interface steps a system administrator through the discovery process.

The `xtdiscover.ini` file enables an administrator to predefine values such as topology class, cabinet layout, and so on. A template `xtdiscover.ini` file is installed with the SMW software. The default location of the file is `/opt/cray/hss/default/etc/xtdiscover.ini`.

NOTE: When `xtdiscover` creates a default partition, it uses `c0-0c0s0n1` as the default for the boot node and `c0-0c0s2n1` as the default SDB node.

The `xtdiscover` command does not use or configure the Cray High Speed Network (HSN). The HSN configuration is done when booting the system with the `xtbootsys` command.

If there are changes to the system hardware, such as adding a new cabinet or removing a blade and replacing it with a blade of a different type (e.g., a service blade replaced with a compute blade), then `xtdiscover` must be executed again, and it will perform an incremental discovery of the hardware changes without disturbing the rest of the system.

For more information, see the `xtdiscover(8)` man page.

Storage Options

All Cray systems require RAID storage. RAID storage consists of one or more physical RAID subsystems; a RAID subsystem is defined as a pair of disk controllers and all disk modules that connect to the controllers.

Functionally, there are two types of RAID subsystems: system RAID (also referred to as *boot RAID*) and file system RAID. The system RAID stores the boot image and system files and is also partitioned for database functionality, while the file system RAID stores user files.

File system RAID subsystems use the Lustre file system. Lustre offers high performance scalable parallel I/O capabilities, POSIX semantics, and scalable metadata access. For more information on Lustre file system configuration, see *Manage Lustre for the Cray Linux Environment (CLE) (S-0010)*.

Cray offers RAID subsystems from two vendors: DataDirect Network (DDN) and NetApp. All DDN RAID subsystems function as dedicated file system RAID, while NetApp RAID subsystems can function as dedicated

file system RAID, a dedicated system RAID, or a combination of both. Different controller models support Fibre Channel (FC), Serial ATA (SATA), and Serial Attached SCSI (SAS) disk options. In addition to vendor solutions for file system RAID, Cray offers the Lustre File System by Cray (CLFS) and Sonexion, an integrated file system, software and storage product.

RAID devices are commonly configured with zoning so that only appropriate service nodes see the disk devices (LUNs) for the services that will be provided by each node; this is done in order to reduce the possibility of accidental or unauthorized access to LUNs.



CAUTION: Because the system RAID disk is accessible from the SMW, the service database (SDB) node, the boot node, and backup nodes, it is important to never mount the same file system in more than one place at the same time. Otherwise, the Linux operating system will corrupt the file system.

For more information about configuring RAID, see the documentation for the site-specific RAID setup.

Logical Machines

The system administrator can subdivide a single Cray system into two or more logical machines (partitions), which can then be run as independent systems. An operable logical machine has its own compute nodes and service nodes, external network connections, boot node, and SDB node. Each logical machine can be booted and dumped independently of the other logical machines. Once booted, a logical machine appears as a normal Cray system to the users, limited to the set of hardware included for the logical machine.

The HSS is common across all logical machines. Because logical machines apply from the system interconnection network layer and up, the HSS functions continue to behave as a single system for power control, diagnostics, low-level monitoring, and so on.

In addition:

- Each logical machine must be routable for jobs to run.
- Cray recommends that only one logical machine is configured per cabinet. That way, if a cabinet is powered down, only one logical machine is affected.
- A logical machine can include more than one cabinet.
- A job is limited to running within a single logical machine.
- Although the theoretical maximum of allowable logical machines per physical Cray system is 31 logical machines (as `p0` is the entire system), the system administrator must consider the hardware requirements to determine a practical number of logical machines to configure.
- No two logical machines can use the same components; therefore, if a system is partitioned into logical machines, `p0` is no longer a valid reference and should be removed or deactivated.
- Only a single instance of SMW software may be run.
- Boot and routing commands affect only a single logical machine.

To create logical machines, see [Create Logical Machines for Cray XE/XK Systems](#) on page 139.

Hardware Component Identification

System components (nodes, blades, chassis, cabinets, etc.) are named and located by node ID, IP address, physical ID, or class number. Some naming conventions are specific to CLE.

Component naming does not change based on the number of cores or processors. Applications start on CPU 0 and are allocated to CPUs either on the same or different processors.

Physical ID for Cray XE/XK Systems

The physical ID identifies the cabinet's location on the floor and the component's location in the cabinet as seen by the HSS. Descriptions within the table below assume the reader is facing the front of the system cabinets.

Gemini MMR space **must** use a "g" name, possibly with an NIC identifier. Processor memory **must** use an "n" name.

Table 1. Physical ID Naming Conventions

Descriptions within the table below assume the reader is facing the front of the system cabinets.

Component	Format	Description
SMW	s0, all	All components attached to the SMW. xtcli power up s0 powers up all components attached to the SMW.
cabinet	cX-Y	Position: row (X) and row (Y) of cabinet; also used as cabinet controller host name. For example: c12-3 is cabinet 12 in row 3.
chassis	cX-Yc#	Physical unit within cabinet: cX-Y; c# is the chassis number and # is 0-2. Chassis are numbered bottom to top. For example: c0-0c2 is chassis 2 of cabinet c0-0.
blade or slot or module	cX-Yc#s#	Physical unit within a slot of a chassis cX-Yc#, s# is the slot number of the blade and # is 0-7; also used as blade controller host name. Blades are numbered left to right. For example: c0-0c2s4 is slot 4 of chassis 2 of cabinet c0-0. For example: c0-0c2s* is all slots (0...7) of chassis 2 of cabinet c0-0.
node	cX-Yc#s#n#	Node on a blade; n# is the location of the node and # is 0-3.

Component	Format	Description
		<p>For example: <code>c0-0c2s4n0</code> is node 0 on blade 4 of chassis 2 in cabinet <code>c0-0</code></p> <p>For example: <code>c0-0c2s4n*</code> is all nodes on blade 4 of chassis 2 of cabinet <code>c0-0</code></p>
Accelerator (Cray XK systems)	<code>cX-Yc#s#n#a#</code>	<p>Accelerator module (GPU); <code>a#</code> is the location of the GPU and <code>#</code> is 0-15.</p> <p>For example: <code>c0-0c2s4n0a2</code> is GPU 2 on node 0 on blade 4 of chassis 2 of cabinet <code>c0-0</code>.</p> <p>Currently, there is one accelerator per node.</p>
Gemini ASIC	<code>cX-Yc#s#g#</code>	<p>Gemini ASIC within a module; <code>g#</code> is the location of the Gemini ASIC within a module and <code>#</code> is 0 or 1.</p> <p>For example: <code>c0-1c2s3g0</code></p>
LCB within a Gemini ASIC	<code>cX-Yc#s#g#lRC</code>	<p>LCB within a Gemini ASIC; these are numbered according to their tile location. There are 8 rows and 8 columns in the tile grid. The row/column numbers are octal. Valid values are: 0-7 for row (<i>R</i>) and 0-7 for column (<i>C</i>).</p> <p>For example: <code>c1-0c2s3g0157</code> (row 5, column 7).</p> <p>NOTE: The link control block (LCB) numbers that are processor links on Gemini ASICs are: 23, 24, 33, 34, 43, 44, 53, and 54. For this reason, a display of the status of LCBs will normally show these LCBs in a different state than the remaining (network link) LCBs.</p>
section	<code>tA-B</code>	<p>Grouping of cabinets; <i>A</i> is the start cabinet number and <i>B</i> is the end cabinet number in the x direction. A section refers to all cabinets in all columns (y-coordinate) in the A through B rows. Section names are</p>

Component	Format	Description
		defined when the xtdiscover command is executed.
		For example: For a site with four rows of 31 cabinets, the section t0-1 refers to c0-0, c0-1, c0-2, c0-3, c1-0, c1-1, c1-2, and c1-3.
logical machine (partition)	p#	A partition is a group of components that make up a logical machine. Logical systems are numbered from 0 to the maximum number of logical systems minus one. A configuration with 32 logical machines would be numbered p0 through p31 (see Logical Machines on page 62). p0, however, is reserved to refer to the entire machine as a partition.
SerDes macro within a Gemini ASIC	cX-Yc#s#g#m#	SerDes macro within a Gemini ASIC. Each macro implements 4 LCBS. Valid values are 0-9.
Node socket	cX-Yc#s#n#s#	Node socket within a physical node. Valid values are 0-7.
Die within a node socket	cX-Yc#s#n#s#d#	Die within a node physical socket. Valid values are 0-3.
core within a die, within a socket, within a node	cX-Yc#s#n#s#d#c#	Core within a die, within a socket, within a node. Valid values are 0-15.
memory controller within a die, within a socket, within a node	cX-Yc#s#n#s#d#m#	Memory controller within a die, within a socket, within a node. Valid values are 0-3. For example: c0-1c2s3n0s0d1m0
DIMM within a node	cX-Yc#s#n#d#	DIMM within a node. Valid values are 0-31. For example: c0-1c2s3n0d3
Gemini NIC	cX-Yc#s#g#n#	NIC (Network Interface Controller) within a Gemini ASIC. Valid values are 0 and 1. For example: c0-1c2s3g0n1
VERTY	cX-Yc#s#v#	VERTY (voltage converter/regulator) on a blade. Valid values are 0-15. For example: c0-1c2s3v0
FPGA	cX-Yc#s#f#	FPGA. 0 is the L0E and 1 is the LOG on Gemini systems.

Component	Format	Description
XDP cabinet	<i>xX-Y</i>	For example: <i>c0-1c2s3f1</i> is the LOG on a Gemini system. Power cooling control cabinet. For example: <i>x0-1</i> .

Node ID (NID) on Cray XE/XK Systems

The node ID (NID) is a decimal numbering of all CLE nodes. NIDs are sequential numberings of the nodes starting in cabinet *c0-0*. Each additional cabinet continues from the highest value of the previous cabinet; therefore, cabinet 0 has NIDs 0-95, and cabinet 1 has NIDs 96-191, and so on.

A cabinet contains three chassis; chassis 0 is the lower chassis in the cabinet. Each chassis contains eight blades and each blade contains four nodes. The lowest numbered NID in the cabinet is in chassis 0 slot 0 (lower left corner); slots (blades) are numbered left to right (slot 0 to slot 7; when facing the front of the cabinet). In cabinet 0 the lower two nodes in chassis 0 slot 0 are numbered NIDs 0 and 1, the numbering continues moving to the right across the lower two node of each slot; so the lower nodes in slot 1 are NIDs 2 and 3 and so on to slot 7 where the lower two nodes are NIDs 14 and 15. The numbering continues with the upper two nodes on each blade, the upper two nodes on slot 7 are 16 and 17 and continues to the left to slot 0; chassis 0 slot 0 then has NIDs numbered 0, 1, 30, and 31. The numbering continues to chassis 1, so slot 0 in chassis 1 has NIDs 32, 33, 62, and 63. Then chassis 2 slot 0 has NIDs 64, 65, 94, and 95.

When identifying components in the system, remember that a single Gemini ASIC is connected to two nodes. If node 61 reported a failure and the HyperTransport (HT) link was the suspected failure, then Gemini 1 on that blade would be one of the suspect parts. Node 61 is in cabinet 0, chassis 1, slot 1 or *c0-0c1s1n3*. Nodes 0 and 1 (*c0-0c1s1n0* and *c0-0c1s1n1*) are connected to Gemini 0 (*c0-0c1s1g0*) and nodes 2 and 3 (*c0-0c1s1n2* and *c0-0c0s1n3*) are connected to Gemini 1 (*c0-0c1s1g1*).

Use the `xtnid2str` command to convert a NID to a physical ID. For information about using the `xtnid2str` command, see the `xtnid2str(8)` man page. To convert a physical ID to a NID number, use the `rtr --system-map` command and filter the output. For example:

```
crayadm@smw:~> rtr --system-map | grep c1-0c0s14n3 | awk '{ print $1 }'
```

Use the `nid2nic` command to print the *nid-to-nic_address* mappings, *nic_address-to-nid* mappings, and a specific *physical_location-to-nic_address* and *nic* mappings. For information about using the `nid2nic` command, see the `xtnid2str(8)` man page.

Class Name

Class names are a CLE construct. The `/etc/opt/cray/sdb/node_classes` file is created as part of the system installation, based on the `node_class*` settings defined in `CLEinstall.conf`. During the boot process, the `service_processor` database table is populated from the `/etc/opt/cray/sdb/node_classes` file, which the administrator can change by adding or removing nodes (see [Change Nodes and Classes](#) on page 149).

IMPORTANT: It is important to keep node class settings in `CLEinstall.conf` and `/etc/opt/cray/sdb/node_classes` consistent in order to avoid errors during update or upgrade installations (see the *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*).

Class names must be valid, e.g., the name of a directory. Any number of classes may be specified, however, the same class names must be used when invoking the `xtspec` specialization command.

Change the class of a node when its function changes, for example, when adding an additional login node. The `/etc/opt/cray/sdb/node_classes` file describes the nodes associated with each class.

Sample `/etc/opt/cray/sdb/node_classes` file

```
# node:classes
0:service
1:service
8:login
9:service
```

Topology Class

Each Cray system is given a topology class based on the number of cabinets and their cabling. Some commands, such as `xtbounce`, enable the administrator to specify topology class as an option.

The follow commands display the topology class of a system in their output:

- `xtcli status`
- `rca-helper -o`
- `xtclass` (executed on the SMW)

For example:

```
smw:~> xtclass
1
```

Manage the System

Caution is encouraged when executing system management commands and procedures; hasty actions can result in down time and lost data.

IMPORTANT: Use persistent SCSI device names

This does not apply to SMW disks: SCSI device names (`/dev/sd*`) are not guaranteed to be numbered the same from boot to boot. This inconsistency can cause serious system problems following a reboot. When installing CLE, the administrator **must** use persistent device names for file systems on the Cray system.

Cray recommends using the `/dev/disk/by-id/` persistent device names. Use `/dev/disk/by-id/` for the root file system in the `initramfs` image and in the `/etc/sysset.conf` installation configuration file as well as for other file systems, including Lustre (as specified in `/etc/fstab` and `/etc/sysset.conf`). For more information, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*.

Alternatively, the administrator can define persistent names using a site-specific `udev` rule or `cray-scsidev-emulation`. However, only the `/dev/disk/by-id` method has been verified and tested.



CAUTION: The administrator must use `/dev/disk/by-id` when specifying the root file system. There is no support in the `initramfs` for `cray-scsidev-emulation` or custom `udev` rules.

Connect the SMW to the Console of a Service Node

The `xtcon` command is a console interface for service nodes. When it is executing, the `xtcon` command provides a two-way connection to the console of any running node.

See the `xtcon(8)` man page for additional information.

Log on to the Boot Node

The standard Cray configuration has a gigabit Ethernet connection between the SMW and boot node. All other nodes on the Cray system are accessible from the boot node.

1. Log on to the SMW as `crayadm`.
2. There are two methods to log on to the boot node: `ssh` to the boot node.
 - Use `ssh`:

```
crayadm@smw:~> ssh boot
crayadm@boot:~>
```

- Open an administrator window on the SMW:

```
crayadm@smw:~> xterm -ls -vb -sb -sl 2049 6&
```

After the window opens, use it to ssh to the boot node.

Generic RPM Usage

A variety of software packages are distributed as standard Linux RPM Package Manager (RPM) packages. RPM packages are self-contained installation files that must be executed with the `rpm` command in order to create all required directories and install all component files in the correct locations.

To install RPMs on a Cray system, use `xtopview` on the boot node to access and modify the shared root. The `rpm` command is not able to modify the RPM database from a login node or other service node; the root directory is read-only from these nodes.

Any changes to the shared root apply to all service nodes. If the RPM being installed modifies files in `/etc`, it is necessary to invoke `xtopview` to perform any class or node specialization that may be required. `xtopview` specialization applies only to `/etc` in the shared root.

For some Cray distributed RPMs, the `CRAY_INSTALL_DEFAULT` environment variable can be set to configure the new version as the default. Set this variable before installing the RPM. For more information, see the associated installation guide.

For more information on installing RPMs, see the `xtopview(8)` man page and the installation documentation for the specific software package.

Install an RPM on the SMW

As `root`, use the following command:

```
smw:~# rpm -ivh /directorypath/filename.rpm
```

Install an RPM on the Boot Node Root

As `root`, use the following command:

```
boot:~ # rpm -ivh /directorypath/filename.rpm
```

Install an RPM on the Shared Root

As `root`, use the following commands:

NOTE: If the SDB node has not been started, include the `-x /etc/opt/cray/sdb/node_classes` option when invoking the `xtopview` command.

```
boot:~ # cp -a /tmp/filename.rpm /rr/current/software
boot:~ # xtopview
default:// # rpm -ivh /software/filename.rpm
```

Service Node and Compute Node Boot Image Preparation

A *boot image* is an archive containing all the components necessary to boot Cray service nodes and compute nodes. In general, a boot image contains the operating system kernel, ramdisk, and boot parameters used to bring up a node. A single boot image can contain multiple sets of these files to support booting service nodes and compute nodes from the same boot image as well as booting different versions of compute node operating systems. The operating systems supported by a particular boot image are described through load files. A *load file* is simply a manifest of operating system components to include (represented as files) and load address information to provide to the boot loader. Load files should not be edited by the administrator.

Cray system compute and service nodes use a RAM disk for booting. Service nodes and compute nodes use the same `initramfs` format and work space environment. This space is created in `/opt/xt-images/machine-xtrelease-LABEL[-partition]/nodetype`, where *machine* is the Cray hostname, *xtrelease* is the CLE release level, *LABEL* is the system set label in `/etc/sysset.conf`, *partition* describes a system partition and is typically omitted if partitions are not used, and *nodetype* is either `compute` or `service`.

Prepare Compute and Service Node Boot Images

Invoke the `shell_bootimage_LABEL.sh` script to prepare boot images for the system set with the specified *LABEL*. When `shell_bootimage_LABEL.sh` is run, it creates a log file in `/var/adm/cray/logs/shell_bootimage_LABEL.sh.PID.log`. This script uses `xtclone` and `xtpackage` to prepare the work space in `/opt/xt-images`.

`shell_bootimage_LABEL.sh` accepts the following options:

- v** Run in verbose mode.
- T** Run in verbose mode.
- h** Display help message.
- c** Create and set the boot image for the next boot. The default is to display `xtbootimg` and `xtcli` commands that will generate the boot image. Use the `-c` option to invoke these commands automatically.
- b *bootimage*** Specify *bootimage* as the boot image disk device or file name. The default *bootimage* is determined by using values for the system set *LABEL* when `CLEinstall` was run. Use this option to override the default and manage multiple boot images.
- C *coldstart_dir*** Specify *coldstart_dir* as the path to the HSS coldstart applets directory. The default is `/opt/hss-coldstart+gemini/default/xt` for Cray XE systems. Use this option to override the default. This option is not applicable to Cray XC30 systems. For more information, see the `xtbounce(8)` man page.

Optionally, this script includes `CNL_*` parameters that can be used to modify the CNL boot image configuration defined in `CLEinstall.conf`. Edit the script and set the associated parameter to `y` to load an optional RPM or change the `/tmp` configuration.

1. Run `shell_bootimage_LABEL.sh`, where *LABEL* is the system set label specified in `/etc/sysset.conf` for this boot image. For example, if the system set label is *BLUE*, log on to the SMW as `root` and type:

```
smw:~# /var/opt/cray/install/shell_bootimage_BLUE.sh
```

On completion, the script displays the `xtbootimg` and `xtcli` commands that are required to build and set the boot image for the next boot. If the `-c` option was specified, the script invokes these commands automatically and the remaining steps in this procedure should be skipped.

2. Create a unified boot image for compute and service nodes by using the `xtbootimg` command suggested by the `shell_bootimage_LABEL.sh` script.

In the following example, replace *bootimage* with the *mountpoint* for `BOOT_IMAGE0` in the system set that is defined in `/etc/sysset.conf`. Set *bootimage* to either a raw device; for example `/raw0` or a file name; for example `/bootimagedir/bootimage.new`.



CAUTION: If *bootimage* is a file, verify that the file exists in the same path on both the SMW and the boot root.

For Cray XC30 systems, type this command (for partitioned systems, replace `s0` with `pN` where *N* is the partition number for which the image is being built):

```
smw:~# xtbootimg \
-L /opt/xt-images/hostname-xtrelease-LABEL-s0/compute/CNL0.load \
-L /opt/xt-images/hostname-xtrelease-LABEL-s0/service/SNL0.load \
-c bootimage
```

- a. At the prompt 'Do you want to overwrite', type **y** to overwrite the existing boot image file.
- b. If *bootimage* is a file, copy the boot image file from the SMW to the same directory on the boot root. If *bootimage* is a raw device, skip this step. For example, if the *bootimage* file is `/bootimagedir/bootimage.new` and `bootroot_dir` is set to `/bootroot0`, type the following command:

```
smw:~# cp -p /bootimagedir/bootimage.new /bootroot0/bootimagedir/bootimage.new
```

3. Set the boot image for the next system boot by using the suggested `xtcli` command.

The `shell_bootimage_LABEL.sh` program suggests an `xtcli` command to set the boot image based on the value of `BOOT_IMAGE0` for the system set being used. The `-i bootimage` option specifies the path to the boot image and is either a raw device; for example, `/raw0` or `/raw1`, or a file such as `/bootimagedir/bootimage.new`.



CAUTION: The next boot, anywhere on the system, uses the boot image set here.

- a. Display the currently active boot image. Record the output of this command.
If the partition variable in `CLEinstall.conf` is `s0`, type:

```
smw:~# xtcli boot_cfg show
```

Or

If the partition variable in `CLEinstall.conf` is a partition value such as `p0`, `p1`, and so on, type:

```
smw:~# xtcli part_cfg show pN
```

- b. Invoke `xtcli` with the `update` option to set the default boot configuration used by the boot manager.

If the partition variable in `CLEinstall.conf` is `s0`, type the following command to select the boot image to be used for the entire system.

```
smw:~# xtcli boot_cfg update -i bootimage
```

Or

If the partition variable in `CLEinstall.conf` is a partition value such as `p0`, `p1`, and so on, type the following command to select the boot image to be used for the designated partition.

```
smw:~# xtcli part_cfg update pN -i bootimage
```

Customize Existing Boot Images

Prerequisites

Root privileges are required to invoke the `xtclone` and `xtpackage` commands.

Cray recommends using [Prepare Compute and Service Node Boot Images](#) on page 37 to prepare production boot images. Additionally, the system administrator can use the `xtclone`, `xtpackage` and `xtbootimg` utilities on the SMW to modify existing compute node or service node images for the purpose of experimenting with custom options. Root privileges are required to invoke these commands.

Customize a boot image on the SMW using this four-step process:

1. Execute the `xtclone` utility to create a new work area copied from an existing work area.
2. Make the necessary changes in the new work area, e.g., install RPMs, edit configuration files, add or remove scripts.
3. Execute the utility to properly package the operating system components and prepare a load file for use by `xtbootimg`.
To create load files to support, for example, different boot parameters or different RAM disk contents, use the `xtpackage` command with the `-L` option.

The following is a sample service node load file (`SNL0.load`):

```
#NODES REALLY READY
SNL0/size-initramfs 0x9021C
#Kernel source: /opt/xt-images/hostname-4.1.20-LABEL-s0/service/boot/
vmlinuz-2.6.32.59-0.3.1_1.0401.6670-cray_gem_s
SNL0/vmlinuz-2.6.32.59-0.3.1_1.0401.6670-cray_gem_s.bin 0x100000
#Parameters source: /opt/xt-images/hostname-4.1.20-LABEL-s0/service/boot/
parameters-snl
SNL0/parameters 0x90800
SNL0/initramfs.gz 0xFA00000
```

4. Execute the `xtbootimg` utility to create a boot image (an archive or `cpio` file) from the work area. The `xtbootimg` utility collects the components described by one or more load files into a single archive. The load files themselves are also included in the archive, along with other components, BIOS, and sources listed in the load file from `xtpackage`. Use the `xtbootimg -L` option to specify the path to the CNL compute node load file and the path to the service node load file.

Create a Boot Image from Existing File System Images

1. Make copies of the compute-node-side and service-node-side of an existing work area. Cray recommends that the work area be in a subdirectory of `/opt/xt-images`.

```
smw:~ # xtcclone /opt/xt-images/machine-xtrelease-LABEL/compute \
/opt/xt-images/test/compute
smw:~ # xtcclone -s /opt/xt-images/machine-xtrelease-LABEL/service \
/opt/xt-images/test/service
```

2. Make any changes to the work area that are necessary for the site; for example, install or erase RPMs, change configuration files, or add or remove scripts. Use the `xtpackage -s` option to create a "service-node-only" boot image. When finished making changes, wrap up (package) the compute-node-side and service-node-side of the work area.

```
smw:~ # xtpackage /opt/xt-images/test/compute
smw:~ # xtpackage -s /opt/xt-images/test/service
```

The `xtpackage` utility automatically creates an `/etc/xt.sn1` file in service node `initramfs`. This allows compute node hardware to boot service node images, if necessary.

3. Make a boot image (a `cpio` file) from the work area. The directory path for `bootimage` must exist on both the SMW and the boot node, and the `bootimage` files in each location must be identical.

Some configurations export `/opt/xt-images/cpio` via NFS, so the SMW and the boot node can see the same files in `/opt/xt-images/cpio`, although this is not recommended for larger systems. Other configurations use a non-networked file system at `/tmp/boot`, in which case, a copy of `smw:/tmp/boot/bootimage.cpio` must be put at `boot:/tmp/boot/bootimage`. This is required for the boot node to be able to distribute `bootimage` to the other service nodes.

```
smw:~ # xtbootimg -L /opt/xt-images/test/service/SNL0.load \
-L /opt/xt-images/test/compute/CNL0.load \
-c /opt/xt-images/cpio/test/bootimage
```

For more information about these utilities, see the `xtclone(8)`, `xtpackage(8)`, and `xtbootimg(8)` man pages.

Change Boot Parameters



CAUTION: Some of the default boot parameters are mandatory. The system may not boot if they are removed.

Updating the parameters passed to the Linux kernel requires recreating the boot image with the `xtpackage` and `xtbootimg` commands. This is accomplished by either editing the files in the file system image or specifying a path to a file containing parameters. If editing the files, the default service and compute node parameters can be found in `boot/parameters-sn1` and `boot/parameters-cn1`, respectively.

Make a boot image with new parameters for service and CNL compute nodes

```
smw:~ # xtpackage -s -p /tmp/parameters-service.new \
/opt/xt-images/test/service
smw:~ # xtpackage -p /tmp/parameters-compute.new /opt/xt-images/test/
compute
```

```
smw:~ # xtbootimg -L /opt/xt-images/test/service/SNL0.load \
-L /opt/xt-images/test/compute/CNL0.load -c /raw0
```

The xtbootsys Command

The `xtbootsys` command is used to manually boot the boot node, service nodes, and CNL compute nodes. An administrator can also boot the system using both user-defined and built-in procedures in automation files (e.g., `auto.generic.cnl`). Before modifying the `auto.generic.cnl` file, Cray recommends making a copy because it will be replaced by an SMW software upgrade. For related procedures, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*.

The `xtbootsys` command prevents unintentional booting of currently booted partitions. If a boot automation file is being used, `xtbootsys` checks that file to determine if the string `shutdown` exists within any actions defined in the file. If it does, `xtbootsys` assumes that a shutdown is being done, and no further verification of operating on a booted partition occurs. If the partition is not being shut down and the boot node is in the `ready` state, `xtbootsys` announces this fact and queries for confirmation to proceed. By default, confirmation is enabled. To disable or enable confirmation when booting booted partitions, use the `xtbootsys config,confirm_booting_booted` and the `config,confirm_booting_booted_last_session` global TCL variables, the `--config name=value` on the `xtbootsys` command line, or the `XTBOOTSYS_CONFIRM_BOOTING_BOOTED` and `XTBOOTSYS_CONFIRM_BOOTING_BOOTED_LAST_SESSION` environment variables.

Manually Boot the Boot Node and Service Nodes

Prerequisites

The Lustre file system should start up before the compute nodes, and compute node Lustre clients should be unmounted before shutting down the Lustre file system.

If more than one boot image is set up to run, the administrator can check which image is set up to boot with the `xtcli boot_cfg show` or `xtcli part_cfg show pN` commands. To change which image is booting, see [Update the Boot Configuration](#) on page 142.

1. Log on to the SMW as `crayadm`.
2. Invoke the `xtbootsys` command to boot the boot node. If the system is partitioned, invoke `xtbootsys` with the `--partition pN` option.

```
crayadm@smw:~> xtbootsys
```

The `xtbootsys` command prompts with a series of questions. Cray recommends answering **yes** by typing **y** to each question.

```
Enter your boot choice:
 0) boot bootnode ...
 1) boot sdb ...
 2) boot compute ...
 3) boot service ...
 4) boot all (not supported) ...
 5) boot all_comp ...
```

```

10) boot bootnode and wait ...
11) boot sdb and wait ...
12) boot compute and wait ...
13) boot service and wait ...
14) boot all and wait (not supported) ...
15) boot all_comp and wait ...
17) boot using a loadfile ...
18) turn console flood control off ...
19) turn console flood control on ...
20) spawn off the network link recovery daemon (xtnlrd)...
q) quit.

```

3. Select option 10 (boot bootnode and wait).

A prompt to confirm the selection is displayed. Press the **Enter** key or type **Y** to each question to confirm.

```

Do you want to boot the boot node ? [Yn] Y
Do you want to send the ec_boot event ('no' means to only load memory) ? [Yn] Y

```

After the boot node is booted, the process returns to the boot choice menu.

4. Select option 11 (boot sdb and wait).

A prompt to confirm the selection is displayed. Press the **Enter** key or type **Y** to each question to confirm.

```

Do you want to boot the sdb node ? [Yn] Y
Do you want to send the ec_boot event ('no' means to only load memory) ? [Yn] Y

```

5. Select option 13 (boot service and wait).

A prompt to enter a list of service nodes to be booted is displayed.

6. Type **p0** to boot the remaining service nodes in the entire system or **pN** (where *N* is the partition number) to boot a partition.

```

Do you want to boot service p0 ? [Yn] Y
Do you want to send the ec_boot event ('no' means to only load memory) ? [Yn] Y

```

To confirm the selection, press the **Enter** key or type **Y** to each question.

7. Log on to any service nodes for which there are local configuration or startup scripts (such as starting Lustre) and run the scripts.

Manually Boot the Compute Nodes

Prerequisites

All service and login nodes are booted and Lustre, if configured at this time, has started.

1. Invoke the `xtbootstsys` command if it is not running.

```
crayadm@smw:~> xtbootstsys
```

```

Enter your boot choice:
 0) boot bootnode ...
 1) boot sdb ...
 2) boot compute ...
 3) boot service ...

```

```

4) boot all (not supported) ...
5) boot all_comp ...
10) boot bootnode and wait ...
11) boot sdb and wait ...
12) boot compute and wait ...
13) boot service and wait ...
14) boot all and wait (not supported) ...
15) boot all_comp and wait ...
17) boot using a loadfile ...
18) turn console flood control off ...
19) turn console flood control on ...
20) spawn off the network link recovery daemon (xtnlrd)...
q) quit.

```

2. Select option 17 (boot using a loadfile). A series of prompts are displayed. Type the responses indicated in the following example. For the `component list` prompt, type `p0` to boot the entire system, or `pN` (where `N` is the partition number) to boot a partition. At the final three prompts, press the **Enter** key.

```

Enter your boot choice: 17
Enter a boot type string (or nothing to do nothing): CNL0
Enter a boot type option (or nothing to do nothing): compute
Enter a component list (or nothing to do nothing): p0
Enter 'any' to wait for any console output,
  or 'linux' to wait for a linux style boot,
  or anything else (or nothing) to not wait at all: Enter
Enter an alternative CPIO archive name (or nothing): Enter
Do you want to send the ec_boot event ('no' means to only load memory) ? [Yn] Enter

```

3. Return to the `xtbootstys` menu after all compute nodes are booted. Type `q` to exit the `xtbootstys` program.
4. Remove the `/etc/nologin` file from all service nodes to permit a non-root account to log on.

```

smw:~# ssh root@boot
boot:~# xtunspec -r /rr/current -d /etc/nologin

```

Reboot a Single Compute Node

A system administrator can initiate a warm boot with the `xtbootstys` command's `--reboot` option. This operation performs minimal initialization followed by a boot of only the selected compute nodes. Unlike the sequence that is used by the `xtbounce` command, there is no power cycling of the Cray ASICs or of the node itself; therefore, the high-speed network (HSN) routing information is preserved. Do not specify a session identifier (`-s` or `--session` option) because `--reboot` continues the last session and adds the selected components to it.

Reboot a single comput node

For this example, reboot node `c1-0c2s1n2`:

```

crayadm@smw:~> xtbootstys --reboot c1-0c2s1n2

```

Reboot Login or Network Nodes

Login or network nodes cannot be rebooted through a shutdown or reboot command issued on the node; they must be restarted through the HSS system using the `xtbootsys --reboot idlist SMW` command. The HSS must be used so that the proper kernel is pushed to the node.

IMPORTANT: Do not attempt to warm boot nodes running other services in this manner.

For additional information, see the `xtbootsys(8)` man page.

Reboot login or network nodes

```
crayadm@smw:~> xtbootsys --reboot idlist
```

Boot a Node or Set of Nodes Using the xtcli boot Command

To boot a specific image or load file on a given node or set of nodes, execute the HSS `xtcli boot boot_type` command, as shown in the following examples. When using a file for the boot image, the same file must be on both the SMW and the `bootroot` at the same path.



WARNING: Each system boot must be started with an `xtbootsys` session to establish a `sessionid`. Perform direct boot commands using the `xtcli boot` command only **after** a session has been established through `xtbootsys`.

Boot all service nodes with a specific image

For this example, the specific image is located at `/raw0`:

```
crayadm@smw:~> xtcli boot all_serv_img -i /raw0
```

Boot all compute nodes with a specific image

For this example, the specific image is located at `/bootimagedir/bootimage`:

```
crayadm@smw:~> xtcli boot all_comp_img -i /bootimagedir/bootimage
```

Boot compute nodes using a load file

The following example boots all compute nodes in the system with using a load file name `CNL0`:

```
crayadm@smw:~> xtcli boot CNL0 -o compute s0
```

Increase the Boot Manager Timeout Value

On systems of 4,000 nodes or larger, the time that elapses until the boot manager receives all responses to the boot requests can be greater than the default 60-second time-out value. This is due, in large part, to the amount of other event traffic that occurs as each compute node generates its console output.

To avoid this problem, change the `boot_timeout` value in the `/opt/cray/hss/default/etc/bm.ini` file on the SMW to increase the the default time-out value; for example:

Increase the `boot_timeout` value

For systems of 4,000 to 7,000 nodes, change the `boot_timeout` line to:

```
boot_timeout 120
```

For systems larger than 7,000 nodes, change the `boot_timeout` line to:

```
boot_timeout 180
```

Request and Display System Routing

Use the HSS `rtr` command to request routing for the HSN, to verify current route configuration, or to display route information between nodes. Upon startup, `rtr` determines whether it is making a routing request or an information request.

For more information, see the `rtr(8)` man page.

Display routing information

The `--system-map` option to `rtr` writes the current routing information to `stdout` or to a specified file. This command can also be helpful for translating node IDs (NIDs) to physical ID names.

```
crayadm@smw:~> rtr --system-map
```

Route the entire system

The `rtr -R | --route-system` command sends a request to perform system routing. If no components are specified, the entire configuration is routed as a single routing domain based on the configuration information provided by the state manager. If a component list (*idlist*) is provided, routing is limited to the listed components. The state manager configuration further limits the routing domain to omit disabled blades, nodes, and links and empty blade slots.

```
crayadm@smw:~> rtr --route-system
```

Bounce Blades Repeatedly Until All Blades Succeed

IMPORTANT: This iterative `xtbounce` should typically be done in concert with an `xtbootsys` automation file where bounce and routing are turned off.

1. Bounce the system.

```
smw:~> xtbounce s0
```

2. Bounce any blades that failed the first bounce. Repeat as necessary.

3. Execute the following command, which copies route configuration files, based on the `idlist` (such as `s0`), to the blade controllers. This avoids having old, partial route configuration files left on the blades that were bounced earlier and ensures that the links are initialized correctly.

```
smw:~> xtbounce --linkinit s0
```

4. Route and boot the system without executing `xtbounce` again. If using a `xtbootsys` automation file, specify `set data(config,xtbounce) 0`, or use the `xtbootsys --config xtbounce=0` command.

Shut Down the System Using the `auto.xtshutdown` File

The preferred method to shut down the system is to use the `xtbootsys` command with the auto shutdown file as follows:

```
crayadm@smw:~> xtbootsys -s last -a auto.xtshutdown
```

Or, for a partitioned system with partition `pN`:

```
smw:~# xtbootsys --partition pN -s last -a auto.xtshutdown
```

This method shuts down the compute nodes (which are commonly also Lustre clients), then executes `xtshutdown` on service nodes, halting the nodes and then stopping processes on the SMW. A system administrator can shut down the system using both user-defined and built-in procedures in the `auto.xtshutdown` file, which is located on the SMW in the `/opt/cray/hss/default/etc` directory.

For related procedures, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*. For more information about using automation files, see the `xtbootsys(8)` man page.

The `xtshutdown` Command

The `xtshutdown` command executes a series of commands locally on the boot node and service nodes to shut down the system in an orderly fashion. The sequence of shutdown steps and the nodes on which to execute them are defined by the system administrator in the `/etc/opt/cray/init-service/xtshutdown.conf` file or in the file specified by the environment variable `XTSHUTDOWN_CONF`.

Root user privileges are required to run `xtshutdown`. Passwordless `ssh` must be enabled for the root user from the boot node to all service nodes.

The `xtshutdown` command uses `pdsh` to invoke commands on the selected service nodes (i.e., boot node, SDB node, a class of nodes, or a single host). A system administrator can define functions to execute when the system is shut down. Place these functions in the `/etc/opt/cray/init-service/xt_shutdown_local` file or the file defined by the `XTSHUTDOWN_LOCAL` environment variable.

Shut Down Service Nodes

Prerequisites

`root` user privileges are required to run `xtshutdown`. Passwordless `ssh` must be enabled for the `root` user from the boot node to all service nodes.



CAUTION: The `xtshutdown` command does not shut down compute nodes. To shut down the compute and service nodes, see [Shut Down the System or Part of the System Using the `xtcli shutdown` Command](#).

For information about shutting down service nodes, see the `xtshutdown(8)` man page.

1. Modify the `/etc/opt/cray/init-service/xtshutdown.conf` file or the file specified by the `XTSHUTDOWN_CONF` environment variable to define the sequence of shutdown steps and the nodes on which to execute them. The `/etc/opt/cray/init-service/xtshutdown.conf` file resides on the boot node.
2. If desired, define functions to execute when the system is shut down. Place these functions in the `/etc/opt/cray/init-service/xt_shutdown_local` file or the file defined by the `XTSHUTDOWN_LOCAL` environment variable.
3. Execute `xtshutdown`.

```
boot:~ # xtshutdown
```

After the software on the nodes is shutdown, the administrator can halt the hardware, reboot, or power down.

Shut Down the System or Part of the System Using the `xtcli shutdown` Command

The HSS `xtcli shutdown` command shuts down the system or a part of the system. To shut down compute nodes, execute the `xtcli shutdown` command. Under normal circumstances, for example to successfully disconnect from Lustre, invoking the `xtcli shutdown` command attempts to gracefully shut down the specified nodes.

For information, see the `xtcli(8)` man page.

Shut down all compute nodes

```
crayadm@smw:~> xtcli shutdown compute
```

Shut down specified compute nodes

For this example, shut down only compute nodes in cabinet c13-2:

```
crayadm@smw:~> xtcli shutdown c13-2
```

Shut down all nodes of a system

```
crayadm@smw:~> xtcli shutdown s0
```

Shut down a partition *pN* of a system

```
crayadm@smw:~> xtcli shutdown pN
```

Force nodes to shut down (immediate halt)

When all nodes of a system must be halted immediately, use the `-f` argument; nodes will not go through their normal shutdown process. Forced shutdown occurs even if the nodes have an alert status present.

```
crayadm@smw:~> xtcli shutdown -f s0
```

After the software on the nodes is shutdown, the system administrator can halt the hardware, reboot, or power down.

Stop System Components

When a system administrator removes, stops, or powers down components, any applications and compute processes that are running on those components are lost.

Reserve a Component

To allow applications and compute processes to complete before stopping components, use the HSS `xtcli set_reserve idlist` command to prevent the selected nodes from accepting new jobs.

A node running CNL and using ALPS is considered to be down by ALPS after it is reserved using the `xtcli set_reserve` command. The output from `apstat` will show the node as down (DN), even though there may be an application running on that node. This DN designation indicates that no other work will be placed on the node after the currently running application has terminated.

For more information, see the `xtcli_set(8)` man page.

Reserve a component

```
crayadm@smw:~> xtcli set_reserve idlist
```

Power Down Blades or Cabinets



WARNING: Power down the cabinets with software commands. Tripping the circuit breakers may result in damage to system components.



WARNING: Before powering down a blade or a cabinet, ensure the operating system is not running.

The `xtcli power down` command powers down the specified cabinet and/or blades within the specified partition, chassis or list of blades. Cabinets must be in the `READY` state to receive power commands. See [System Component States](#) on page 340. The `xtcli power down` command has the following form, where *physIDlist* is a comma-separated list of cabinets or blades present on the system.

```
xtcli power down physIDlist
```

The `xtcli power force_down` and `xtcli power down_slot` commands are aliases for the `xtcli power down` command. For information about disabling and enabling components, see [Disable Hardware Components](#), and [Enable Hardware Components](#), respectively.



WARNING: Although a blade is powered off, the HSS in the cabinet is live and has power.

For information about powering down a component, see the `xtcli_power(8)` man page.

Power down a specified blade

For this example, power down a blade with the ID `c0-0c0s7`:

```
crayadm@smw:~> xtcli power down c0-0c0s7
```

Halt Selected Nodes

Use the HSS `xtcli halt` command to halt selected nodes. For more information, see the `xtcli(8)` man page.

Halt a node

For this example, halt node `157`:

```
crayadm@smw:~> xtcli halt 157
```

Restart a Blade or Cabinet

IMPORTANT: Change the state of the hardware only when the operating system is not running or is shut down.

The `xtcli power up` command powers up the specified cabinet and/or blades within the specified partition, chassis or list of blades. Cabinets must be in the `READY` state (see [System Component States](#) on page 340) to

receive power commands. The `xtcli power up` command does not attempt to power up network mezzanine cards or nodes that are handled by the `xtbounce` command during system boot.

The `xtcli power up_slot` command is an alias for the `xtcli power up` command.

The `xtcli power up` command has the following form, where *physIDlist* is a comma-separated list of cabinets or blades present on the system.

```
xtcli power up physIDlist
```

For more information, see the `xtcli_power(8)` man page.

```
Power up blades in c0-0c0s7
```

```
crayadm@smw:~> xtcli power up c0-0c0s7
```

Abort Active Sessions on the HSS Boot Manager

Use the HSS `xtcli session abort` command to abort sessions in the boot manager. A session corresponds to executing a specific command such as `xtcli power up` or `xtcli boot`.

For more information about manager sessions, see the `xtcli(8)` man page.

1. Display all running sessions in the boot manager. Only the boot manager supports multiple simultaneous sessions.

```
crayadm@smw:~> session show BM all
```

2. Abort the selected session, *session_id*.

```
crayadm@smw:~> xtcli session abort BM session_id
```

Display and Change Software System Status

The user command `xtnodestat` provides a display of the status of nodes: how they are allocated and to what jobs. The `xtnodestat` command provides current job and node status summary information, and it provides an interface to ALPS and jobs running on CNL compute nodes. ALPS must be running in order for `xtnodestat` to report job information.

For more information, see the `xtnodestat(1)` man page.

View and Change the Status of Nodes

Use the `xtprocadmin` command on a service node to view the status of components of a booted system in the `processor` table of the SDB. The command enables the system administrator to retrieve or set the processing mode (`interactive` or `batch`) of specified nodes. The administrator can display the state (`up`, `down`,

admindown, route, or unavailable) of the selected components, if needed. The administrator can also allocate processor slots or set nodes to become unavailable at a particular time. The node is scheduled only if the status is up.

When the `xtprocadmin -ks` option is used, then the option can either a normal argument (up, down, etc.), or it can have a colon in it to represent a conditional option; for example, the option of the form `up:down` means "if state was up, mark down".

For more information, see the `xtprocadmin(8)` man page.

View node characteristics

```
login:~> xtprocadmin
```

NID	(HEX)	NODENAME	TYPE	STATUS	MODE
0	0x0	c0-0c0s0n0	service	up	other
1	0x1	c0-0c0s0n1	service	up	batch
2	0x2	c0-0c0s1n0	compute	up	interactive
3	0x3	c0-0c0s1n1	compute	up	interactive
4	0x4	c0-0c0s2n0	compute	up	interactive
5	0x5	c0-0c0s2n1	compute	up	interactive
6	0x6	c0-0c0s3n0	compute	up	batch
7	0x7	c0-0c0s3n1	compute	up	batch
8	0x8	c0-0c0s4n0	service	up	batch
9	0x9	c0-0c0s4n1	service	up	batch
10	0xa	c0-0c0s5n0	compute	up	batch

View all node attributes

```
login:~> xtprocadmin -A
```

NID	(HEX)	NODENAME	TYPE	ARCH	OS	CPUS	CU	AVAILMEM	PAGESZ	CLOCKMHZ	GPU	SOCKETS	DIES	C/CU
0	0x0	c0-0c0s0n0	service	xt	(service)	6	6	16384	4096	2400	0	1	1	1
1	0x1	c0-0c0s0n1	service	xt	(service)	6	6	16384	4096	2400	0	1	1	1
2	0x2	c0-0c0s1n0	compute	xt	CNL	24	24	32768	4096	1900	0	2	4	1
3	0x3	c0-0c0s1n1	compute	xt	CNL	24	24	32768	4096	1900	0	2	4	1
4	0x4	c0-0c0s2n0	compute	xt	CNL	24	24	32768	4096	1900	0	2	4	1
5	0x5	c0-0c0s2n1	compute	xt	CNL	24	24	32768	4096	1900	0	2	4	1
6	0x6	c0-0c0s3n0	compute	xt	CNL	32	16	131072	4096	2100	0	2	4	2
7	0x7	c0-0c0s3n1	compute	xt	CNL	32	16	131072	4096	2100	0	2	4	2
8	0x8	c0-0c0s4n0	service	xt	(service)	6	6	16384	4096	2400	0	1	1	1
9	0x9	c0-0c0s4n1	service	xt	(service)	6	6	16384	4096	2400	0	1	1	1
10	0xa	c0-0c0s5n0	compute	xt	CNL	32	16	32768	4096	2100	0	2	4	2

View selected attributes of selected nodes

For this example, the `-a` option lists the selected attributes to display:

```
login:~> xtprocadmin -n 8 -a arch,clockmhz,os,cores
```

NID	(HEX)	NODENAME	TYPE	ARCH	CLOCKMHZ	OS	CPUS
8	0x8	c0-0c0s4n0	service	xt	2400	(service)	6

Disable a node

For this example, the `admindown` option disables node `c0-0c0s3n1` such that it cannot be allocated:

```
crayadm@nid00004:~> xtprocadmin -n c0-0c0s3n1 -k s admin
```

Disable all processors

```
crayadm@nid00004:~> xtprocadmin -k s admindown
```

Mark a Compute Node as a Service Node

Use the `xtcli mark_node` command to mark a node in a compute blade to have a role of `service` or `compute`; `compute` is the default. It is not permitted to change the role of a node on a service blade, which always has the `service` role.

Marking a node on a compute blade as `service` or `compute` allows the administrator to load the desired boot image at boot time. Compute nodes marked as `service` can run software-based services. A request to change the role of a running node (that is, the node is in the `ready` state and the operating system is running) will be denied.

For more information, see the `xtcli(8)` man page and [Check the Status of System Components](#) on page 86.

Find Node Information**Translate Between Physical ID Names and Integer NIDs**

To translate between physical ID names (`cnames`) and integer NIDs, generate a system map on the System Management Workstation (SMW) and filter the output, enter the following command:

```
crayadm@smw:~> rtr --system-map | grep cname | awk '{ print $1 }'
```

To translate between physical ID names (`rnames`) and integer NIDs, generate a system map on the System Management Workstation (SMW) and filter the output, enter the following command:

```
crayadm@smw:~> rtr --system-map | grep rname | awk '{ print $1 }'
```

For more information, see the `rtr(8)` man page.

Find Node Information Using the `xtnid2str` Command

The `xtnid2str` command converts numeric node identification values to their physical names (`cnames`). This allows conversion of Node ID values, ASIC NIC address values, or ASIC ID values.

For additional information, see the `xtnid2str(8)` man page.

Find the physical ID for node 38

```
smw:~> xtnid2str 28
node id 0x26 = 'c0-0c0s1n2'
```

Find the physical ID for nodes 0, 1, 2, and 3

```
smw:~> xtnid2str 0 1 2 3
node id 0x0 = 'c0-0c0s0n0'
```

```
node id 0x1 = 'c0-0c0s0n1'
node id 0x2 = 'c0-0c0s1n0'
node id 0x3 = 'c0-0c0s1n1'
```

Find the physical IDs for Gemini IDs 0-7

```
smw:~> xtnid2str -g 0-7
gem id 0x0 = 'c0-0c0s0g0'
gem id 0x1 = 'c0-0c0s1g0'
gem id 0x2 = 'c0-0c0s2g0'
gem id 0x3 = 'c0-0c0s3g0'
gem id 0x4 = 'c0-0c0s4g0'
gem id 0x5 = 'c0-0c0s5g0'
gem id 0x6 = 'c0-0c0s6g0'
gem id 0x7 = 'c0-0c0s7g0'
```

Find Node Information Using the nid2nic Command

The `nid2nic` command prints the *nid*-to-*nic* address mappings, *nic*-to-*nid* address mappings, and a specific *physical_location*-to-*nic* address and *nid* mappings.

For information about using the `nid2nic` command, see the `nid2nic(8)` man page.

Print the *nid*-to-*nic* address mappings for the node with NID 31

```
smw:~> nid2nic 31
NID:0x1f      NIC:0x21      c0-0c0s0n3
```

Print the *nid*-to-*nic* address mappings for the node with NID 31, but specify the NIC value in the command line

```
smw:~> nid2nic -n 0x21
NIC:0x21      NID:0x1f      c0-0c0s0n3
```

Display and Change Hardware System Status

A system administrator can execute commands that look at and change the status of the hardware.



CAUTION: Execute commands that change the status of hardware only when the operating system is shut down.

Generate HSS Physical IDs

The HSS `xtgenid` command generates HSS physical IDs, for example, to create a list of blade controller identifiers for input to the flash manager. Selection can be restricted to components of a particular type. Only user `root` can execute the `xtgenid` command.

For more information, see the `xtgenid(8)` man page.

Create a list of node identifiers that are not in the `DISABLE`, `EMPTY`, or `OFF` state

```
smw:~ # xtgenid -t node --strict
```

Disable Hardware Components

If links, nodes, or Cray ASICs have hardware problems, the system administrator can direct the system to ignore the components with the `xtcli disable` command.

By default, when disabling a component, this command takes into consideration the hierarchy of components, performs the action upon the identified component(s) and cascades that action to any subcomponent of the identified component(s), unless the `-n` option is specified.

The `xtcli disable` command has the following form, where *idlist* is a comma-separated list of components (in `cname` format) that the system is to ignore. The system disregards these links or nodes.

```
xtcli disable [{-t type [-a] } | -n] [-f] idlist
```

IMPORTANT: The `-n` option with the `xtcli disable` command must be used carefully because this may create invalid system state configurations.

NOTE: If a node has a GPU, then the state of the node and the state of its GPU are equal, except when a GPU has been disabled. If the GPU is disabled, it does not take part in any further state transitions, and no flags are set on the GPU until the GPU is re-enabled.

Disabling of a cabinet, chassis, or blade will fail if any nodes under the component are in the `ready` state, unless the force option (`-f`) is used. An error message will indicate the reason for the failure.

Disabling of a node in the `ready` state will fail, unless the force option (`-f`) is used. An error message will indicate the reason for the failure.

The state of `empty` components will not change when using the `disable` command, unless the force option (`-f`) is used.

For detailed information about using the `xtcli disable` command, see the `xtcli(8)` man page.

Disable the Aries ASIC `c0-0c1s3a0`

1. Determine that the ASIC is in the `OFF` state.

```
crayadm@smw:~> xtcli status -t aries c0-0c1s3a0
```

2. If the ASIC is not in the `OFF` state, power down the blade that contains the ASIC.

```
crayadm@smw:~> xtcli power down c0-0c1s3
```

3. Disable the ASIC.

```
crayadm@smw:~> xtcli disable c0-0c1s3a0
```

4. Power up the blade that contains the ASIC.

```
crayadm@smw:~> xtcli power up c0-0c1s3
```

Disable the Gemini ASIC c0-0c1s3g0

1. Determine that the Gemini ASIC is in the `OFF` state.

```
crayadm@smw:~> xtcli status -t gemini c0-0c1s3g0
```

2. If the ASIC is not in the `OFF` state, power down the blade that contains the ASIC.

```
crayadm@smw:~> xtcli power down c0-0c1s3
```

3. Disable the ASIC.

```
crayadm@smw:~> xtcli disable c0-0c1s3g0
```

4. Power up the blade that contains the ASIC.

```
crayadm@smw:~> xtcli power up c0-0c1s3
```

Enable Hardware Components

If links, nodes, or Cray ASICs that have been disabled are later fixed, the system administrator can add them back to the system with the `xtcli enable` command.

The `xtcli enable` command has the following form, where *idlist* is a comma-separated list of components (in `cname` format) for the system to recognize.

```
xtcli enable [{-t type [-a] } | -n] [-f] idlist
```

By default, when disabling a component, this command takes into consideration the hierarchy of components, performs the action upon the identified component(s) and cascades that action to any subcomponent of the identified component(s), unless the `-n` option is specified.

IMPORTANT: The `-n` option with the `xtcli disable` command must be used carefully because this may create invalid system state configurations.

The state of `empty` components does not change when using the `xtcli enable` command, unless the force option (`-f`) is used.

The state of `off` means that a component is present on the system. If the component is a blade controller, node, or ASIC, then this will also mean that the component is powered off. If the administrator disables a component, the state shown becomes `disabled`. When the `xtcli enable` command is used to enable that component for use once again, its state switches from `disabled` to `off`. In the same manner, enabling an empty component means that its state switches from `empty` to `off`.

For more information, see the `xtcli(8)` man page.

For this example, enable Gemini ASIC `c0-0c1s3g0`:

```
crayadm@smw:~> xtshow_disabled s0 | grep c0-0c1s3
c0-0c1s3g0: - OP| disabled [noflags|]
c0-0c1s3g0100: - OP| disabled [noflags|]
c0-0c1s3g0101: - OP| disabled [noflags|]
c0-0c1s3g0102: - OP| disabled [noflags|]
c0-0c1s3g0103: - OP| disabled [noflags|]
c0-0c1s3g0104: - OP| disabled [noflags|]
```

```
.  
crayadm@smw:~> xtcli enable c0-0c1s3g0  
Network topology: class 0  
All components returned success.
```

Set Hardware Components to **EMPTY**

Use the `xtcli set_empty` command to set a selected component to the `EMPTY` state. HSS managers and the `xtcli` command ignore empty or disabled components.

Setting a selected component to the `EMPTY` state is typically done when a component, usually a blade, is physically removed. By setting it to `EMPTY`, the system ignores it and routes around it.

By default, when disabling a component, this command takes into consideration the hierarchy of components, performs the action upon the identified component(s) and cascades that action to any subcomponent of the identified component(s), unless the `-n` option is specified.

IMPORTANT: The `-n` option with the `xtcli disable` command must be used carefully because this may create invalid system state configurations.

For more information, see the `xtcli(8)` man page.

Set a blade to the **EMPTY** state

```
crayadm@smw:~> xtcli set_empty -a c0-0c1s7
```

Lock Hardware Components

Components are automatically locked when a command that can change their state is running. As the command is started, the state manager locks these components so that nothing else can affect their state while the command executes. When the manager is finished with the command, it unlocks the components.

Use the HSS `xtcli lock` command to lock components. Locking a component prints out the state manager session ID.

For more information, see the `xtcli(8)` man page.

Lock cabinet c0-0

```
crayadm@smw:~> xtcli lock -l c0-0
```

Show all session (lock) data

```
crayadm@smw:~> xtcli lock show
```

Unlock Hardware Components

Use the HSS `xtcli lock` command to unlock components. This command is useful when an HSS manager fails to unlock some set of components.

The system administrator can manually check for locks with the `xtcli lock show` command and then unlock them. Unlocking a component does not print out the state manager session ID. The `-u` option must be used to unlock a component as follows:

```
crayadm@smw:~> xtcli lock -u lock_number
```

Where `lock_number` is the value given when initiating the lock; it is also indicated in the `xtcli lock show query`. Unlocking does nothing to the state of the component other than to release locks associated with it.

HSS daemons cannot affect components that are locked by a different session.

Perform Parallel Operations on Service Nodes

Use `pdsh`, the CLE parallel remote shell utility for service nodes, to issue commands to groups of nodes in parallel. The system administrator can select the nodes on which to use the command, exclude nodes from the command, and limit the time the command is allowed to execute. Only user `root` can execute the `pdsh` command. The command has the following form:

```
pdsh [options] command
```

For more information, see the `pdsh(1)` man page.

Restart the NTP service

```
boot:~ # pdsh -w 'login[1-9]' /etc/init.d/ntp restart
```

Perform Parallel Operations on Compute Nodes

The parallel command tool (`pcmd`) facilitates execution of the same commands on groups of compute nodes in parallel, similar to `pdsh`. Although `pcmd` is launched from a service node, it acts on compute nodes. It allows administrators and/or, if the site deems it feasible, other users to securely execute programs in parallel on compute nodes. The user can specify on which nodes to execute the command. Alternatively, the user can specify an application ID (`apid`) to execute the command on all the nodes available under that `apid`.

An unprivileged user must execute the command targeting nodes where the user is currently running an aprun. A `root` user is allowed to target any compute node, regardless of whether there are jobs running there or not. In either case, if the `aprun` exits and the associated applications are killed, any commands launched by `pcmd` will also exit.

By default, `pcmd` is installed as a `root`-only tool. It must be installed as `setuid root` in order for unprivileged users to use it.

The `pcmd` command is located in the `nodehealth` module. If the `nodehealth` module is not part of the default profile, load it by specifying:

module load nodehealth

For additional information, see the `pcmd(1)` man page.

xtbounce Error Message Indicates Cabinet Controller and Its Blade Controllers Not in Sync

During the `gather_cab_pwr_states` phase of `xtbounce`, if the HSS software on a cabinet controller and any of its blade controllers is out of sync, error messages such as the following will be printed during the `xtbounce`.

```
***** gather_cab_pwr_states *****
18:28:42 - Beginning to wait for response(s)

ERROR: rs_phys_node2ascii() failed for node struct 0xb7e70150080700f8
```

If this occurs, it indicates that the blade controller software is at a different revision than the cabinet controller software. `xtbounce` will print a list of cabinets for which this error has occurred. The message will be similar to the following:

```
ERROR: power state check error on 2 cabinet(s)
WARNING: unable to find c0-0 in err_cablist
WARNING: unable to find c0-2 in err_cablist
```

This error is an indication that when the HSS software was previously updated, the cabinet controllers and the blade controllers were not updated to the same version.

To correct this error, cancel out of `xtbounce` (with **Ctrl-C**), wait approximately five minutes for the `xtbounce` related activities on the blade controllers to finish, then reboot the cabinet controller(s) and their associated blade controllers to get the HSS software synchronized. Following this, the `xtbounce` may be executed once again.

Power-cycle a Component to Handle Bus Errors

Bus errors are caused by machine-check exceptions. If a bus error occurs, try power-cycling the component.

1. Power down the components. The `physIDlist` is a comma-separated list of components present on the system.

```
crayadm@smw:~> xtcli power down physIDlist
```

2. Power up the components.

```
crayadm@smw:~> xtcli power up physIDlist
```

When a Component Fails

Components that fail are replaced as field replaceable units (FRUs). FRUs include compute blade components, service blade components, and power and cooling components.

When a field replaceable unit (FRU) problem arises, contact a Customer Service Representative to schedule a repair.

Capture and Analyze System-level and Node-level Dumps

The `xtumpsys` command collects and analyzes information from a Cray system that is failing or has failed, has crashed, or is hung. Analysis is performed on, for example, event log data, active heartbeat probing, voltages, temperatures, health faults, in-memory console buffers, and high-speed interconnection network errors. When failed components are found, detailed information is gathered from them.

To collect similar information for components that have not failed, invoke the `xtumpsys` command with the `--add` option and name the components from which to collect data. The HSS `xtumpsys` command saves dump information in `/var/opt/cray/dump/timestamp` by default.

NOTE: When using the `--add` option to add multiple components, separate components with spaces, not commas.

Dump information about a working component

For this example, dump the entire system and collect detailed information from all blade controllers in chassis 0 of cabinet 0:

```
crayadm@smw:~> xtumpsys --add c0-0c0s0
```

The `xtumpsys` command is written in Python and support plug-ins written in Python. A number of plug-in scripts are included in the software release. Call `xtumpsys --list` to view a list of included plug-ins and their respective directories. The `xtumpsys` command also now supports the use of configuration files to specify `xtumpsys` presets, rather than entering them via the command line.

For more information, see the `xtumpsys(8)` man page.

cdump and crash Utilities for Node Memory Dump and Analysis

NOTE: The `ldump` and `lcrash` utilities are deprecated and replaced by `cdump` and `crash`. Sites using `ldump` and `lcrash` must convert to `cdump` and `crash`.

The `cdump` and `crash` utilities may be used to analyze the memory on any Cray service node or CNL compute node. The `cdump` command is used to dump node memory to a file. After `cdump` completes, the `crash` utility can be used on the dump file generated by `cdump`.

Cray recommends executing the `cdump` utility only if a node has panicked or is hung, or if a dump is requested by Cray.

To select the desired access method for reading node memory, use the `cdump -r access` option. Valid access methods are:

- xt-bhs** The `xt-bhs` method uses a basic hardware system server that runs on the SMW to access and read node memory. `xt-bhs` is the default access method for these systems.
- xt-hsn** The `xt-hsn` method utilizes a proxy that reads node memory through the High-speed Network (HSN). The `xt-hsn` method is faster than the `xt-bhs` method, but there are situations where it will not work (for example, if the ASIC is not functional). However, the `xt-hsn` method is preferable because the dump completes in a short amount of time and the node can be returned to service sooner.
- xt-file** The `xt-file` method is used for memory dump file created by the `-z` option. The compressed memory dump file must be uncompressed prior to executing this command. Use the file name for `node-id`.
- xc-knc** The `xc-knc` method is used to dump Intel Xeon Phi nodes. Use this method when dumping only the Xeon Phi coprocessor without dumping the host node. When dumping the host node, do not use `xc-knc`. A host node dump automatically includes dumping the Xeon Phi coprocessors unless they are suppressed by specifying the `-n` option.

To dump Cray node memory, `access` takes the following form:

```
method[@host]
```

For additional information, see the `cdump(8)` and `crash(8)` man pages.

Dump and Reboot Nodes Automatically

The SMW daemon `dumpd` initiates automatic dump and reboot of nodes when requested by the Node Health Checker (NHC).



CAUTION: The `dumpd` daemon is invoked automatically by `xtbootsys` on system (or partition) boot. In most cases, system administrators do not need to use this daemon directly.

A system administrator can set global variables in the `/etc/opt/cray/nodehealth/nodehealth.conf` configuration file to control the interaction of NHC and `dumpd`. For more information about NHC and the `nodehealth.conf` configuration file, see [Configure the Node Health Checker \(NHC\)](#) on page 120.

Variables can also be set in the `/etc/opt/cray-xt-dumpd/dumpd.conf` configuration file on the SMW to control how `dumpd` behaves on the system.

Each CLE release package also includes an example `dumpd` configuration file, `/etc/opt/cray-xt-dumpd/dumpd.conf.example`. The `dumpd.conf.example` file is a copy of the `/etc/opt/cray-xt-dumpd/dumpd.conf` file provided for an initial installation.

IMPORTANT: The `/etc/opt/cray-xt-dumpd/dumpd.conf` file is not overwritten during a CLE upgrade if the file already exists. This preserves the site-specific modifications previously made to the file. Cray recommends comparing the site's `/etc/opt/cray-xt-dumpd/dumpd.conf` file content with the `/etc/opt/cray-xt-dumpd/dumpd.conf.example` file provided with each release to identify any changes and then update the site's `/etc/opt/cray-xt-dumpd/dumpd.conf` file accordingly.

If the `/etc/opt/cray-xt-dumpd/dumpd.conf` file does not exist, then the `/etc/opt/cray-xt-dumpd/dumpd.conf.example` file is copied to the `/etc/opt/cray-xt-dumpd/dumpd.conf` file.

The CLE installation and upgrade processes automatically install `dumpd` software, but it must be explicitly enabled.

Enable dumpd

1. In the `nodehealth.conf` configuration file on the shared root (located in `/etc/opt/cray/nodehealth/nodehealth.conf`) change:

```
dumpdon: off
```

to

```
dumpdon: on
```

This allows node health to make requests to `dumpd`.

2. In the same file, set the `maxdumps` variable to some number greater than zero.
3. Specify an action of `dump`, `reboot`, or `dumpreboot` for any tests for which NHC should make a request of `dumpd` when that test fails.
4. In the `dumpd.conf` configuration file on the SMW (in `/etc/opt/cray-xt-dumpd/dumpd.conf`), change:

```
enable: no
```

to

```
enable: yes
```

After the changes to the configuration files are made, NHC will request action from `dumpd` for any test that fails with an action of `dump`, `reboot`, or `dumpreboot`.

The `/etc/opt/cray-xt-dumpd/dumpd.conf` Configuration File

The `dumpd` configuration file, `/etc/opt/cray-xt-dumpd/dumpd.conf`, is located on the SMW. There is no need to change any installation configuration parameters, but a system administrator can edit the `/etc/opt/cray-xt-dumpd/dumpd.conf` file to customize how `dumpd` behaves on the system using the following configuration variables.

enable: `yes|no` Provides a quick on/off switch for all `dumpd` functionality.
Default is `no`.

partitions: `number` Specifies whether or not `dumpd` acts on specific partitions or ranges of partitions. Placing `!` in front of a partition or range disables it.

For example, specifying

```
partitions: 1-10,!2-4
```

enables partitions 1, 5, 6, 7, 8, 9, and 10 but not 2, 3, or 4. Partitions must be explicitly enabled. Leaving this option blank disables all partitions.

disabled_action: `ignore|queue` Specifies what to do when requests come in for a disabled partition. If `ignore` is specified, requests are removed from the database and not acted upon. If `queue` is specified, requests continue to build while `dumpd` is disabled on a partition. When the partition is reenabled, the requests will be acted on. Specifying `queue` is not

recommended if `dumpd` will be disabled for long periods of time, as it can cause SMW stress and database problems.

save_output:
always|errors|
never

Indicates when to save `stdout` and `stderr` from `dumpd` commands that are executed. If `save_output` is set to `always`, all output is saved. If `errors` is specified, output is saved only when the command exits with a nonzero exit code. If `never` is specified, output is never saved.

The default is to save output on errors.

command_output:
directory

Specifies where to save output of `dumpd` commands, per the `save_output` variable. The command output is put in the file `action.pid.timestamp.out` in the directory specified by this option.

Default directory is `/var/opt/cray/dump`.

dump_dir: directory

Specifies the directory in which to save dumps.

Default directory is `/var/opt/cray/dump`.

max_disk: nnnMB|
unlimited

Specifies the amount of disk space beyond which no new dumps will be created. This is not a hard limit; if `dumpd` sees that this directory has less than this amount of space, it starts a new dump, even if that dump subsequently uses enough space to exceed the `max_disk` limit.

The default value is `max_disk: unlimited`.

no_space_action:
action

Specifies a command to be executed if the directory specified by the variable `dump_dir` does not have enough space free, as specified by `max_disk`. For example:

Deletes the oldest dump in the dump directory:

```
no_space_action: rm -rf $dump_dir/$(ls -rt $dump_dir | head -1)
```

Moves the oldest dump somewhere useful:

```
no_space_action: mv $dump_dir/$(ls -t $dump_dir|head -1) /some/dump/archive
```

Sends E-mail to an administrator at admin@fictionalcraysite.com:

```
no_space_action: echo "" | mail -s "Not Enough Space in $dump_dir" \
admin@fictionalcraysite.com
```

The `dumpd`-dbadmin Tool

The `dumpd` daemon sits and waits for requests from NHC (or some other entity using the `dumpd-request` tool on the shared root.) When `dumpd` gets a request, it creates a database entry in the `mznhc` database for the request, and calls the script `/opt/cray-xt-dumpd/default/bin/executor` to read the `dumpd.conf` configuration file and perform the requested actions.

Use the `dumpd-dbadmin` tool to view or delete entries in the `mznhc` database in a convenient manner.

The `dumpd-request` Tool

Use the `dumpd-request` tool to send dump and reboot requests to `dumpd` from the SMW or the shared root. A request includes a comma-separated list of actions to perform, and the node or nodes on which to perform the actions.

A typical request from NHC looks like this:

```
cname: c0-0c1s4n0 actions: halt,dump,reboot
```

A system administrator can define additional actions in the `dumpd.conf` configuration file. To use, execute the `dumpd-request` tool located on the shared root or the SMW. A typical call would be:

```
dumpd-request -a halt,dump,reboot -c c0-0c1s4n0
```

Or

```
dumpd-request -a myaction1,myaction2 -c  
c1-0c0s0n0,c1-0c0s0n1,c1-0c0s0n2,c1-0c0s0n3
```

For this example to work, `myaction1` and `myaction2` must be defined in the `dumpd.conf` file. See the examples in the configuration file for more detail.

Collect Debug Information From Hung Nodes Using the `xtnmi` Command



CAUTION: This is not a harmless tool to use to repeatedly get information from a node at various times; only use this command when debugging data from nodes that are in trouble is needed. The `xtnmi` command output may be used to determine problems such as a core hang.

The sole purpose of the `xtnmi` command is to collect debug information from unresponsive nodes. As soon as that debug information is displayed to the console, the node panics.

For additional information, see the `xtnmi(8)` man page.

Manage System Access

Password Management

The default passwords for the `root` and `crayadm` accounts are the same for the SMW, the boot node, and the shared root.

Default passwords for the `root`, `crayadm`, and MySQL accounts are provided in the *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*. Cray recommends changing these default passwords as part of the software installation process.

Change the Default SMW Passwords

After completing the installation, change the default SMW passwords on the SMW. The SMW contains its own `/etc/passwd` file that is separate from the password file for the rest of the system. To change the passwords on the SMW, log on to the SMW as `root` and execute the following commands:

```
crayadm@smw> su - root
smw# passwd root
smw# passwd crayadm
smw# passwd cray-vnc
smw# passwd mysql
```

For rack-mount SMWs (both R815 and R630 models), it is also necessary to change the default iDRAC password. See [Change the Default iDRAC Password](#) on page 292.

Change root and crayadm Passwords on Boot and Service Nodes

For security purposes, it is desirable to change the passwords for the `root` and `crayadm` accounts on a regular basis.

Use the Linux `passwd` command to change the `/etc/passwd` file. For information about using the `passwd` command, see the `passwd(1)` man page.

1. Log on to the boot node.

```
smw# ssh root@boot
```

2. Change the passwords on the boot node.

The boot node contains its own `/etc/passwd` file that is separate from the password file for the rest of the system.

```
boot# passwd root
boot# passwd crayadm
```

The system prompts for the new passwords.

3. Invoke `xtopview` to access the shared root in order to change passwords on other service nodes. If the SDB node is not started, add the `-x /etc/opt/cray/sdb/node_classes` option to the `xtopview` command.

```
boot# xtopview
```

4. Change the passwords on the other service nodes. These commands must be invoked on the shared root.

```
default/#!/# passwd root
default/#!/# passwd crayadm
default/#!/# exit
```

Again, the system prompts for the new passwords.

Change the root Password on CNL Compute Nodes

For compute nodes, the `root` account password is maintained in the `/opt/xt-images/templates/default/etc/shadow` file on the SMW.

To make these changes for a system partition, rather than for the entire system, replace `/opt/xt-images/templates` with `/opt/xt-images/templates-pN`, where *N* is the partition number.

1. Copy the master password file to the template directory.

```
smw# cp /opt/xt-images/master/default/etc/shadow \
/opt/xt-images/templates/default/etc/shadow
```

2. Edit the password file to include a new encrypted password for the `root` account.

```
smw# vi /opt/xt-images/templates/default/etc/shadow
```

3. Update the boot image by following the steps in [Prepare Compute and Service Node Boot Images](#) on page 37.

Change the HSS Data Store (MySQL) Password

Use the `hssds_init` command to change the HSS data store (MySQL) `root` password. The `hssds_init` command prompts for the current HSS data store (MySQL) `root` password. When the current and new passwords are typed, they are not echoed.

The `hssds_init` utility is run by the `SMWinstall` command, and does not need to be run during installation of an SMW release package.

For additional information, see the `hssds_init(8)` man page.

Change Default MySQL Passwords on the SDB

Access to MySQL databases requires a user name and password. The MySQL accounts and privileges are

MySQL basic	Read access to most tables; most applications use this account
MySQL sys_mgmt	Most privileged; access to all information and commands

For security, Cray recommends changing the default passwords for MySQL database accounts. The valid characters for use in MySQL passwords are:

```
! " # $ % & ' ( )
* + , - . / 0 1 2 3
4 5 6 7 8 9 : ; < =
> ? @ A B C D E F G
H I J K L M N O P Q
R S T U V W X Y Z [
\ ] ^ _ ` a b c d e
f g h i j k l m n o
p q r s t u v w x y
z { | } ~
```

1. If a site-specific MySQL password for root has not been set, complete this step.

- a. Log on to the SDB.

```
boot:~ # ssh root@sdb
```

- b. Invoke the MySQL monitor.

```
sdb:~ # mysql -h localhost -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.5.31-log Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

- c. Set passwords. Use the actual name of the system database (SDB) node if it is not named `sdb`. For example, the node could be named `sdb-p3` on a partitioned system.

```
mysql> set password for 'root'@'localhost' = password('newpassword');
Query OK, 0 rows affected (0.00 sec)
mysql> set password for 'root'@'%' = password('newpassword');
Query OK, 0 rows affected (0.00 sec)
mysql> set password for 'root'@'sdb' = password('newpassword');
Query OK, 0 rows affected (0.00 sec)
```

2. Optional: Set a site-specific password for other MySQL database accounts.

- a. Change the password for the `sys_mgmt` account. This requires an update to `.my.cnf` in step 4 on page 67.

```
mysql> set password for 'sys_mgmt'@'%' = password('newpassword');
Query OK, 0 rows affected (0.00 sec)
```

- b. Change the password for the `basic` account. This requires an update to `/etc/opt/cray/sysadm/odbc.ini` in step 5 on page 68.

Changing the password for the `basic` MySQL user account will not provide any added security. This read-only account is used by the system to allow all users to run `xtprocdadmin`, `xtnodestat`, and other commands that require SDB access.

```
mysql> set password for 'basic'@'%' = password('newpassword');
Query OK, 0 rows affected (0.00 sec)
```

The connection may time out when making changes to the MySQL database, but it is automatically reconnected. If this happens, messages similar to the following are displayed. These messages may be ignored.

```
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 21127
Current database: *** NONE ***

Query OK, 0 rows affected (0.00 sec)
```

3. Exit from MySQL and the SDB.

```
mysql> exit
Bye
sdb# exit
```

4. Optional: If a site-specific password for `sys_mgmt` was set, update the `.my.cnf` file for `root` with the new password. Additionally, update the `.odbc.ini.root` file with the new password.

a. Edit `.my.cnf` for `root` on the boot node.

```
boot# vi /root/.my.cnf

[client]
user=sys_mgmt
password=newpassword
```

b. Edit `.my.cnf` for `root` in the shared root.

```
boot# xtopview
default:// # vi /root/.my.cnf

[client]
user=sys_mgmt
password=newpassword
```

c. Exit `xtopview` and edit `.odbc.ini.root` for the `root` on the boot node. Update **each** database section with the new password.

```
default:// # exit
boot# vi /root/.odbc.ini.root

Driver          = MySQL_ODBC
Description     = Connector/ODBC Driver DSN
USER            = sys_mgmt
PASSWORD       = newpassword
```

- d. Copy `.odbc.ini.root` to `.odbc.ini`.

```
boot:~ # cp -p /root/.odbc.ini.root /root/.odbc.ini
```

- e. Invoke `xtopview` and edit `.odbc.ini.root` for the root in the shared root. Update **each** database section with the new password.

```
boot# xtopview
default:/ # vi /root/.odbc.ini.root
```

```
Driver          = MySQL_ODBC
Description     = Connector/ODBC Driver DSN
USER           = sys_mgmt
PASSWORD       = newpassword
```

- f. Exit `xtopview` and restart the SDB service on all service nodes.

```
default:/ # exit
boot:~ # pdsh -a /etc/init.d/sdb restart
```

5. Optional: If a site-specific password for `basic` was set, update **each** datasource name in the `/etc/opt/cray/sysadm/odbc.ini` file with the new password. Additionally, update the `/root/.odbc.ini` file with the new password.

- a. Edit `/etc/opt/cray/sysadm/odbc.ini` for the `basic` user on the boot node. Update **each** database section with the new password.

```
boot# vi /etc/opt/cray/sysadm/odbc.ini
```

```
Driver          = MySQL_ODBC
Description     = Connector/ODBC Driver DSN
USER           = basic
PASSWORD       = newpassword
```

- b. Invoke `xtopview` and edit `/root/odbc.ini` in the shared root. Update **each** database section with the new password.

```
boot# xtopview
default:/ # vi /root/odbc.ini
```

```
Driver          = MySQL_ODBC
Description     = Connector/ODBC Driver DSN
USER           = basic
PASSWORD       = newpassword
```

- c. Exit `xtopview`.

Assign and Change User Passwords

Because Cray systems have a read-only shared-root configuration, users cannot execute the `passwd` command on a Cray system to change their password. If a site has an external authentication service such as Kerberos or LDAP, users should follow site instructions to update their passwords. If a site does not have external authentication set up, the system administrator can implement a manual mechanism, such as having users change their password on an external system and then periodically copying their entries in the

external `/etc/passwd`, `/etc/shadow`, and `/etc/group` files to the equivalent Cray system files in the default `xtopview`.



WARNING: Be careful to not overwrite Cray system accounts (`crayadm`, `cray_vnc` and standard Linux accounts such as `root`) in the `/etc/passwd`, `/etc/shadow`, and `/etc/group` files.

Logins That Do Not Require Passwords

All logins must have passwords; however, an administrator can set up a passwordless `ssh` by creating an `ssh` key with a null passphrase and distributing that `ssh` key to another computer.

While the key-based authentication systems such as OpenSSH are relatively secure, convenience and security are often mutually exclusive. Setting up passphrase-less `ssh` is convenient, but the security ramifications can be dire; if the local host is compromised, access to the remote host will be compromised as well.

If passphrase-less authentication is needed at the site, Cray recommends using `ssh-agent` if available, or take other steps to mitigate risk.

User Account Administration

Cray systems support several types of accounts:

Boot node accounts	<p>The boot node does not support user accounts. The only accounts that are supported on the boot node are <code>root</code> (superuser), <code>crayadm</code> (administrator), and accounts for various services such as network time protocol (NTP). To modify configuration files, the administrator must become superuser by supplying the <code>root</code> account password.</p> <p>The boot node has an <code>/etc/passwd</code> file that is separate from the password file for the rest of the system. For a list of default passwords, see <i>Install and Configuring Cray Linux Environment (CLE) Software (S-2444)</i></p>
SMW user accounts	<p>Access is managed using local files (e.g., <code>/etc/passwd</code>, <code>/etc/shadow</code>). In addition to the standard Linux system accounts, Cray includes an account named <code>crayadm</code> that is used for many of the Cray system management functions such as booting the system.</p>
Virtual Network Computing (VNC) account	<p>Cray provides a VNC account on the SMW; for details, see Remote Access to SMW on page 329.</p>
Login node accounts	<p>Typically, user account and passwords are managed through an external LDAP or Kerberos server. It is, however, possible to manage accounts that must be directly maintained on the Cray system when LDAP or Kerberos service are not available.</p> <p>User accounts are set up on the shared-root file system by using the <code>xtopview</code> command. Cray systems support 16-bit and 32-bit user IDs (UIDs). The 16-bit user IDs run 0-65535; that is 0-($2^{16}-1$). The 32-bit user IDs run 0-($2^{32}-1$), although Cray systems are limited to a maximum of 65,536 user accounts, including those that are predefined, such as <code>root</code>, <code>crayadm</code>, and <code>mysql</code>.</p>

Add or Modify a Group

Follow this procedure to add or modify groups to the shared root for login nodes. For more information, see the `groupadd(8)` man page.

To modify a group in the shared root, use the `groupmod` command in the default view of the `xtopview` command. For more information, see the `groupmod(8)` man page.

1. Log on to the boot node and invoke `xtopview`.

```
smw# ssh root@boot
boot# xtopview
```

2. Add a group.

For this example, add the group `xtusers` with `gid 5605`.

```
default/#!/# groupadd -g 5605 xtusers
```

3. Exit `xtopview`

```
default/#!/# exit
```

Group `xtusers` is added to `/etc/group`.

Add or Modify a User Account

Follow this procedure to add user accounts to the shared root for login nodes. For more information, see the `useradd(8)` and `passwd(1)` man pages.

To modify user accounts on the shared root, use the `usermod` command in the default view of the `xtopview` command. For more information, see the `usermod(8)` man page.

1. Log on to the boot node and invoke `xtopview`.

```
smw# ssh root@boot
boot# xtopview
```

2. Add a user.

For this example, add user `bobp` with user ID `12645`, group `5605`, home directory `bobp`, and the `/bin/bash` login shell:

```
default/#!/ # useradd -d /home/users/bobp -g 5605 -s /bin/bash -u 12645 bobp
```

3. Exit `xtopview` and log on to a login node.

```
default/#!/ # exit
boot# ssh root@login
```

4. Create the user's home directory.

```
login# mkdir -p /home/users/bobp
```

5. Change the ownership and group of the user's home directory.

```
login# chown -R bobp:xtusers /home/users/bobp
```

6. Set an account password in either `/etc/passwd` or `/etc/shadow`.

Remove a User or Group Account

1. Remove all files, jobs, and other references to the user.
2. Log on to the boot node and invoke `xtopview`.

```
smw# ssh root@boot
boot# xtopview
```

3. Remove the user with the `userdel` command. For this example, remove the user `bobp` and the user's home directory.

```
default/./# userdel -r bobp
```

4. Exit `xtopview` and log on to the login node.

```
default/./# exit
boot# ssh root@login
```

5. Remove the user's home directory.

```
login# rm -rf /home/users/bobp
```

6. Exit for the login node.

Assign Groups of Compute Nodes to a User Group

Use the `label` attribute of the `/etc/opt/cray/sdb/attr.defaults` file to assign groups of CNL compute nodes to specific user groups without the need to partition the system. For more information, see [Initially Set Node Attributes](#) on page 153.

Associate Users with Projects

The system administrator can assign project names for users to submit jobs in order to determine project charges. Project names can be up to 80 characters long.

To associate users with project names, add the following line to their individual login scripts in their home directories:

```
set_account a_project_name
```

After accounts are set, users do not have to manually run the `set_account` command at each login.

Users running batch jobs can typically set a project code within the workload manager. For example, when using PBS Professional, a user can set a project code with the `ENVIRONMENT` variable. This associates the project

code with the job in the accounting database. For more information, see the documentation provided by the batch system vendor.

Enable LDAP Support for User Authentication

To enable LDAP support, several files must be edited in addition to any other site-specific standard LDAP configuration setting changes.

1. Log on to the boot node and invoke `xtopview`.

```
smw:~ # ssh root@boot
boot:~ # xtopview
```

2. Edit `/etc/pam.d/common-account-pc` and replace:

```
account required      pam_unix2.so
```

with

```
account sufficient    pam_ldap.so config=/etc/openldap/ldap.conf
account required      pam_unix2.so
```

3. Edit `/etc/pam.d/common-auth-pc` and replace:

```
account required      pam_env.so
auth required         pam_unix2.so
```

with

```
account required      pam_env.so
auth sufficient        pam_ldap.so config=/etc/openldap/ldap.conf
auth required         pam_unix2.so
```

4. Edit `/etc/pam.d/common-password-pc` and replace:

```
password required     pam_pwcheck.so nullok
password required     pam_unix2.so     nullok use_authok
```

with

```
password required     pam_pwcheck.so nullok
password sufficient    pam_ldap.so config=/etc/openldap/ldap.conf
password required     pam_unix2.so     nullok use_authok
```

5. Edit `/etc/pam.d/common-session-pc` and replace:

```
session required      pam_limits.so
session required      pam_unix2.so
session optional      pam_umask.so
```

with

```
session required      pam_limits.so
session sufficient    pam_ldap.so config=/etc/openldap/ldap.conf
session required      pam_unix2.so
session optional      pam_umask.so
```

6. Make site-specific changes (on the boot root in the `xtopview` default view) to the `/etc/openldap/ldap.conf` or `/etc/ldap.conf`, `/etc/nsswitch.conf`, `/etc/sysconfig/ldap`, `/etc/passwd`, and `/etc/group` files.
7. Exit `xtopview`.

Adding the LDAP servers to the local host file means that DNS does **not** need to run on the SDB and MDS. The SDB and MDS need access to the LDAP server. Configure this access through RSIP or NAT; see [Configure Realm-specific IP Addressing \(RSIP\)](#).

Load Balancing Across Login Nodes

Having all users log on to the same login node may overload the node. (Also, see the Caution in [Login Nodes](#).) For typical interactive usage, a single login node is expected to handle 20 to 30 batch users or 20 to 40 interactive users with double this number of user processes.



CAUTION: Login nodes, as well as other service nodes, do not have swap space. If users consume too many resources, Cray service nodes can run out of memory. When an out of memory condition occurs, the node can become unstable or may crash. System administrators should take steps to manage system resources on service nodes. For example, resource limits can be configured using the `pam_limits` module and the `/etc/security/limits.conf` file. For more information, see the `limits.conf(5)` man page.

The system administrator can use the `lbname`d load-balancing software to distribute logins to different login nodes. The `lbname`d daemon is a name server that gathers the output of `lbcd` client daemons to select the least loaded node, provides DNS-like responses, interacts with the corporate DNS server, and directs the user login request to the least busy login node.

Because `lbname`d runs on the SMW, `eth0` on the SMW must be connected to the same network from which users log on the login nodes. If security considerations do not allow the SMW to be located on the public network, `lbname`d may be installed on an external server. This can be any type of computer running the SUSE Linux Enterprise Server (SLES) operating system (not a 32-bit system). However, this option is not a tested or supported Cray configuration.

Configure the Load Balancer

NOTE: The load balancer service is optional on systems that run CLE.

The load balancer can distribute user logins to multiple login nodes, allowing users to connect by using the same Cray host name, for example `xthostname`.

Two main components are required to implement the load balancer, the `lbname`d service (on the SMW and Cray login nodes) and the site-specific domain name service (DNS).

When an external system tries to resolve *xthostname*, a query is sent to the site-specific DNS. The DNS server recognizes *xthostname* as being part of the Cray domain and shuttles the request to `lbname`d on the SMW. The `lbname`d service returns the IP address of the least-loaded login node to the requesting client. The client connects to the Cray system login node by using that IP address.

The CLE software installation process installs `lbname`d in `/opt/cray-xt-lbname`d on the SMW and in `/opt/cray/lbcd` on all service nodes. Configure `lbname`d by using the `lbname`d.conf and `poller`.conf configuration files on the SMW. For more information about configuring `lbname`d, see the `lbname`d.conf(5) man page.

Configure `lbname`d on the SMW

1. If site-specific versions of `/etc/opt/cray-xt-lbname`d/`lbname`d.conf and `/etc/opt/cray-xt-lbname`d/`poller`.conf do not already exist, copy the provided example files to these locations.

```
smw:~ # cd /etc/opt/cray-xt-lbname/d/
smw:/etc/opt/cray-xt-lbname/d/ # cp -p lbname.d.conf.example lbname.d.conf
smw:/etc/opt/cray-xt-lbname/d/ # cp -p poller.conf.example poller.conf
```

2. Edit the `lbname`d.conf file on the SMW to define the `lbname`d host name, domain name, and polling frequency. For example, if `lbname`d is running on the host name `smw.mysite.com`, set the login node domain to the same domain specified for the `$hostname`. The Cray system *xthostname* is resolved within the domain specified as `$login_node_domain`.

```
smw:/etc/opt/cray-xt-lbname/d/ # vi lbname.d.conf
```

```
$poller_sleep = 30;
$hostname = "mysite-lb";
$lbname_domain = "smw.mysite.com";
$login_node_domain = "mysite.com";
$hostmaster = "rootmail.mysite.com";
```

3. Edit the `poller`.conf file on the SMW to configure the login node names. Because `lbname`d runs on the SMW, `eth0` on the SMW must be connected to the same network from which users log on to the login nodes. Do not put the SMW on the public network.

```
smw:/etc/opt/cray-xt-lbname/d/ # vi poller.conf
```

```
#
# groups
# -----
# login      mycray1-mycray3

mycray1 1 login
mycray2 1 login
mycray3 1 login
```

Install the Load Balancer on an External "White Box" Server

Install `lbname`d on an external "white box" server as an alternative to installing it on the SMW. **Cray does not test or support this configuration.** A "white box" server is any workstation or server that supports the `lbname`d service.

1. Shut down and disable `lbname`d.

```
smw:~# /etc/init.d/lbname stop
smw:~# chkconfig lbname off
```

2. Locate the `cray-xt-lbname`d RPM on the Cray CLE 5.0.UP nn Software media and install this RPM on the "white box." Do **not** install the `lbcd` RPM.
3. Follow the instructions in the `lbname.conf(5)` man page to configure `lbname`d, taking care to substitute the name of the external server wherever `SMW` is indicated, then enable the service.

Prevent Login Node Hangs

If all available processes on a login node are in use, the node can hang or become nearly unresponsive. A hang of this type can be identified primarily by the presence of `cannot fork` error messages, but it is also associated with an unusually large number of processes running concurrently, the machine taking several minutes to make a prompt available, or never making a prompt available. In the case of an overwhelming number of total processes, it is often a large number of the same process overwhelming the system, which indicates a `fork()` system call error in that particular program.

This problem can be prevented by making a few changes to configuration files in `/etc` on the shared root of the login node. These configurations set up the `ulimit` built-in and the Linux Pluggable Authentication Module (PAM) to enforce limits on resources as specified in the configuration files. There are two types of limits that can be specified, a soft limit and a hard limit. Users receive a warning when they reach the soft limit specified for a resource, but they can temporarily increase this limit up to the hard limit using the `ulimit` command. The hard limit can never be exceeded by a normal user. Because of the shared root location of the configuration files, the changes must be made from the boot node using the `xtopview` tool.

1. Log on to the boot node and invoke `xtopview` for the login nodes, where `login` is the class name for login nodes.

```
boot:~ # xtopview -c login
```

2. Edit the `/etc/security/limits.conf` file and add the following lines, where `soft_lim_num` and `hard_lim_num` are the number of processes at which the hard and soft limits are to be enforced. The `*` represents "apply to all users" but can also be configured to apply specific limits by user or group (see the `limits.conf` file for further options).

```
class/login:/ # vi /etc/security/limits.conf
```

```
* soft nproc      soft_lim_num
* hard nproc      hard_lim_num
```

- Verify that the following line is included in the appropriate PAM configuration files for any authentication methods for which limits are to be enforced; the PAM configuration files are located in the `/etc/pam.d/` directory. For example, to enforce limits for users connecting through `ssh`, add the `pam_limits.so` line to the file `/etc/pam.d/sshd`. Other applicable authentication methods to also include are `su` in the file `/etc/pam.d/su` and local logins in `/etc/pam.d/login`.

```
session    required    pam_limits.so
```

For more information about the Pluggable Authentication Module (PAM), see the `PAM(8)` man page.

- Exit to the boot node; the changes are effective immediately on login nodes.

```
class/login:/ # exit
```

- Log on to a login node to test that the limits are in place. which should return the number specified as the soft limit for the number of processes available to a user, for example:

For more information about using the `ulimit` command, see the `ulimit(P)` man page.

```
boot:~ # ssh login
login:~ # ulimit -u
```

The number specified as the soft limit for the number of processes available to a user is displayed.

Set Disk Quotas for a User on the Cray Local, Non-Lustre File System

The `quota` and `quota-nfs` RPMs are installed by default. Disk quotas are activated for a user on service nodes on the Cray local, non-Lustre file system by invoking two boot scripts, as discussed in the `README.SUSE` file located in `/usr/share/doc/packages/quota`.

- Follow the procedure described in the `README.SUSE` file located in `/usr/share/doc/packages/quota`.

IMPORTANT: When following the procedure in the `README.SUSE` file, remember that any commands should be issued from within the default view of `xtopview`. Also, use the `chkconfig` command instead of the `yast2` run level editor to turn on `quota` and `quotad` services:

```
boot:~ # xtopview
default:/ # chkconfig boot.quota on
default:/ # chkconfig quotad on
default:/ # exit
```

- Start the services on all service nodes; either reboot the system or execute `/etc/init.d/boot.quota start` and `/etc/init.d/quotad start` on each service node.
- Use standard Linux quota commands to enable, check, and set quotas for each user.
 - Enable quotas (`quotaon` command)
 - Check quotas (`quotacheck` command)
 - Set quotas (`edquota` command)

When a quota is exceeded, the quotas subsystem warns the user that the allotted limit has been exceeded. Some extra space is allocated for current work, that is, there is a hard limit and a soft limit.

For more information, see the `quotaon(8)`, `quotacheck(8)`, and `edquota(8)` Linux man pages.

About Modules and Modulefiles

The Modules software package enables users to modify their environment dynamically by using *modulefiles*. The `module` command is a user interface to the Modules package. The `module` command interprets modulefiles, which contain Tool Command Language (Tcl) code, and dynamically modifies shell environment variables such as `PATH`, and `MANPATH`.

The shell configuration files `/etc/csh.cshrc.local` and `/etc/bash.bashrc.local` contain module commands that establish the default user environment, which is set by the system at login time.

To support customer-specific needs, the system administrator can create modulefiles for a product set for the users. For more information about the Modules software package, see the `module(1)` and `modulefile(4)` man pages.

System-wide Default Modulefiles

The `/etc/csh/cshrc.local` and `/etc/bash/bashrc.local` files load `Base-opts`, which loads two lists of modulefiles: a default list and a site-specified local list. The default list differs between the SMW and the Cray system. On the SMW, the file `/etc/opt/cray/modules/Base-opts.default.SMW` contains the list of the SMW modulefiles to load by default. On the Cray system, the file `/etc/opt/cray/modules/Base-opts.default` contains the list of SMW modulefiles to load by default.

Additionally, all the modulefiles listed in the file `/etc/opt/cray/modules/Base-opts.default.local` are loaded. Edit this file to make site-specific changes. The `/etc/opt/cray/modules/Base-opts.default.local` file initially includes the `admin-modules` modulefile, which loads a full set of modulefiles. The `admin-modules` modulefile does not need to be manually loaded, unless it was removed from the default list. The CLE installation process removes `admin-modules` modulefile from the default list on login nodes.

The files on the Cray system are installed on both the boot root and the shared root.

An example file, `/etc/opt/cray/modules/Base-opts.default.local.example`, is also provided. The example file is a copy of the `/etc/opt/cray/modules/Base-opts.default.local` file provided for an initial installation.

Configure the Default Programming Environment (PE)

The Cray, PGI, GCC, PathScale (systems only), and Intel compilers are available to Cray system users, if installed. A programming environment is comprised of a compiler and its supporting libraries and tools.

The system wide default PE is set to `PrgEnv-cray` in the `PE-set-up` block in the `/etc/*rc.local` files. For Cray XE sites without a Cray Compiling Environment license, `PrgEnv-pgi` is the default. Do not edit the `PE-set-up` block to alter the system default PE.

To change the default PE and add more PE user defaults, add appropriate instructions to the `SITE-set-up` block. The instructions in the `SITE-set-up` block are not altered by operating system installations. The

instructions are evaluated after the `PE-set-up` block, therefore, it is important that instructions added in the `SITE-set-up` block do not conflict with those in the `PE-set-up` block.

For this example, change the default PE to `PrgEnv-abc` in the `SITE-set-up` block in `/etc/*rc.local` files:

```
module unload PrgEnv-cray
module load PrgEnv-abc
```

Targeting modules are released in the `xt-asyncpe` and `craype` packages and installed in the `/opt/cray/xt-asyncpe/default/modulefiles` or `/opt/cray/craype/default/modulefiles` directories, respectively.

For this example, display the list of available targeting modulefiles:

```
% module avail xt-asyncpe
% module avail craype
```

If there are no targeting modules loaded in the user's environment, the compiler driver scripts (`cc`, `CC`, `ftn`) set the CPU target to `sandybridge` on systems and to `interlagos` on and Cray XK systems. To change the default CPU target, configure `/etc/*rc.local` to load the appropriate `craype-*` target module.

For this example, set the default target to `xyz` in the `/etc/*rc.local` files by modifying the `SITE-set-up` section:

```
module add craype-xyz
```

Other commonly used modules and settings in the `SITE-set-up` block are:

```
# load a workload manager
module load pbs
# define target architecture
module load craype-abudhabi
# mpich2 is not loaded by PrgEnv-cray
module load cray-mpich2
# enable abnormal termination processing (see intro_atp)
setenv ATP_ENABLED 1
```

Deny or Allow Services Using the `pam_listfile` Module in the Shared Root Environment

The Linux `pam_listfile` Pluggable Authentication Module (PAM) can be used to maintain a list of authorized users. Using the `pam_listfile` PAM may also help to reduce impacts on service nodes if users consume too many resources.

The `pam_listfile` PAM requires that the file specified with the `file=` parameter be a regular file. The usual approach of storing the file in the `/etc` directory does not work in the shared-root environment of Cray systems

because files in the `/etc` directory are symbolic links. Therefore, the required file must be created in a directory other than the `/etc` directory. For example, it can be placed in persistent `/var` or another directory that is not controlled by the shared root.

Create a `pam_listfile` list file

For this example, add authorized users to a `pam_listfile` module file. This example assumes an empty `pam_listfile` called `/var/path_to/pam_listfile_authorized_users_list` has been created.

```
boot:~ # xtopview -x /etc/opt/cray/sdb/node_classes -c login
class/login/:# vi /var/path_to/pam_listfile_authorized_users_list
```

```
user1
user2
...
```

Add a line to `/etc/pam.d/ssh` to enable `pam_listfile`

```
class/login/:# vi /etc/pam.d/ssh
```

```
auth required pam_listfile.so file=/var/path_to/
pam_listfile_authorized_users_list
```

Nodes can have different `pam_listfile` module files. To do this, create the module files and specialize the PAM configuration files (such as `pam.d/ssh`) to point to them.

Stop a Job Running in Interactive Mode

If the job is running on a CNL compute node in interactive mode (through `aprun`), invoke the following command:

```
% apkill - signal apid
```

This sends a signal to all processes that are part of the specified application (`apid`); signal 15 (`SIGTERM`) is sent by default.

The signaled application must belong to the current user unless the user is a privileged user. For more information, see the `aprun(1)` and `apkill(1)` man pages.

Monitor the System

Display Release Levels and Configuration Information

Following a successful installation, the following information is available.

Display Installed SMW Release Level

Following a successful installation, the file `/opt/cray/hss/default/etc/smw-release` is populated with the installed SMW release level.

```
% cat /opt/cray/hss/default/etc/smw-release
7.2.UP03
```

Display Current and Installed CLE Release Information

Following a successful installation, several files in the `/etc/opt/cray/release` directory are populated with various CLE release information.

- `/etc/opt/cray/release/xtrelease` Contains the `xtrelease` (build number).
- `/etc/opt/cray/release/clerelease` Contains the most recently installed CLE version number and update level.
- `/etc/opt/cray/release/CLEinfo` Contains a list of install-time options that other scripts may want access to, such as network type, installer version, and existence of the Lustre® file system. The `CLEinfo` file is an install-time "snapshot" and does not change.

Display the current `xtrelease` value:

```
crayadm@login:~> cat /etc/opt/cray/release/xtrelease
DEFAULT=5.2.14
```

Display the most recently installed CLE release information

```
crayadm@login:~> cat /etc/opt/cray/release/CLEinfo
CLERELEASE=5.2.UP01
INSTALLERVERSION=b12
LUSTRE=yes
NETWORK=gem
XTRELEASE=5.2.14
```

The `CLEinfo` file is an install-time "snapshot" and does not change; the release values may not be the currently booted version on the system.

Display Boot Configuration Information

Use the `xtcli` command to display the configuration information for the primary and backup boot nodes, the primary and backup SDB nodes, and the `cpio` path.

Display boot configuration information for the entire system

```
crayadm@smw:~> xtcli boot_cfg show
Network topology: class 1
=== xtcli_boot_cfg ===
[boot]: c0-0c0s0n1:ready,c0-0c0s0n1:ready
[sdb]: c1-0c0s2n1:ready
[cpio_path]: /tmp/boot/kernel.cpio_5.2.14-scrub-off
```

Display boot configuration information for one partition in a system

```
crayadm@smw:~> xtcli part_cfg show pN
```

Where `pN` is the partition number. `p0` is always the whole system.

Manage Log Files Using CLE and HSS Commands

Boot, diagnostic, and other Hardware Supervisory System (HSS) events are logged on the SMW in the `/var/opt/cray/log` directory, which is created during the installation process. The time-stamped `bootinfo`, `console`, `consumer`, and `netwatch` log files are located in the `/var/opt/cray/log/sessionid` directory by default.

For example, the HSS `xtbootsys` command starts the `xtconsole` command, which redirects the output to a time-stamped log file, such as `/var/opt/cray/log/p0-20120716t104708/console-20120716`.

The `SMWinstall`, `SMWconfig`, and `SMWinstallCLE` commands create several detailed log files in the `/var/adm/cray/logs` directory. The log files are named using the PID of the `SMWinstall` or the `SMWinstallCLE` command; the exact names are displayed when the command is invoked.

CLE logs are saved on the SMW in `/var/opt/cray/log/sessionid`.

Controller logs are saved on the SMW

in `/var/opt/cray/log/controller/cabinet/controller/messages-yyyyymmdd`, where *cabinet* is of the form `c0-0`, `c1-0`, etc.; and *controller* is either of the form `c0-0`, `c1-0` for cabinet controllers (L1) or `c0-0c0s0` for blade controllers (L0).

For more information, see the `intro_llm_logfiles(5)` man page.

Filter the Event Log

The `xtlogfilter` command enables the system administrator to filter the event log for information such as the time a particular event occurred or messages from a particular cabinet.

For more information, see the `xtlogfilter(8)` man page.

Finding information in the event log

For this example, search for all console messages from node `c9-2c0s3n2`:

```
crayadm@smw:~> xtlogfilter -f /var/opt/cray/log/event-yyyymmdd c9-2c0s3n2
```

Add Entries to Log Files

The system administrator can add entries (e.g., the start or finish of system activities) to the `syslog` with the `logger` command. The entry is then available to anyone who reads the log.

For more information, see the `logger(1)` man page.

Add entries to syslog file

For this example, mark the start of a new system test:

```
login# logger -is "Start of test 4A $(date) "
Start of test 4A Thu Jul 14 16:20:43 CDT 2011
```

The system log shows:

```
Jul 14 16:20:43 nid00003 xx[21332]:
Start of test 4A Thu Jul 13 16:20:43 CDT 2012
```

Examine Log Files

Time-stamped log files of boot, diagnostic and other HSS events are located on the SMW in the `/var/opt/cray/log` directory. The time-stamped `bootinfo`, `console`, `consumer`, and `netwatch` log files are located in the `/var/opt/cray/log/sessionid` directory by default.

For example, the HSS `xtbootsys` command starts the `xtconsole` command, which redirects the output to a time-stamped log file, such as `/var/opt/cray/log/p0-20120716t104708/console-20120716`.

The `SMWinstall`, `SMWconfig`, and `SMWinstallCLE` commands create several detailed log files in the `/var/adm/cray/logs` directory. The log files are named using the PID of the `SMWinstall` or the `SMWinstallCLE` command; the exact names are displayed when the command is invoked.

Remove Old Log Files

Deprecated: The `xtclean_logs` command is deprecated. Functionality provided by `xtclean_logs` is provided by the `xtrim` utility.

The `xtrim` utility provides a simple and configurable method to automate the compression and deletion of old log files. The `xtrim` utility is intended to be run on the SMW from `cron` and is automatically configured to do this as part of the SMW software installation process. Review the `xtrim.conf` configuration file and ensure that `xtrim` will manage the desired directories and that the compression and deletion times are appropriate.

The `xtrim` utility does not perform any action unless the `--confirm` flag is used (to avoid unintended actions), nor will `xtrim` perform any action on open files. All actions are based on file-modified time.

For additional information, see the `xtrim(8)` and `xtrim.conf(5)` man pages.

SEC Software for Log Monitoring and Event Processing

The simple event correlator (SEC) is released under the GNU Public License (GPL) v2. SEC parses every line being appended to system log files, watches for specific strings to show up that represent significant events occurring in the system, and sends out email notification that the event has occurred.

Find additional information about SEC at:

- The following GPL-related documents, contained in the `sec-2.7.0` package (RPM):
 - `/usr/share/doc/sec/COPYING`
 - `/usr/share/doc/sec/ChangeLog`
 - `/usr/share/doc/sec/README`
 - `/usr/share/man/man1/sec.1.gz`
- The website: <http://simple-evcorr.sourceforge.net>. At that website, the `sec(1)` man page and FAQ are especially helpful, as is *Working with SEC – the Simple Event Correlator* by Jim Brown (a tutorial paper where part 1 (2003) provides an introduction to SEC and part 2 (2004) covers several advanced topics).
- Searches from the newsgroup: http://sourceforge.net/search/?group_id=42089&type_of_search=mlists.
- A brief introduction to SEC under the following article, (located under "Tools, Tips, and Tweaks") at: <http://arstechnica.com/information-technology/2005/05/linux-20050519/>.
- On an SMW with the SMW release installed, type `man sec` to display the `sec(1)` man page.

For information about optionally using SEC for the Cray system, see Configure Cray SEC Software (S-2542).

Use `cray_pam` to Log Failed Login Attempts

The `cray_pam` module is a Pluggable Authentication Module (PAM) that, when configured, provides information to the user at login time about any failed login attempts since their last successful login. The module provides:

- Date and time of last successful login
- Date and time of last unsuccessful login
- Total number of unsuccessful logins since the user's last successful login

Cray recommends configuring login failure logging on all service nodes. The RPMs are installed by default on the boot root and shared root file systems.

To use this feature, configure the `pam_tally` and `cray_pam` PAM modules. The PAM configuration files provided with the CLE software allow the administrator to manipulate a common set of configuration files that will be active for all services.

The `cray_pam` module requires an entry in the PAM `common-auth` and `common-session` files or an entry in the PAM `auth` section and an entry in the PAM `session` section of any PAM application configuration file. Use of the common files is typically preferable so that other applications such as `su` also report failed login information; for example:

```
crayadm@boot:~> su -  
2 failed login attempts since last login.
```

```
Last failure Thu May  8 11:41:20 2008 from smw.
boot:~ #
```

For each login attempt, a per-user counter is updated. When a successful login occurs, the statistics are displayed and the counter is cleared. The default location of the `pam_tally` counter file is `/var/log/faillog`. Additionally, `cray_pam` uses a temporary directory, by default, `/var/opt/cray/faillog`, to store information about the users. Change these defaults by editing `/etc/opt/cray/pam/faillog.conf` and by using the `file=` option for each `pam_tally` and `cray_pam` entry. An example `faillog.conf` file is located in `/opt/cray/pam/pamrelease-version/etc`.

The administrator can configure a number of nodes to share information by modifying the default location for these directories to use a common set of directories, writable to all nodes. Edit `/etc/opt/cray/pam/faillog.conf` to reflect an alternate, `root-writable` directory. Configure `pam_tally` to save tally information in an alternate location using the `file=` option; each entry for `cray_pam` must also include the `file=` option to specify the alternate location.

For additional information on using the `cray_pam` PAM module, see the `pam(8)` and `pam_tally(8)` man pages.

Limitations

- If a login attempt fails, `cray_pam`, within the `auth` section, creates a temporary file; but because the login attempt failed, the `session` section is not called and, as a result, the temporary file is not removed. This is harmless because the file will be overwritten at the next login attempt and removed at the next successful login.
- Logins that occur outside of the PAM infrastructure will not be noted.
- Host names are truncated after 12 characters. This is a limitation in the underlying `faillog` recording.
- The `cray_pam` module requires `pam_tally` to be configured.

Configure `cray_pam` to Log Failed Login Attempts

1. Edit the `/etc/pam.d/common-auth`, `/etc/pam.d/common-account`, and `/etc/pam.d/common-session` files on the boot node.

In these examples, the `pam_faillog.so` and `pam_tally.so` entries can include an optional `file=/path/to/pam_tally/counter/file` argument to specify an alternate location for the tally file.

- a. Edit the `/etc/pam.d/common-auth` file and add the following lines as the first and last entries:

```
boot:~ # vi /etc/pam.d/common-auth

auth required pam_faillog.so [file=alternatepath]
auth required pam_tally.so [file=alternatepath]
```

This example shows the file modified to report failed login using an alternate location for the tally file.

```
#
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.). The default is to use the
```

```
# traditional Unix authentication mechanisms.
#
auth    required    pam_faillog.so file=/ufs/logs/tally.log
auth    required    pam_env.so
auth    required    pam_unix2.so
auth    required    pam_tally.so file=/ufs/logs/tally.log
```

- b. Edit the `/etc/pam.d/common-account` file and add the following line as the last entry:

```
boot:~ # vi /etc/pam.d/common-account

account required pam_tally.so [file=alternatepath]
```

This example shows the file modified to report failed login using an alternate location for the tally file.

```
#
# /etc/pam.d/common-account - authorization settings common to all
# services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authorization modules that define
# the central access policy for use on the system. The default is to
# only deny service to users whose accounts are expired.
#
account required    pam_unix2.so
account required    pam_tally.so file=/ufs/logs/tally.log
```

- c. Edit the `/etc/pam.d/common-session` file and add the following line as the last entry:

```
boot:~ # vi /etc/pam.d/common-session

session optional pam_faillog.so [file=alternatepath]
```

This example shows the file modified to report failed login using an alternate location for the tally file.

```
#
# /etc/pam.d/common-session - session-related modules common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of modules that define tasks to be performed
# at the start and end of sessions of *any* kind (both interactive and
# non-interactive). The default is pam_unix2.
#
session required    pam_limits.so
session required    pam_unix2.so
session optional    pam_umask.so
session optional    pam_faillog.so file=/ufs/logs/tally.log
```

2. Copy the edited files to the shared root by using `xtopview` in the default view.

```
boot:~ # cp -p /etc/pam.d/common-auth /rr/current/software
boot:~ # cp -p /etc/pam.d/common-account /rr/current/software
boot:~ # cp -p /etc/pam.d/common-session /rr/current/software
boot:~ # xtopview -m "configure login failure logging PAM"
default:/ # cp -p /software/common-auth /etc/pam.d/common-auth
default:/ # cp -p /software/common-account /etc/pam.d/common-account
default:/ # cp -p /software/common-session /etc/pam.d/common-session
```

3. Exit xtopview.

```
default/:// # exit
```

Check the Status of System Components

Check the status of the system or a component with the `xtcli status` command on the System Management Workstation (SMW). By default, the `xtcli status` command returns the status of nodes.

The `xtcli status` command has the following form:

```
xtcli status [-n] [-m] [{-t type -a}] node_list
```

Where *type* may be: `node`, `10`, `cage`, `11`, `xdp`, `verty`, `dim`, `socket`, `die`, `core`, `memctrl`, `gemini`, `nic`, `lcb`, `serdes_macro`, `fpga`, or `accel` (`accel` applies to Cray XK blades with GPUs only). The list must have component IDs only and contain no wild cards.

Use the `-m` option to display all nodes that were repurposed by using the `xtcli mark_node` command. (The `xtcli mark_node` command can be used to repurpose a service node to a compute role or to repurpose a compute node to a service role.)

For more information, see the `xtcli(8)` man page.

Show the status of a component

For this example, display all nodes that were repurposed using the `xtcli mark_node` command:

```
crayadm@smw:~> xtcli status -m c0-0c0
```

```
Network topology: class 1
```

```
Network type: Gemini
```

Nodeid	Service	Core	Arch	Comp	state	[Flags]
c0-0c0s2n0:	-	IB06	X86		off	[noflags]
c0-0c0s3n0:	service	IN32	X86		off	[noflags]

This shows that `c0-0c0s2n0` is a service node repurposed as a compute node, and that `c0-0c0s3n0` is a compute node repurposed as a service node.

Check the Status of Compute Processors

Use the `xtprocadmin` command on a service node to check that compute nodes are available after the system is booted.

Use the `xtprocadmin` command on a node to check that compute nodes are available after the system is booted.

Identify nodes in down or admindown state

```
nid00007:~> xtprocadmin | grep down
```

Use the user `xtnodestat` command to display the current allocation and status of each compute processing element and the application that it is running. A simplified text display shows each processing element on the Cray system interconnection network.

Display current allocation and status of each compute processing element and the application that it is running

```
nid00007:~> xtnodestat
Current Allocation Status at Wed Jul 06 13:53:26 2011
```

```

C0-0
n3 AAaaaaaa
n2 AAaaaaaa
n1 Aeeaaaa-
c2n0 Aeeaaaa
n3 Acaaaaa-
n2 cb-aaaa-
n1 AA-aaaa-
c1n0 Aadaaaa-
n3 SASaSa--
n2 SbSaSa--
n1 SaSaSa--
c0n0 SASaSa--
s01234567
```

Legend:

```

nonexistent node          S  service node
; free interactive compute node  -  free batch compute node
A allocated interactive or ccm node ?  suspect compute node
W waiting or non-running job      X  down compute node
Y down or admindown service node   Z  admindown compute node
```

```
Available compute nodes:          0 interactive,          15 batch
```

Job ID	User	Size	Age	State	command line	
a	3772974	user1	48	0h06m	run	app1
b	3773088	user2	2	0h01m	run	app2
c	3749113	user3	2	28h26m	run	app3
d	3773114	user4	1	0h00m	run	app4
e	3773112	user5	4	0h00m	run	app5

For more information, see the `xtprocadmin(8)` and `xtnodestat(1)` man pages.

Check CNL Compute Node Connection

Use the Linux `ping` command to verify that a compute node is connected to the network. The command must be run from a node, not the SMW.

For more information, see the Linux ping(8) man page.

Verify that a compute node is connected to the network

```
nid00007:~> ping nid00015
PING nid00015 (10.128.0.16) 56(84) bytes of data.
64 bytes from nid00015 (10.128.0.16): icmp_seq=1 ttl=64 time=0.032 ms
64 bytes from nid00015 (10.128.0.16): icmp_seq=2 ttl=64 time=0.010 ms
```

Check Link Control Block and Router Errors

The HSS `xtnetwatch` command monitors the Cray system interconnection network. It requests link control block (LCB) and router error information from the blade controller-based router daemons and specifies how often to sample for errors. It then detects events that contain the error information sent by these daemons and displays the information as formatted output in a log file. The system administrator can specify which system components to sample and control the level of verbosity of the output, select the sampling interval, and log results to an output file.

Although the command can be invoked standalone from the SMW prompt, Cray recommends running `xtnetwatch` each time the system is booted with the `xtbootsys` command (the default). The output is a time-stamped log file such as:

```
/var/opt/cray/log/p0-20120803t185511/netwatch.p0-20120803t185511
```

Check the log file for fatal link errors and router errors. Fatal link errors signal faulty hardware. Fatal router errors can be generated either by hardware or software; they do not cause the network or individual links to become inoperable but imply that a single transfer was discarded.

Use `xtnetwatch -d` to turn off blade controller high-speed interconnect link monitoring.

The L0 `netwatch` daemon (`gmnwd`) monitors routing tables for corruption. When a corrupted entry is detected, an `ec_hw_error` event is sent upwards with a special error code (`0x1501`) that is used by `xtnlrd` on the SMW to initiate a quiesce and re-route operation, which fixes the routing tables. `xtnetwatch` will have new log entries when it sees the `ec_hw_error` event with the new error code. This feature is disabled by setting the `xtdaemonconfig rtr_table_monitor` parameter to `false`.

This example shows a log entry after an `ec_hw_error` event has occurred:

```
130410 14:18:35 #####
130410 14:18:35 LCB ID      Peer LCB      Soft Errors   Fatal Errors
130410 14:18:35 #####
...
130410 17:23:45 c1-0c2s8g0                               Routing Table Corruption
```

Run xtnetwatch to monitor the system interconnection network

For this example, sample the network once every 10 seconds using the least verbose display format:

Running `xtnetwatch` to monitor the system interconnection network

```
crayadm@smw:~> xtnetwatch -i 10
120207 23:01:58 #####
120207 23:01:58 LCB ID      Peer LCB      Soft Errors   Fatal Errors
```

```

120207 23:01:58 #####
120207 23:01:59 c1-0c0s3g0130 c0-0c0s3g0130 1 TX lanemask=5
120207 23:02:27 c1-0c0s4g1103 c1-0c0s3g1103 1 Mode Exchanges
120207 23:02:27 c1-3c1s2g1103 c2-2c0s7g0153 1 Link Inactive

```

Display System Network Congestion Protection Information

Two utilities help to identify the time and duration of system network congestion events, either by parsing through logs (`xtcpreport`) or in real time (`xtcptop`):

- xtcpreport** This command uses information contained in the given `xtn1rd` file to extract and display information related to system network congestion protection. See the `xtcpreport(8)` man page for additional information.
- xtcptop** This command monitors an `xtn1rd` file that is currently being updated and displays real-time system network congestion protection information, including start time, duration, and `apid`. See the `xtcptop(8)` man page for additional information.

To use these utilities, load the `congestion-tools` module if it is not already loaded.

```
crayadm@smw:~> module load congestion-tools
```

Monitor the System with the System Environmental Data Collector (SEDC)

The System Environment Data Collections (SEDC) manager, `sedc_manager`, monitors the system's health and records the environmental data and status of hardware components such as power supplies, processors, temperature, and fans. SEDC can be set to run at all times or only when a client is listening. The SEDC configuration file provided by Cray has automatic data collection set as the default action.

The SEDC configuration file (`/opt/cray/hss/default/etc/sedc_srv.ini` by default) configures the SEDC server. In this file, the administrator can create sets of different configurations as groups so that the blade and cabinet controller daemons can scan components at different frequencies. The `sedc_manager` sends out the scanning configuration for specific groups to the cabinet and blade controllers and records the incoming data by group.

For information about configuring the SEDC manager, see *Using and Configuring System Environment Data Collections (SEDC) (S-2491)*.

HSN Network Metrics Monitoring

The Cray system includes compute node kernel modifications that provide Gemini network metrics monitoring and aggregation through third-party tools such as Sandia National Laboratories' OVIS https://ovis.ca.sandia.gov/mediawiki/index.php/Main_Page data collection and analysis system. Specifically, this functionality is provided with OVIS' Lightweight Distributed Metric Service (LDMS). The metrics are collected on a per-node and per-NIC basis using the Gemini network performance counters. It is therefore possible to examine metrics for each node without the use of OVIS. Metrics monitoring must be installed using the `CLEInstall.sh` utility. See *Installing and*

Configuring Cray Linux Environment (CLE) Software for information on how to install network metrics monitoring. In order to use OVIS/LDMS, you must install and configure OVIS per its installation instructions at <https://ovis.ca.sandia.gov/mediawiki/index.php/CRAY-LDMS> after running `CLEInstall.sh`.

Userspace programs are provided to facilitate metric collection and aggregation. These include:

gpcdr-init A program that provides metrics specified in `gpcdr-init.conf` to the `gpcdr` kernel module at boot time.

gpcdr-ctl A tool for managing metricsets and metrics. Administrators may add, modify, or remove metrics using this tool. For more information, see the `gpcdr-ctl(8)` man page.

The Gemini link-to-tile mapping is provided by either the output from the `rtr` command or by using the `rca-helper` command with the `-0` option. `gpcdr` is the name of the kernel module that provides metrics based on these mappings. In the case of OVIS, the daemon `ldms-gemctrs`, consumes data from `gpcdr` for each node. The LDMS system will then gather and aggregate these metrics for use within OVIS.

The configuration file, `gpcdr-init.conf`, specifies Gemini performance counters and the metrics to be collected. For metric descriptions with `PERDIM=1`, `gpcdr-init` modifies the target registers by prefixing `GM_n_m_TILE_` to the specified register name. For example, if `PERFORMANCE_COUNTERS_0` is specified, the corresponding register name used is `GM_n_m_TILE_PERFORMANCE_COUNTERS_0` for the corresponding values of `n` and `m`.

For detailed questions about tuning metrics for Gemini networks, contact a Cray representative.

Access Lustre File System Activity Data with the Lustre Monitoring Tool

The Lustre Monitoring Tool (LMT) is a distributed system that provides a top-like display of activity on server-side nodes (MDS, OSS and LNET routers) on Lustre file systems. It collects data using the Cerebro monitoring system and, if configured, stores it in a MySQL database. Graphical and text clients that display historical and real time data pulled from the database are provided.

View Data

Two commands display data provided by LMT:

- `ltop` displays live data
- `lmtsh` displays historical data from the MySQL database

These commands are only available on the LMT server, which for CLE with DAL is the MGS. For further information, see the `ltop(8)` or `lmtsh(8)` man pages; or view usage information using the `--help` option.

Aggregate Data

Configuration of the data aggregation cron job is handled through the IMPS configurator during CLE initial installation or upgrade. For further information, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*. For information regarding LMT and the Cray Linux File System (CLFS), see *Data Management Platform (DMP) Administrator's Guide (S-2327)*.

Remove Data

LMT does not provide an automated utility for removing old data from the MySQL database, but this can be done manually. MySQL commands can be used to clear data that is older than a certain date. These commands must be run on the MGS.

This example demonstrates how to clear all data for filesystem *fsname* that is older than October 4th at 15:00:00 from the `MDS_OPS_DATA` table using MySQL commands.

```
mysql -p -e "use filesystem_ fsname;  
delete MDS_OPS_DATA from  
MDS_OPS_DATA inner join TIMESTAMP_INFO  
on MDS_OPS_DATA.TS_ID=TIMESTAMP_INFO.TS_ID  
where TIMESTAMP < '2014-10-04 15:00:00';"
```

Additionally, `lmtsh` provides a way to remove all data from a table via the `clr` command. This example demonstrates how to remove all data from the table *TABLE_NAME*:

```
lmtsh> clr TABLE_NAME
```

This example demonstrates how to remove all data from all aggregated tables:

```
lmtsh> clr agg
```

See the `lmtsh(8)` man page for further information.

Examine Activity on the HSS Boot Manager

Use the HSS `xtcli session show` command to examine sessions in the boot manager. A session corresponds to running a specific command such as `xtcli power up` or `xtcli boot`. This command reports on sessions, not daemons.

For more information, see the `xtcli(8)` man page.

[View a session running on the boot manager](#)

```
crayadm@smw:~> xtcli session show BM
```

Poll a Response from an HSS Daemon, Manager, or the Event Router

Use the HSS `xtalive` command to verify that an HSS daemon, manager, or the event router is responsive.

For more information, see the `xtalive(8)` man page.

Check the boot manager

```
crayadm@smw:~> xtalive -l smw -a bm s0
```

Validate the Health of the HSS

The `xtcheckhss` command initiates a series of tests that validate the health of the HSS by gathering and displaying information supplied by scripts located on blade controllers (BCs) and cabinet controllers (CCs). `xtcheckhss` includes the following tests:

- **Version Checker:** Reads the current version running on the L0C, QLOC, L0Ds, BC micro, CC micro, CC FPGA, CHIA FPGAs, Tolapai BIOSes, and Node BIOS. The version that is read from each device is compared to the currently installed versions on the SMW.
- **Sensor Checker:** Reads environment sensors including temperatures, voltages, currents, and other data.
- **SEEP Checker:** Reads serial electrically erasable PROMs (SEEPs) in the system. This test can report any un-initialized, zeroed, or unreadable SEEPs.
- **AOC Checker:** Reads all active optical cable (AOC) data. This test displays any outliers relative to the average data calculated by previous runs.
- **ITP Checker:** Validates the embedded ITP path
- **NTP Checker:** Reads system time on all controllers and compares them with the SMW time; displays any mismatches.
- **Control Checker:** Examines and modifies system controls.
- **Configuration Information Checker:** Reads the system hardware configuration and reports the system setup, including the blade type, daughter card type, CPU type and count, and the CPU and PDC mask.
- **PCI checker:** Checks for missing or degraded PCIe connectivity on add-in cards on an IBB. This test requires that the nodes be powered up and bounced. Any cards that do not train to the PCIe Gen or Width specified in the Link Capability register are flagged. Any cards that are reported as physically present but not seen by the node are flagged.

For complete information, see the `xtcheckhss(8)` man page.

Monitor Event Router Daemon (erd) Events

The HSS `xtconsumer` command enables the system administrator to monitor events mediated by the event router daemon `erd`, which runs passively.

Monitor for specific events

For this example, watch two events: `ec_heartbeat_stop`, which will be sent if either the node stops sending heartbeats or if the system interconnection network ASIC stops sending heartbeats, and `ec_l0_health`, which will be sent if any of the subcomponents of a blade controller report a bad health indication:

```
crayadm@smw:~> xtconsumer -b ec_heartbeat_stop ec_10_health
```

Use the `xthb` command to confirm the stopped heartbeat. Use the `xthb` command only when actively looking into a known problem because it is intrusive and degrades system performance.

Check events except heartbeat

```
crayadm@smw:~> xtconsumer -x ec_11_heartbeat
```

For more information, see the `xtconsumer(8)` and `xthb(8)` man pages.

Monitor Node Console Messages

The `xtbootsys` command automatically initiates an `xtconsole` session, which displays the console text of a specified node(s) or accelerator(s). The `xtconsole` command operates in a shell window and monitors the event router daemon (`erd`) for console messages. The node or accelerator ID appears at the beginning of each line. The messages are written into `/var/opt/cray/log/sessionid/console-yyyymmdd` where the administrator may monitor them.

The `xtconsole` utility may only have one concurrent instance.

For more information, see the `xtconsole(8)` man page.

View Component Alert, Warning, and Location History

Use the `xtcli comp_hist` command to display component alert, warning, and location history. Either an error history, which displays alerts or warnings found on designated components, or a location history may be displayed.

Display the location history for component c0-0c0s0n1

```
crayadm@smw:~> xtcli comp_hist -o loc c0-0c0s0n1
```

For more information, see the `xtcli(8)` man page.

Display Component Information

Use the HSS `xtshow` command to identify compute and service components. Commands are typed as `xtshow --option_name`. Combine the `--service` or `--compute` option with other `xtshow` options to limit the selection to the specified type of node.

For a list of all `xtshow --option_name` options, see the `xtshow(8)` man page.

Identify all service nodes

```

crayadm@smw:~> xtshow --service
XDPs ...
L1s ...
Cages ...
L0s ...
    c0-0c0s0: service      X86|      ready      [noflags|]
    c0-0c0s2: service      X86|      ready      [noflags|]
    c1-0c0s4: service      X86|      ready      [noflags|]
    c1-0c0s6: service      X86|      ready      [noflags|]
Nodes ...
    c0-0c0s0n0: service    IB06 X86|  ready      [noflags|]
    c0-0c0s0n1: service    IB06 X86|  ready      [noflags|]
    c0-0c0s0n2: service    IB06 X86|  ready      [noflags|]
    c0-0c0s0n3: service    IB06 X86|  ready      [noflags|]
    c0-0c0s2n0: service    IB06 X86|  ready      [noflags|]
    c0-0c0s2n1: service    IB06 X86|  ready      [noflags|]
    c0-0c0s2n2: service    IB06 X86|  ready      [noflags|]
.
.
.
GPUs ...
Geminis ...
    c0-0c0s0g0: service    X86|      on         [noflags|]
    c0-0c0s0g0: service    X86|      on         [noflags|]
    c0-0c0s2g0: service    X86|      on         [noflags|]
    c0-0c0s2g0: service    X86|      on         [noflags|]
    c0-0c0s4g0: service    X86|      on         [noflags|]
    c0-0c1s4g0: service    X86|      on         [noflags|]
    c0-0c0s6g0: service    X86|      on         [noflags|]
    c0-0c1s6g0: service    X86|      on         [noflags|]
NICs ...
    c0-0c0s0g0n0: service  X86|      off        [noflags|]
    c0-0c0s0g0n1: service  X86|      off        [noflags|]
    c0-0c0s0gln0: service  X86|      off        [noflags|]
    c0-0c0s0gln1: service  X86|      off        [noflags|]
    c0-0c0s2g0n0: service  X86|      off        [noflags|]
    c0-0c0s2g0n1: service  X86|      off        [noflags|]
    c0-0c0s2gln0: service  X86|      off        [noflags|]
.
.
.
Serdes_macros
    c0-0c0s0g0m0: service  X86|      off        [noflags|]
    c0-0c0s0g0m1: service  X86|      off        [noflags|]
    c0-0c0s0g0m2: service  X86|      off        [noflags|]
    c0-0c0s0g0m3: service  X86|      off        [noflags|]
    c0-0c0s0g0m4: service  X86|      off        [noflags|]

```

Identify compute nodes in the disabled state

```

crayadm@smw:~> xtshow --compute --disabled
XDPs ...
L1s ...
Cages ...
L0s ...
Nodes ...

```

c2-0c1s2n1:	-	X86	disabled	[noflags]
c3-0c0s5n2:	-	X86	disabled	[noflags]
c3-0c1s5n0:	-	X86	disabled	[noflags]
c3-0c2s2n0:	-	X86	disabled	[noflags]
GPUs ...				
Geminis ...				
Nics ...				
Serdes_macros ...				
Sockets ...				
c2-0c1s2n1s0:	-	X86	disabled	[noflags]
c3-0c0s5n2s0:	-	X86	disabled	[noflags]
c3-0c1s5n0s0:	-	X86	disabled	[noflags]
c3-0c2s2n0s0:	-	X86	disabled	[noflags]
Dies ...				
c2-0c1s2n1s0d0:	-	X86	disabled	[noflags]
c3-0c0s5n2s0d0:	-	X86	disabled	[noflags]
c3-0c1s5n0s0d0:	-	X86	disabled	[noflags]
c3-0c2s2n0s0d0:	-	X86	disabled	[noflags]
Cores ...				
c2-0c1s2n1s0d0c0:	-	X86	disabled	[noflags]
c3-0c0s5n2s0d0c0:	-	X86	disabled	[noflags]
c3-0c1s5n0s0d0c0:	-	X86	disabled	[noflags]
c3-0c2s2n0s0d0c0:	-	X86	disabled	[noflags]
Memctrls ...				
c2-0c1s2n1s0d0m0:	-	X86	disabled	[noflags]
c3-0c0s5n2s0d0m0:	-	X86	disabled	[noflags]
c3-0c1s5n0s0d0m0:	-	X86	disabled	[noflags]
c3-0c2s2n0s0d0m0:	-	X86	disabled	[noflags]
Dimms ...				
c2-0c1s2n1d0:	-	X86	disabled	[noflags]
c2-0c1s2n1d1:	-	X86	disabled	[noflags]
c2-0c1s2n1d2:	-	X86	disabled	[noflags]

Identify components with a status of not empty

```
crayadm@smw:~> xtshow --not_empty c0-0c0s0
```

XDPs ...				
L1s ...				
c0-0:	-		ready	[noflags]
Cages ...				
L0s ...				
c0-0c0s0: service		X86	ready	[noflags]
Nodes ...				
c0-0c0s0n0: service IB06		X86	on	[noflags]
c0-0c0s0n1: service IB06		X86	on	[noflags]
c0-0c0s0n2: service IB06		X86	on	[noflags]
c0-0c0s0n3: service IB06		X86	on	[noflags]
GPUs ...				
Geminis ...				
c0-0c0s0g0: service		X86	on	[noflags]
c0-0c0s0g0: service		X86	on	[noflags]
Nics ...				
c0-0c0s0g0n0: service		X86	off	[noflags]
c0-0c0s0g0n1: service		X86	off	[noflags]
c0-0c0s0g1n0: service		X86	off	[noflags]
c0-0c0s0g1n1: service		X86	off	[noflags]
Serdes_macros ...				
c0-0c0s0g0m0: service		X86	off	[noflags]
c0-0c0s0g0m1: service		X86	off	[noflags]

c0-0c0s0g0m2:	service	X86	off	[noflags]
c0-0c0s0g0m4:	service	X86	off	[noflags]
.				
.				
.				
Sockets ...				
c0-0c0s0n0s0:	service	X86	on	[noflags]
c0-0c0s0n1s0:	service	X86	on	[noflags]
c0-0c0s0n2s0:	service	X86	on	[noflags]
c0-0c0s0n3s0:	service	X86	on	[noflags]
Dies ...				
c0-0c0s0n0s0d0:	service	X86	on	[noflags]
c0-0c0s0n1s0d0:	service	X86	on	[noflags]
c0-0c0s0n2s0d0:	service	X86	on	[noflags]
c0-0c0s0n3s0d0:	service	X86	on	[noflags]
Cores ...				
c0-0c0s0n0s0d0c0:	service	X86	on	[noflags]
c0-0c0s0n0s0d0c1:	service	X86	on	[noflags]
c0-0c0s0n0s0d0c2:	service	X86	on	[noflags]
c0-0c0s0n0s0d0c3:	service	X86	on	[noflags]
.				
.				
.				
Memctrls ...				
c0-0c0s0n0s0d0m0:	-	X86	on	[noflags]
c0-0c0s0n1s0d0m0:	-	X86	on	[noflags]
c0-0c0s0n2s0d0m0:	-	X86	on	[noflags]
c0-0c0s0n3s0d0m0:	-	X86	on	[noflags]
Dimms ...				
c0-0c0s0n0d0:	-	X86	on	[noflags]
c0-0c0s0n0d1:	-	X86	on	[noflags]
c0-0c0s0n0d2:	-	X86	on	[noflags]
.				
.				
.				
Vertys ...				
c0-0c0s0v0:	service	X86	off	[noflags]
c0-0c0s0v1:	service	X86	off	[noflags]
c0-0c0s0v2:	service	X86	off	[noflags]
c0-0c0s0v3:	service	X86	off	[noflags]
c0-0c0s0v4:	-	X86	off	[noflags]
c0-0c0s0v5:	-	X86	off	[noflags]
FPGAs ...				
c0-0c0s0f0:	-	X86	off	[noflags]
c0-0c0s0f1:	-	X86	off	[noflags]
Lcbs ...				
c0-0c0s0g0l00:	service	X86	off	[noflags]
c0-0c0s0g0l01:	service	X86	off	[noflags]
c0-0c0s0g0l02:	service	X86	off	[noflags]
c0-0c0s0g0l03:	service	X86	off	[noflags]
.				
.				
.				

Display Alerts and Warnings

Use the `xtshow` command to display alerts and warnings. Type commands as `xtshow --option_name`, where *option_name* is `alert`, `warn`, or `noflags`.

Alerts are not propagated through the system hierarchy, only information for the component being examined is displayed. For example, invoking the `xtshow --alert` command for a cabinet does not display an alert for a node. Similarly, checking the status of a node does not detect an alert on a cabinet.

Show all alerts on the system

```
crayadm@smw:~> xtshow --alert
```

Alerts and warnings typically occur while the HSS `xtcli` command operates; these alerts and warnings are listed in the command output with an error message. After they are generated, alerts and warnings become part of the state for the component and remain set until manually cleared.

For example, the temporary loss of a heartbeat by the blade controller may set a warning state on a chip.

For additional information, see the `xtshow(8)` man page.

Clear Component Flags

Use the `xtclear` command to clear system information for selected components. Type commands as `xtclear --option_name`, where *option_name* is `alert`, `reserve`, or `warn`.

Clear all warnings in specified cabinet

For this example, clear all warnings in cabinet `c13-2`:

```
smw:~> xtclear --warn c13-2
```

Alerts, reserves, and warnings must be cleared before a component can operate. Clearing an alert on a component frees its state so that subsequent commands can execute [System Component States](#) on page 340.

For more information, see the `xtclear(8)` man page.

Display Error Codes

When an HSS event error occurs, the related message is displayed on the SMW. The `xterrorcode` command on the SMW displays a single error code or the entire list of error codes.

Display HSS error codes

```
crayadm@smw:~> xterrorcode errorcode
```

A system error code entered in a log file is a bit mask; invoking the `xterrorcode bitmask_code_number` command on the SMW displays the associated error code.

Display an HSS error code using its bit mask number

```
crayadm@smw:~> xterrorcode 131279
Maximum error code (RS_NUM_ERR_CODE) is 447
code = 207, string = 'Node Voltage Fault'
```

Modify an Installed System

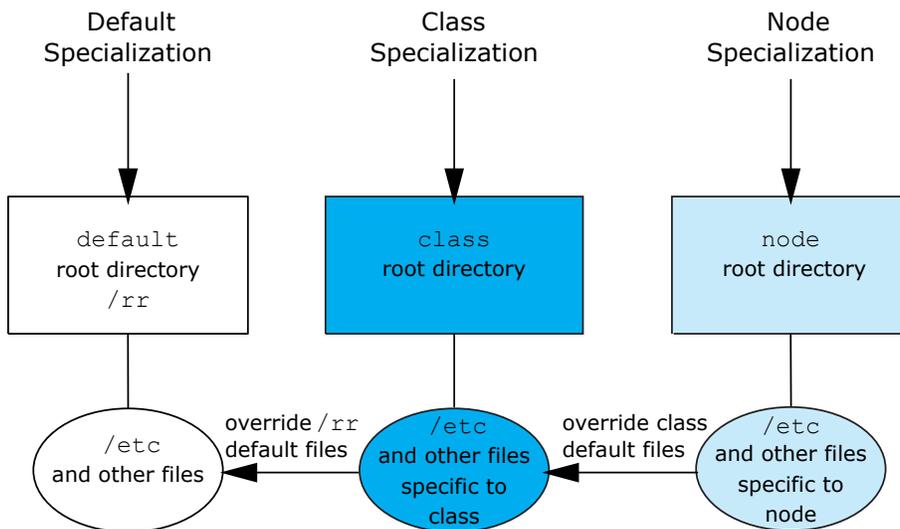
About the Shared-root File System on Service Nodes

CLE implements a shared-root file system where `/` is exported from the boot node and is mounted as read-only on all service nodes. To overcome the restriction that all nodes must have the same shared-root file system, `/etc` directories can be symbolic links to unique directories that have the same structure as the default `/etc` directory but contain modified files. These node-specific files reside in subdirectories in the `/.shared/base` directory.

Specialization is the process of changing the link to a file in the `/etc` directory to point to a unique file for one, a few, or all nodes. The administrator can specialize one or more files for an individual node or for a class (type) of nodes, such as `login`. Configuring the shared-root file system in this manner requires `root` privileges. Files are specialized either when the system is install or at a later time.

The hierarchical structure of the specialized files is shown below. Node specialization is more specific than class specialization. Class specialization is more specific than default specialization. Generally, about 98% of what the service nodes use is the default version of the shared root.

Figure 2. Types of Specialization



File Specialization

Files are specialized when there is a need to point to a unique version of a file in the `/etc` directory rather than to the standard version of the file that is shared on all nodes. For example, an administrator might specialize files

when differences exist in hardware, network configuration, or boot scripts or when there are services that run on a single node. Files can also be specialized for a class of nodes that have a particular function, such as login.

Generally, files are specialized as part of the installation process, but the process can be done at any time. It is good practice to enter the `xtopview` shell and then specialize the files.

The following tables lists files and directories that can be specialized by class, and files and directories that can be specialized by node, respectively, and the reasons to do so. In these tables, `*` is a "wildcard" character that represents no characters or any number of characters.

Table 2. File Specialization by Class

File or Directory	Reason for Specialization
<code>/etc/cron*</code>	Different classes need custom crontabs
<code>/etc/fstab</code>	I/O nodes need to mount other file systems
<code>/etc/hosts.{allow,deny}</code>	Must restrict logins on login nodes
<code>/etc/init.d/boot.d/*</code>	Different classes have different start-up scripts enabled
<code>/etc/init.d/rc*/</code>	Different classes have different start-up scripts enabled
<code>/etc/issue</code>	Different classes have different messages
<code>/etc/modprobe.conf</code>	I/O and login nodes have different hardware
<code>/etc/motd</code>	Different classes have different messages
<code>/etc/pam*</code>	Authentication is class-specific
<code>/etc/profile.d/*</code>	Login nodes have custom environments
<code>/etc/resolv.conf</code>	Hosts that interact with external servers need special resolver configurations
<code>/etc/security/*</code>	Authorization and system limits are class-specific
<code>/etc/sysconfig/network/*</code>	I/O and login nodes need custom network configuration

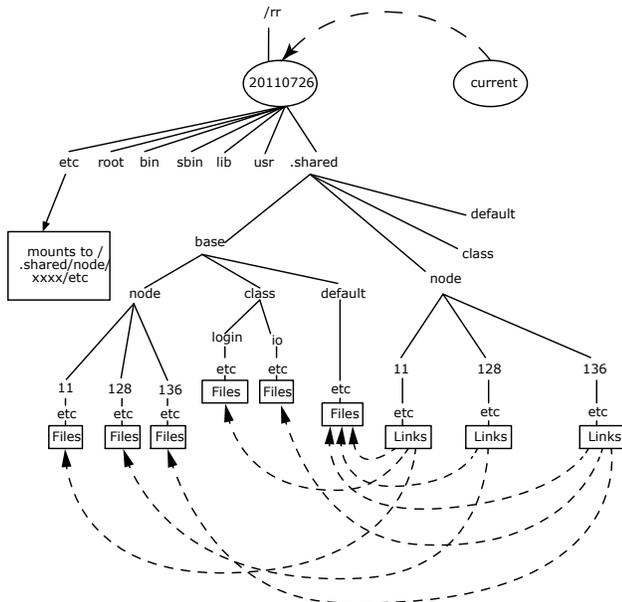
Table 3. File Specialization by Node

File or Directory	Reason for Specialization
<code>/etc/cron*</code>	Certain service nodes, such as <code>sdb</code> and <code>syslog</code> , need custom crontabs
<code>/etc/ntp.conf</code>	A node that runs an NTP server needs a different configuration than NTP clients
<code>/etc/sysconfig/network/*</code>	Each network node should have a different IP address
<code>/etc/syslog-ng/syslog-ng.conf.in</code>	A node that runs a syslog server needs a different configuration than syslog clients
<code>/etc/ssh/*key*</code>	Use when sharing keys across systems is unacceptable

Visible Shared-root File System Layout

Shared-root Implementation is a detailed illustration of shared-root directory structure. The directory current is a subdirectory of /rr. The current directory links to a time-stamped directory (in this example 20110726). The timestamp indicates the date of the software installation, not the date of the release.

Figure 3. Shared-root Implementation



Service nodes mount the /rr/current directory from the boot node as read-only for use as their root file system. The visible file layout, that is, how it appears from the node on which it is viewed, contains the following files:

- / Root file system
- /root Equivalent to directory of the same name in /rr/current
- /bin Equivalent to directory of the same name in /rr/current
- /lib Equivalent to directory of the same name in /rr/current
- /sbin Equivalent to directory of the same name in /rr/current
- /usr Equivalent to directory of the same name in /rr/current
- /opt Equivalent to directory of the same name in /rr/current
- /etc Contains links to the shared-root files
- /home Link to /home, a customer-specific location
- /tmp Implemented through the tmpfs (in RAM)
- /var Directory in the tmpfs and RAMFS but populated with skeleton files if there is no have persistent /var
- /proc Per-node pseudo-file system
- /dev Per-node pseudo-file system implemented through the DEVFS
- /ufs Mount point for the /ufs file system to be mounted from the ufs node

How Specialization Is Implemented

The shared-root file system is implemented in the `/.shared` directory. Only the `/etc` directory has been set up for specialization. Files in `/etc` are symbolic links to files in `/.shared/base`. A specialized file is a unique version of the file in the `/.shared/base` directory.

The `/.shared` directory contains four subdirectories: `base`, `node`, `class`, and `default`. The `node`, `class`, and `default` directories are also known as *view directories*, because the file system can be looked at (with the `xtopview` command) as if the view directory were `/`.

The `base` subdirectory also contains subdirectories called `node`, `class`, and `default`. These are referred to as *base directories*. They contain files that are specific to a certain node, specific to a class of nodes, or shared as the default among all nodes. Under each of the base directories is a rooted directory hierarchy where files are stored.

The path of the link shows the type of specialization for the file.

Share-root links

Default specialization:

```
default/: # ls -la /etc/hosts
lrwxrwxrwx 1 root root 31 Dec 8 17:12 /etc/hosts -> /.shared/base/default/etc/hosts
```

Class specialization:

```
class/login/: # ls -la /etc/security/access.conf
lrwxrwxrwx 1 root root 46 Dec 8 17:14 /etc/security/access.conf -> \
/.shared/base/class/login/etc/security/access.conf
```

Node specialization:

```
node/128/: # ls -la /etc/resolv.conf
lrwxrwxrwx 1 root root 36 Dec 8 17:15 /etc/resolv.conf -> \
/.shared/base/node/128/etc/resolv.conf
```

Shared-root Activity Log

All specialization activity is logged in the log file `/.shared/log`, which tracks additions, deletions, and modifications of files. To view the details of changes made, the RCS logs that were created during the `xtopview` session must be accessed.

If `xtopview` is exited with **Ctrl-c**, the operations performed within the shell are not logged. The changes to the system are present nonetheless. This means that to back out of changes, it is not sufficient to exit `xtopview`; commands must be submitted to undo the changes made.

Control and Monitor the Shared-root File System

This table briefly describes the commands to control and monitor the shared-root file system. Further information is contained in the following sections and the respective man pages.

Table 4. Shared-root Commands

Command	Function
<code>xtopview</code>	View file layout from the specified node
<code>xtopcommit</code>	Record file specialization before leaving <code>xtopview</code> shell

Command	Function
<code>xtspec</code>	Specialize; create a directory structure that links files to non-default files
<code>xthowspec</code>	Determine the type of specialization
<code>xtverifyshroot</code>	Verify that node-specialized and class-specialized files are linked correctly
<code>xtverifyconfig</code>	Verify that start/stop links generated by tools such as <code>chkconfig</code> are consistent across all views of the shared root. The system administrator can configure <code>xtopview</code> to invoke <code>xtverifyconfig</code> automatically; this is the preferred usage. <code>xtverifyconfig</code> is not intended for direct use.
<code>xtverifydefaults</code>	Verify and fix inconsistent system default links within the shared root. The system administrator can configure <code>xtopview</code> to invoke <code>xtverifydefaults</code> automatically; this is the preferred usage. <code>xtverifydefaults</code> is not intended for direct use.
<code>xtcloneshared</code>	Create a directory structure for a new node or class based on an existing node or class
<code>xtnce</code>	Modify the class of a node or display the current class of a node
<code>xtunspec</code>	Remove specialization
<code>xtoprlog</code>	Display RCS log information for shared root files
<code>xtopco</code>	Check out (restore) RCS versioned shared root files
<code>xtoprdump</code>	Print a list of file specifications that can be used as the list of files to operate on an archive of shared root file system files
<code>xtoparchive</code>	Perform operations on an archive of shared root configuration files

Manage the System Configuration with the `xtopview` Tool

The `xtopview` tool manages the files in the shared-root file system. The administrator specifies the view of the system desired, such as from a particular node, when invoking the command. The system appears as if it were logged in directly to the specified location; that is, the files that are specialized for that node appear in the `/etc` directory. Location is specified by node ID or hostname.

Changes made within `xtopview` are logged to a revision control system (RCS) file. When exiting the shell, the administrator is prompted to type a message about each change made. Use the `c` command to comment work done in `xtopview`.

TIP: Use the `-m msg` option when starting an `xtopview` session to make similar changes to multiple files.

The changed files and messages are then logged to create a history that is stored in the `/.shared/base` directory by its specialization (node, class, or default) and file name. For example, changes and messages relating to default-specialized file `/etc/spk` are stored in `/.shared/base/default/etc/RCS`. Use standard RCS tools, such as `rlog`, for retrieving information.



WARNING: There is no automatic way to back out of changes made in an `xtopview` session. The individual commands necessary to revert the changes must be invoked.

Cray recommends configuring the shared root from within the `xtopview` shell. Only operations that take place within the `xtopview` shell are logged. Specialization commands used outside of `xtopview` are not logged. Logs reside in the `/rr/current/.shared/log` path relative to the boot node.

New files that are created from with the `xtopview` shell automatically have the specialization that is associated with the view under which they are created; they do not need to be specialized manually. For a file to be used by all service nodes, create the file in the default view.

The `xtopview` command is typically executed on the boot node; but an administrator can perform `xtopview` work from the SMW if the system is not booted, e.g., the system is undergoing hardware maintenance.

NOTE: If using the `emacs` editor within the `xtopview` shell, the following message may be displayed:

```
Symbolic link to RCS-controlled source file; follow link [yes or no]?
```

The symbolic link points to a real file in the `/.shared` directory. Choosing `yes` edits the file directly. Choosing `no` replaces the symbolic link with a real file, but upon exiting the `xtopview` shell, the file is moved to the correct location and the link is recreated. The difference is that if editing the real file, modifications appear immediately in other views.

Start the `xtopview` shell for a node

For this example, start the `xtopview` shell for node 131:

```
boot:~ # xtopview -n 131
node/131/: #
```

Start the `xtopview` shell for a class of nodes

For this example, start the `xtopview` shell for the login nodes:

```
boot:~ # xtopview -c login
class/login/: #
```

Start the `xtopview` shell for a directory other than `/rr/current`

```
boot:~ # xtopview -r /rr/20120901
default/:// #
```

Sample `xtopview` session

```
boot:~ # xtopview -n 3
node/3:/ # vi etc/fstab
. . . (edited the file)
node/3:/ # exit
exit
***File /etc/fstab was MODIFIED
operation on file /etc/fstab? (h for help):c
enter description, terminated with single '.' or end of file:
>changed the fstab file to add support for xyz.
boot:~ #
```

Generally, the `xtopview` command obtains node and class information from the SDB. If the SDB is not running, the administrator can direct `xtopview` to access the `/etc/opt/cray/sdb/node_classes` file by selecting the `-x` option.

Start xtopview using node_classes for information

For nodes:

```
boot:~ # xtopview -x /etc/opt/cray/sdb/node_classes -n 4
```

For classes:

```
boot:~ # xtopview -x /etc/opt/cray/sdb/node_classes -c login
```

Run xtopview from the SMW while the System is not Booted

Use xtopview from the SMW to perform software work while the system is not booted. This can be quite useful during hardware maintenance periods.

For more information, see the xtopview(8) man page.

1. Log on to the SMW as `root`.

2. Verify that the boot node is down.

```
smw:~ # ping boot
```

3. Mount the boot root and shared root file systems (if they are not already mounted). This example uses the default `bootroot_dir` and the shared root mount point is `/rr`.

```
smw:~ # mount /dev/disk/by-id/system_bootroot_ID /bootroot0
smw:~ # mount /dev/disk/by-id/system_shareroot_ID /bootroot0
```

4. Start a default view xtopview session. Additional options like `-c class` or `-n nid` can be used here as well.

```
smw:~ # chroot /bootroot0
smw:/ # xtopview -x /etc/opt/cray/sdb/node_classes -r /rr/current
```

5. Make changes and exit from the xtopview and chroot sessions when finished.

```
default/::~ # exit
smw:/ # exit
```

6. Unmount the boot root and shared root file systems.

```
smw:~ # umount /bootroot0/rr
smw:~ # umount /bootroot0
```

7. Verify that the boot root and shared root file systems are not mounted.

```
smw:~ # mount
smw:~ # df
```

Update Specialized Files From Within the xtopview Shell

Changes made within the `xtopview` shell are propagated to the shared-root file system when `xtopview` is exited. Use the `xtopcommit` command to immediately update the shared root with modifications. This can be done within the `xtopview` shell.

Update a file within xtopview shell

```
boot:~ # xtopview -n 3
node/3:/ # vi /etc/fstab
node/3:/ # xtopcommit
***File /etc/fstab was MODIFIED
operation on file /etc/fstab? (h for help):h
c:check-in - record changes in RCS file
d:diff - diff between file and backup RCS file
h:help - print this help message
m:message - set message for later checkins
M:nomsg - clear previously set message
l:list - list file info (ls -l)
s:skip - check-in file with empty log message
q:quit - check-in ALL files without querying
```

How to Specialize Files

Specifying a view with the `xtopview` command does not automatically specialize existing files. To specialize existing files, use the specialization command `xtspec`. The command runs on the boot node and creates a copy of a file that is unique to a node or class. The `xtspec` command has the form:

```
xtspec [options] file
```

The command specializes the file at the location *file* and updates each node or class of nodes that contains the newly specialized file if the new file is the most specialized file in its view. For example, if a file is specialized by class `io`, for all nodes with class `io` the symbolic links associated with this file are updated to point to the new file unless they are already specialized by `node`, which is a more restrictive class.

To specialize a file when not within `xtopview`, specify the path of the shared root with the `-r` option. In addition, the RCS log of changes has a generic entry for each file.

The `xtspec` command can be used only to specify files or directories residing in or under the `/etc` directory. If attempting to specify a file or directory outside of the `/etc` directory, the command fails and an error message is generated.

The `-V` option of the `xtspec` command specifies the location from which the file that is to be the specialized file is copied. If the `-V` option is specified, the newly specialized file is a duplicate of the file from the target's view. If the `-V` option is not specified, the newly specialized file is a duplicate of the file from the default view.

If a file is not specialized, the default specialization level is based on the current view if running in the `xtopview` shell or on the default view if operating outside the `xtopview` shell.

Classes are defined in the `node_classes` file.

Specialize a File by Class login

1. Start the `xtopview` shell for the login nodes.

```
boot:~ # xtopview -c login
```

2. Specialize the selected file.
For this example, specialize the file `/etc/dhcpd.conf`:

```
class/login:~ # xtspec /etc/dhcpd.conf
```

3. Edit `/etc/dhcpd.conf` if it is the default copy of the file. If a unique copy of the file was pointed to in the `xtspec` command, omit this step.
As a result of this procedure, each node in the class `login` links to the "new" `/etc/dhcpd.conf` file unless the node is already specialized by node. For example, node 23 might already be specialized and link to a different `/etc/dhcpd.conf` file.

Specialize a File by Node

1. Start the `xtopview` shell for node `nid`.
For this example, use node 11:

```
boot:~ # xtopview -n 11
```

2. Specialize the selected file.
For this example, specialize the file `/etc/motd`:

```
node/11/: # xtspec /etc/motd
```

This procedure creates a node-specific copy of `/etc/motd`. That is, the directory entry in the `/etc` file associated with node 11 is updated to point to the new version of `/etc/motd` and the activity is logged.

Specialize a File by Node Without Entering xtopview

Specify the root path and view mode.

For this example, the root path is `/rr/current` and the view is from node 11:

```
boot:~ # xtspec -r /rr/current -V -n 11 /etc/motd
```

As a result of this procedure, the directory entry in the `/etc` file associated with node 11 is updated to point to the new version of `/etc/motd` but the activity is not logged.

Specialized nodes can be unspecialized using the `xtunspec` command. To determine how they are specialized, use the `xthowspec` command. To view or change the class type of a particular node, use the `xtnce` command.

Specialization commands are only invoked from the boot node by `root` user. For more information, see the `shared_root(5)`, `xtspec(8)`, `xtunspec(8)`, `xthowspec(8)`, and `xtnce(8)` man pages.

Determine Which Files Are Specialized

The CLE `xthowspec` command displays how the files in a specified path are specialized. For example, `xthowspec` can be used to examine restrictions on login nodes.

The `xthowspec` command has the following form:

```
xthowspec [options] path
```

An administrator can display file specialization for all nodes or all classes, for a particular node or class, for the default view, or for a selection of parameters. Inside the `xtopview` shell, the `xthowspec` command acts on files in the current view by default.

Output has the form `TYPE.ITEM.FILE.SPEC`, where the fields are as follows:

TYPE Node, class or default
ITEM The specific node or class type (this field is empty for the default view)
FILE The file upon which the command is acting
SPEC The specialization level of the file in the view

To examine specialization outside the `xtopview` shell, the full path name must be entered.

For more information, see the `xthowspec(8)` man page.

Find files in /etc that are specialized by node

Inside the `xtopview` shell:

```
boot:~ # xtopview -n 4
node/4/: # xthowspec -t node /etc
node:4:/etc/fstab.h:node
node:4:/etc/hostname:node
```

Outside the `xtopview` shell:

```
boot:~ # xthowspec -r /rr/current -t node -n 4 /etc
node:4:/etc/fstab.h:node
node:4:/etc/hostname:node
```

Find files in /etc that are specialized by class

```
class/login:~ # xthowspec -r /rr/current -t class /etc
node:4:/etc/init.d/rc3.d/K01pbs:class
node:4:/etc/init.d/rc3.d/S11pbs:class
node:16:/etc/init.d/rc3.d/K01pbs:class
node:16:/etc/init.d/rc3.d/S11pbs:class
class/login:/etc/HOSTNAME:class
class/login:/etc/sysconfig/network/routes:class
...
```

Find the specialization of a file on node

```
boot:~ # xtopview -n 4
node/4/: # xthowspec /etc/dhcpd.conf
node:4:/etc/dhcpd.conf:default
```

Find the nodes on which a file is specialized

```
boot:~ # xthowspec -r /rr/current -N /etc/fstab
node:0:/etc/fstab:default
node:1:/etc/fstab:default
node:8:/etc/fstab:class
node:9:/etc/fstab:node
```

Find the specialization of a file on a node without invoking the xtopview shell

```
boot:~ # xthowspec -r /rr/current -n 102 /etc/fstab
node:102:/etc/fstab:node
```

Find specialization of files by class without invoking the xtopview shell

```
boot:~ # xthowspec -r /rr/current -N -t class /etc
node:11:/etc/crontab:class
node:1:/etc/crontab:class
```

Check the Shared-root Configuration

To check the configuration of the shared-root file system use the `xtverifyshroot` command in the `xtopview` shell. `xtverifyshroot` has the following format:

```
xtverifyshroot [options] path
```

If there are node-specialized or class-specialized files, the command verifies that they are linked correctly. If a problem is detected with a file, it is reported but not corrected.

For more information, see the `xtverifyshroot(8)` man page.

Verify the Coherency of /etc/init.d Files Across all Shared Root Views

The `xtopview` command is configured to invoke the `xtverifyconfig` utility automatically to resolve potential inconsistencies in the mechanism used to configure various CLE software services on or off. This is the preferred usage; the `xtverifyconfig` utility is not intended for direct use.

When the `chkconfig` utility is used to configure services on or off, a collection of encoded symbolic links are generated to determine which system services are started or shut down and in what order. The `chkconfig` utility does not account for the multiple levels of specialization within the shared root when `xtopview` is used. As a result, `chkconfig` occasionally produces a startup or shutdown order that violates dependencies between services

when all levels of specialization are taken into account. To resolve this problem, the system administrator can configure `xtopview` to invoke the `xtverifyconfig` verification utility upon exit. The `xtverifyconfig` utility will detect inconsistencies and may rename startup and shutdown links to maintain the proper dependency ordering. The `/.shared/log` log file in the shared root contains a log of modifications `xtverifyconfig` makes to the shared root.

The `xtopview` command will run `xtverifyconfig` upon exit if the `XTOPVIEW_VERIFY_INITD` environment variable is non-zero when `xtopview` is invoked, or if the `XTOPVIEW_VERIFY_INITD` variable is set to non-zero in the `/etc/sysconfig/xt` file on the boot node. By default, this parameter is not included in the configuration file and this feature is not enabled.

For more information, see the `xtverifyconfig(8)` man page.

Clone a Shared-root Hierarchy

To create a directory structure for a new node or class name in the shared-root hierarchy based on an existing node or class, use the `xtcloneshared` command. For more information, see the `xtcloneshared(8)` man page.

Change the Class of a Node

If nodes are removed, the class of the remaining nodes may need to change. If adding a login node, it must be added to class `login`. The `xtnce` command displays the current class of a node or modifies its class. The command has the form:

```
xtnce [options] nodename
```

Find the class of a node

```
boot:~ # xtnce 750
750:snx-lnet
```

Add a node to a class

```
boot:~ # xtopview -x /etc/opt/cray/sdb/node_classes
default:/ # xtnce -c login 104
```

The `/etc/opt/cray/sdb/node_classes` file on the boot node must be modified so the data is preserved across a boot; this is because the `node_classes` file is used to initialize the SDB data on the next boot, and the boot node file cannot be updated from within `xtopview`.

IMPORTANT: When making changes to `/etc/opt/cray/sdb/node_classes`, the same changes must be made to the node class settings in `CLEinstall.conf` before performing an update or upgrade installation; otherwise, the install utility will complain about the inconsistency.

For more information, see the `xtnce(8)` man page.

The `xtnodeclasses2db` command inserts the node-class list into the database, but it does not make any changes to the shared root.

Remove Specialization

Specialization is removed with the `xtunspec` command, which only runs on the boot node.

```
xtunspec [options] path
```

The system administrator can unspecialize files for all nodes and classes (default), for a specified class of nodes or for a particular node. Cray strongly recommends that this is done from within the `xtopview` shell. If not, the path for the shared root must also be specified.

For more information, see the `xtunspec(8)` man page.

Remove node specialization

For this example, remove all versions of `/etc/fstab` specialized by node:

```
boot:~ # xtopview  
default/:// # xtunspec -N /etc/fstab
```

Each node is updated so that it uses a version of `/etc/fstab` based on its class, or if that is not available, based on the default version of `/etc/fstab`.

Remove class specialization

For this example, remove all versions of `/etc/fstab` that are specialized by class `io`:

```
boot:~ # xtopview  
default/:// # xtunspec -c io /etc/fstab
```

I/O nodes that link to the class-specialized version of the file are changed to link to the default version of `/etc/fstab`. However, I/O nodes that already link to node-specialized versions of `/etc/fstab` are unchanged.

Display RCS Log Information for Shared Root Files

The `xtoprlog` command displays Revision Control System (RCS) log information for shared root files. Specify the file name using the required `filename` command-line argument. The `xtoprlog` command can be executed from within an `xtopview` shell or from the boot node as `root`. If `xtoprlog` is executed from within `xtopview`, it will operate on the current view; if the command is executed outside of `xtopview` on the boot node, then specify a view to use with the `-d`, `-c`, or `-n` options and also the `xtopview` root location with the `--root` option.

The scope of this tool is limited to identification and manipulation of `/etc` configuration data within the shared root. Configuration files on the boot root file system or on the SMW are not managed by this utility.

For more information, see the `xtoprlog(8)` man page.

Print the latest version (revision) number of a file

```
default/:// # xtoprlog --version /etc/fstab  
1.1
```

Print the RCS log for /etc/fstab in the node 3 view

```
default:/ # xtoprlog -n 3 /etc/fstab
RCS file: /.shared/base/node/3/etc/RCS/fstab,v
Working file: /.shared/base/node/3/etc/fstab
head: 1.6
...
```

Display differences between two versions of /etc/fstab

```
default:/ # xtoprlog -x -r 1.3 /etc/fstab
=====
RCS file: /.shared/base/default/etc/RCS/fstab,v
retrieving revision 1.3
diff -r1.3 /.shared/base/default/etc/fstab
1,3c1,4
< # Default view fstab file 1.3
---> # Default view fstab file 1.7
```

Check Out an RCS Version of Shared Root Files

Use the `xtopco` command to check out a version of shared root files. The `xtopco` command should be run on the boot node using the `xtopview` utility in the default view.

The scope of this tool is limited to identification and manipulation of `/etc` configuration data within the shared root. Configuration files on the boot root file system or on the SMW are not managed by this utility.

For more information, see the `xtopco(8)` man page.

Check out a version 1.2 copy of /etc/fstab

```
boot:~ # xtopview
default:/ # xtopco -r 1.2 /etc/fstab
```

Recreate the file link for /etc/fstab to the current view's /etc/fstab file

```
boot:~ # xtopview
default:/ # xtopco --link /etc/fstab
```

List Shared Root File Specification and Version Information

Using RCS information, combined with the `xtopview` specialization information, `xtoprdump` prints a list of file specifications that can be used as the list of files to operate on an archive of shared root file system files. The `xtoprdump` command should be invoked using the `xtopview` utility unless the `--root` option is specified.

The scope of this tool is limited to identification and manipulation of `/etc` configuration data within the shared root. Configuration files on the boot root file system or on the SMW are not managed by this utility.

For more information, see the `xtoprdump(8)` man page.

Print specifications for login class specialized files

```

boot:~ # xtopview
default:// # xtoprdump -c login
class:login:/etc/HOSTNAME:1.2:*
class:login:/etc/fstab:1.2:*
class:login:/etc/fstab.old:1.1:*
class:login:/etc/modprobe.conf.local:1.2:*
class:login:/etc/opt/cray/modules/Base-opts.default.local:1.2:*
class:login:/etc/sysconfig/network/config:1.2:*
class:login:/etc/sysconfig/network/routes:1.1:*
class:login:/etc/yp.conf:1.1:*

```

Print specifications for all node specialized files

```

boot:~ # xtopview
default:// # xtoprdump -n all
node:22:/etc/sysconfig/network/ifcfg-ib0:1.1:*
node:23:/etc/sysconfig/network/ifcfg-ib0:1.1:*
node:30:/etc/new_file:1.1:*
node:30:/etc/opt/cray/rsipd/rsipd.conf:1.1:*
node:30:/etc/sysconfig/network/ifcfg-eth0:1.1:*
node:30:/etc/sysctl.conf:1.2:*
node:30:/etc/sysctl.conf.14524:1.1:*
node:30:/etc/udev/rules.d/77-network.rules:1.1:*
node:5:/etc/exports:1.2:*
node:5:/etc/fstab:1.5:*
node:5:/etc/fstab.old:1.4:*
node:5:/etc/init.d/boot.local:1.1:*
node:5:/etc/motd:1.2:*
node:5:/etc/sysconfig/syslog:1.1:*
node:5:/etc/syslog-ng/syslog-ng.conf:1.2:*
node:8:/etc/sysconfig/network/ifcfg-ib0:1.1:*
node:9:/etc/sysconfig/network/ifcfg-ib0:1.1:*

```

Print specifications for files modified in the default view and include any warning messages

```

boot:~ # xtopview
default:// # xtoprdump -m -d -w
default:./etc/alps.gpus:1.2:*
default:./etc/bash.bashrc.local:1.5:*
default:./etc/bash.bashrc.local.rpmsave:1.2:*
...

```

Perform Archive Operations on Shared Root Files

Use the `xtoparchive` command to perform operations on an archive of shared root configuration files. Run the `xtoparchive` command on the boot node using the `xtopview` utility in the default view. The archive is a text-based file similar to a tar file and is specified using the required `archivefile` command-line argument. The `xtoparchive` command is intended for configuration files only. Binary files will not be archived. If a binary file is contained within a specification file list, it will be skipped and a warning will be issued.

The scope of this tool is limited to identification and manipulation of `/etc` configuration data within the shared root. Configuration files on the boot root file system or on the SMW are not managed by this utility.

For more information, see the `xtoparchive(8)` man page.

Add files specified by specifications listed in `specfile` to an archive file

```
boot:~ # xtopview
default:/:/ # xtoparchive -a -f specfile archive.20110422
```

List specifications for files currently in the `archive.20110422` archive file

```
boot:~ # xtopview
default:/:/ # xtoparchive -l archive.20110422
```

Disable Secure Shell (SSH) on Compute Nodes

By default, the SSH daemon, `sshd`, is enabled on compute nodes. To disable `sshd` follow this procedure.

1. Edit the `CLEinstall.conf` file and set `ssh_generate_root_sshkeys` to `no` (by default, this is set to `yes`).

```
smw:~ # vi CLEinstall.conf
ssh_generate_root_sshkeys=no
```

2. Invoke the `CLEinstall` program on the SMW specifying the `xtrelease` that is currently installed on the system set being used and is located in the `CLEmedia` directory.

```
smw:~ # /home/crayadm/install.xtrelease/CLEinstall --upgrade \
--label=system_set_label --XTrelease=xtrelease \
--configfile=/home/crayadm/install.xtrelease/CLEinstall.conf \
--CLEmedia=/home/crayadm/install.xtrelease
```

3. Type `y` and press the **Enter** key to proceed when prompted to update the boot root and shared root.

```
*** Do you wish to continue? (y/n) --> y
```

Upon completion, `CLEinstall` lists suggested commands to finish the installation. Those commands are also described here. For more information about running the `CLEinstall` program, see [Install and Configuring Cray Linux Environment \(CLE\) Software \(S-2444\)](#).

4. Rebuild the boot image using the `/var/opt/cray/install/shell_bootimage_LABEL.sh` script and the `xtbootimg` and `xtcli` commands. Suggested commands are included in output from `CLEinstall` and `shell_bootimage_LABEL.sh`. For more information about creating boot images, follow [Create a Boot Image from Existing File System Images](#).
5. Run the `shell_post.sh` script on the SMW to unmount the boot root and shared root file systems and perform other cleanup as necessary.

```
smw:~ # /var/opt/cray/install/shell_post_install.sh /bootroot0 /sharedroot0
```

Modify SSH Keys for Compute Nodes

The `dropbear` RPM is provided with the CLE release. Using `dropbear` SSH software, an administrator can supply and generate site-specific SSH keys for compute nodes in place of the keys provided by Cray.

Follow these steps to replace the RSA™ and DSA/DSS keys provided by the `CLEinstall` program.

1. Load the `dropbear` module.

```
crayadm@smw:~> module load dropbear
```

2. Create a directory for the new keys on the SMW.

```
crayadm@smw:~> mkdir dropbear_ssh_keys
crayadm@smw:~> cd dropbear_ssh_keys
```

3. Generate a `dropbear` compatible RSA key.

```
crayadm@smw:~/dropbear_ssh_keys> dropbearkey -t rsa -f ssh_host_rsa_key.db
Will output 1024 bit rsa secret key to 'ssh_host_rsa_key.db'
Generating key, this may take a while...
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgCQ9ohUgsrrBw5GNk7w2H5RcaBGajmUv8XN6fxg/
YqrsL4t5
CIkNghI3DQDxoiuC/ZVIJCtdwZLQJe708eiZee/tg5y2g8JIb3stg+ol/
9BLPDLMeX24FBhCweUpfGCO6Jfm4
Xg4wjKJIGrcmtDJAYoCRj0h9IrdDXXjpS7eI4M9XYZ
Fingerprint: md5 00:9f:8e:65:43:6d:7c:c3:f9:16:48:7d:d0:dd:40:b7
```

4. Generate a `dropbear` compatible DSS key.

```
crayadm@smw:~/dropbear_ssh_keys> dropbearkey -t dss -f ssh_host_dss_key.db
Will output 1024 bit dss secret key to 'ssh_host_dss_key.db'
Generating key, this may take a while...
Public key portion is:
ssh-dss
AAAAB3NzaC1kc3MAAACBAMEkThlE9N8iczLpfg0wUtuPtPcpIs7Y4KbG3Wg1T4CAEXDnmCKSyuCy
21TMAvVGCvYd80zPtL04yc1eUtD5RqEKy0h8jSBs0huEvhaJGHx9FzKfGhWi1ZOVX5vG3R+UCOXG
+71wZp3LU
yOcv/U+GWhalTWpUDaRU81MPRLW7rnAAAAFQCEqncqW61bouSORQ52d
+MRIwp27MwAAAIEAho69yAfGrNzxEI/
kjjDE5IaxjJpIBF262N9UsxleTX6F65OjNoL84fcKq1SL6NV5XJ5O00SKgTuVZjpXO913q9SEhkcI0Zy
0vRQ8
H5x3osZZ+Bq20QWof+CtWTqCoWN2xvne0NtET4lg81qCt/KGRq1tY6WG
+a01yrvunzQuafQAAACASXvs8h8AA
EK+3TEDj57rBRV4pz5JqWS1UaZStSQ2wJ3Oy1pIJiHkfqGWytv/
nSoWnr8YbQbvH9k1BsyQU8sOc5IJyCFu7+
Exomlyrxq/oirfeSgg6xC2rodcs+jH/K8EKoVtTak3/jHQeZWijRok4xDxwHdZ7e3l2HgYbZLmA5Y=
Fingerprint: md5 cd:a0:0b:41:40:79:f9:4a:dd:f9:9b:71:3f:59:54:8b
```

5. As `root`, copy the SSH keys to the boot image template.
To make these changes for a system partition, rather than for the entire system, replace `/opt/xt-images/templates` with `/opt/xt-images/templates-pN`, where `N` is the partition number.

For the RSA key:

```
crayadm@smw:~/dropbear_ssh_keys> su root
smw:/home/crayadm/dropbear_ssh_keys # cp -p ssh_host_rsa_key.db /opt/xt-images/
templates/default/etc/ssh/ssh_host_rsa_key
```

For the DSA/DSS key:

```
crayadm@smw:~/dropbear_ssh_keys> su root
smw:/home/crayadm/dropbear_ssh_keys # cp -p ssh_host_dss_key.db /opt/xt-images/
templates/default/etc/ssh/ssh_host_dss_key
```

6. Update the boot image to include these changes; follow the steps in [Prepare Compute and Service Node Boot Images](#) on page 37.

Configure Optional RPMs in the CNL Boot Image

There are two ways to configure which optional RPMs are installed into the CNL boot image for the system. First, several parameters are available in the `CLEinstall.conf` file to control whether specific RPMs are included during installation or upgrade of the system software. When editing `CLEinstall.conf` prior to running `CLEinstall`, set the `CNL_` parameters to either `yes` or `no` to indicate which optional RPMs should be included in the CNL compute node boot images. For example, to include these optional RPMs, change the following lines.

```
CNL_dvs=yes
CNL_ntpclient=yes
CNL_rsip=yes
```

The second method is to add or remove specific RPMs by editing the `/var/opt/cray/install/shell_bootimage_LABEL.sh` command used when preparing boot images for CNL compute nodes. Change the settings for these parameters to `y` or `n` to indicate which optional RPMs should be included. For example, to include the optional DVS and RSIP RPMs, change the following lines.

IMPORTANT: If changes are made directly to `/var/opt/cray/install/shell_bootimage_LABEL.sh`, it is important to make similar changes to the `CLEinstall.conf` file in order to avoid unexpected configuration changes during update or upgrade installations.

```
CNL_DVS=y
CNL_RSIP=y
```

Configure Memory Control Groups

CLE allows an administrator to force compute node applications to execute within memory control groups. Memory control groups are a Linux kernel feature that can improve the resiliency of the kernel and system services running on compute nodes while also accounting for application memory usage.

Before ALPS launches an application on a compute node, it determines how much memory is available. It then creates a memory control group for the application with a memory limit that is slightly less than the amount of available memory on the compute node. CLE tracks the application's memory usage, and if any allocations meet or try to exceed this limit, the allocation fails and the application aborts.

Since non-application processes execute outside of the memory control group and are not bound to this limit, system services should continue to execute normally during these low memory scenarios, resulting in improved resiliency for the kernel and system services. For details on how the memory control group limit is calculated, see the description of the `-M` option in the `apinit(8)` man page.

Adjust the Memory Control Group Limit

There are two methods for adjusting the memory control group limit.

1. (Method 1) Edit the `rcad_svcs.conf` in the compute node image in `/opt/xt-images`.
 - a. Change the `apinit -M` value in the compute node image `/etc/opt/cray/rca/rcad_svcs.conf` file. The following illustrates the `apinit` line within `rcad_svcs.conf`. The total amount of memory taken by the memory control group is the `-M` value multiplied by the number of cores on the reserved compute node.


```
apinit 0 3 1 0x7000016 0 /usr/sbin/apinit -n -r -M 400k
```
 - b. Rebuild the compute node and service node boot images as detailed in [Service Node and Compute Node Boot Image Preparation](#) to ensure the new value is used whenever the new boot image is used.
 - c. Reboot the compute nodes.
 - d. Copy the modified compute node `/etc/opt/cray/rca/rcad_svcs.conf` file into the default template directory in `/opt/xt-images/templates/` to ensure the change is not lost during an upgrade of CLE.
2. (Method 2) Set the memory control group limit using the `mcgroup` option with the `apmgr` command while the compute node is booted. Keep in mind that when the compute node is rebooted it will revert to the settings in the compute node image `/etc/opt/cray/rca/rcad_svcs.conf`. See the `apmgr(8)` man page for more details.

Disable Memory Control Groups

1. Open the `/etc/opt/cray/rca/rcad_svcs.conf` file in the compute node image and remove or comment-out the `apinit -M` option and corresponding value.
2. Within the compute node image edit `/boot/parameters-cn1` and set the `cgroup_disable` parameter to memory:

```
cgroup_disable=memory
```

3. Rebuild the compute node boot image as detailed in [Prepare Compute and Service Node Boot Images](#) on page 37.
4. Reboot the compute nodes using the newly created boot image.

There is a slightly higher likelihood that some applications will cause the compute nodes to experience OOM (out of memory) conditions if they run low on memory and memory control groups are disabled. However, most programs will not see this condition as it is highly dependent on application and site configurations.

Configure cron Services

NOTE: Configuring `cron` services is optional on CLE systems.

The `cron` daemon is disabled, by default, on the shared root file system and the boot root. It is enabled, by default, on the SMW. Use standard Linux procedures to enable `cron` on the boot root, following [Configure cron for the SMW and the Boot Node](#) on page 118.

On the shared root, configuring `cron` for CLE depends on whether persistent `/var` is set up. If persistent `/var` exists, follow [Configure cron for the Shared Root with Persistent /var](#) on page 118; otherwise, follow [Configure cron for the Shared Root without Persistent /var](#) on page 119.

The `/etc/cron.*` directories include a large number of `cron` scripts. During new system installations and any updates or upgrades, the `CLEinstall` program disables execute permissions on these scripts and they must be manually enabled to be used.

Configure cron for the SMW and the Boot Node

By default, the `cron` daemon on the SMW is enabled and this procedure is required only on the boot node.

1. Log on to the target node as `root` and determine the current configuration status for `cron`.

On the SMW:

```
smw:~# chkconfig cron
cron on
```

On the boot node:

```
boot:~ # chkconfig cron
cron off
```

2. Configure the `cron` daemon to start.
For this example, enable `cron` on the boot node:

```
boot:~ # chkconfig --force cron on
```

The `cron` scripts shipped with the Cray customized version of SLES are located under `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly`, and `/etc/cron.monthly`. The system administrator can enable these scripts by using the `chkconfig` command. However, if the system does not have a persistent `/var`, Cray recommends following [Configure cron for the Shared Root without Persistent /var](#).

Configure cron for the Shared Root with Persistent /var

Use this procedure for service nodes by using the shared root on systems that are set up with a persistent `/var` file system.

1. Invoke the `chkconfig` command in the default view to enable the `cron` daemon.

```
boot:~ # xtopview -m "configuring cron"
default:/: # chkconfig --force cron on
```

- Examine the `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly`, and `/etc/cron.monthly` directories and change the file access permissions to enable or disable distributed cron scripts to meet site needs. To enable a script, invoke `chmod ug+x` to make the script executable. By default, CLEinstall removes the execute permission bit to disable all distributed cron scripts.



CAUTION: Some distributed scripts impact performance negatively on a CLE system. To ensure that all scripts are disabled, type the following:

```
default:/ # find /etc/cron.hourly /etc/cron.daily /etc/cron.weekly \
/etc/cron.monthly -type f -follow -exec chmod ugo-x {} \;
```

- Exit `xtopview`.

```
default:/ # exit
```

Configure cron for the Shared Root without Persistent /var

Because CLE has a shared root, the standard cron initialization script `/etc/init.d/cron` activates the cron daemon on all service nodes. Therefore, the cron daemon is disabled by default and must be turned on with the `xtservconfig` command to specify the nodes on which the daemon should run.

- Edit the `/etc/group` file in the default view to add users who do not have root permission to the "trusted" group. The operating system requires that all cron users who do not have root permission be in the "trusted" group.

```
boot:~ # xtopview
default:/ # vi /etc/group
default:/ # exit
```

- Create a `/var/spool/cron` directory in the `/ufs` file system on the `ufs` node which is shared among all the nodes of class `login`.

```
boot:~ # ssh root@ufs
ufs:~# mkdir /ufs/cron
ufs:~# cp -a /var/spool/cron /ufs
ufs:~# exit
```

- Designate a single login node on which to run the scripts in this directory. Configure this node to start cron with the `xtservconfig` command rather than the `/etc/init.d/cron` script. This enables users, including `root`, to submit cron jobs from any node of class `login`. These jobs are executed only on the specified login node.

- Create or edit the following entry in the `/etc/sysconfig/xt` file in the shared root file system in the default view.

```
boot:~ # xtopview
default:/ # vi /etc/sysconfig/xt
```

```
CRON_SPOOL_BASE_DIR=/ufs/cron
```

```
default:/ # exit
```

- Start an `xtopview` shell to access all login nodes by class and configure the spool directory to be shared among all nodes of class `login`.

```
boot:~ # xtopview -c login
class/login:/ #
```

- c. Edit the `/etc/init.d/boot.xt-local` file to add the following lines.

```
class/login:/ # vi /etc/init.d/boot.xt-local
```

```
MYCLASS_NID=`rca-helper -i`
MYCLASS=`xtnce $MYCLASS_NID | awk -F: '{ print $2 }' | tr -d [:space:]`
CRONSPPOOL=`xtgetconfig CRON_SPOOL_BASE_DIR`
if [ "$MYCLASS" = "login" -a -n "$CRONSPPOOL" ];then
    mv /var/spool/cron /var/spool/cron.$$
    ln -sf $CRONSPPOOL /var/spool/cron
fi
```

- d. Examine the `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly`, and `/etc/cron.monthly` directories and change the file access permissions to enable or disable distributed cron scripts to meet site needs. To enable a script, invoke `chmod ug+x` to make the file executable. By default, CLEinstall removes the execute permission bit to disable all distributed cron scripts.



CAUTION: Some distributed scripts impact performance negatively on a CLE system. To ensure that all scripts are disabled, type the following:

```
class/login:/ # find /etc/cron.hourly /etc/cron.daily /etc/cron.weekly \
/etc/cron.monthly -type f -follow -exec chmod ugo-x {} \;
```

- e. Exit from the login class view.

```
class/login:/ # exit
```

- f. Enable the cron service on a single login node (node 8).

```
boot:~ # xtopview -n 8
node/8:/ # xtserveconfig -n 8 add CRON
node/8:/ # exit
```

The `cron` configuration becomes active on the next reboot. For more information, see the `xtserveconfig(8)` man page.

Configure the Node Health Checker (NHC)

For an overview of NHC (sometimes referred to as *NodeKARE*), see the `intro_NHC(8)` man page. For additional information about ALPS and how ALPS cooperates with NHC to perform application cleanup, see [About Modules and Modulefiles](#) on page 77.

The `nodehealth` Modulefile

To gain access to the NHC functions, the `nodehealth` module must be loaded. The `admin-modules` modulefile loads the `nodehealth` module, or load it directly, as follows:

```
% module load nodehealth
```

The `Base-opts.default.local` file includes the `admin-modules` modulefile. For additional information about the `Base-opts.default.local` file, see [System-wide Default Modulefiles](#).

Introduction

NHC can be run under two basic circumstances:

- immediately after applications within a reservation have terminated and immediately after a reservation has terminated
- when a node boots

To support running NHC at boot time and after applications and reservations complete, NHC uses two separate and independent configuration files, which enable NHC to be configured differently for these situations.

After Application and Reservation Termination

The configuration file that controls NHC behavior after a job has terminated is `/etc/opt/cray/nodehealth/nodehealth.conf`, located in the shared root. The CLE installation and upgrade processes automatically install this file and enable NHC software; there is no need to change any installation configuration parameters or issue any commands; however, the system administrator may edit this configuration file to customize NHC behavior. After editing, the changes made are reflected in the behavior of NHC the next time that it runs.

Compute Node Cleanup (CNCU) is called after an application or reservation terminate. Its objective is to efficiently return compute nodes to the pool of available nodes with as much free memory as they have when they are first booted. ALPS invokes NHC after every application completes and after every reservation completes. The NHC tests that run after applications are an *application set*. The NHC tests that run after reservations exit are a *reservation set*. With multiple test sets executing, CNCU requires more than one instance of NHC to be running simultaneously. The `advanced_features` NHC control variable must be enabled to use CNCU. The default setting of `advanced_features` in the example NHC configuration file is `on`.

When a Node Boots

The configuration file that controls NHC behavior on boot is located on the compute node. To change this file, the administrator must instead change its template, which is located on the SMW in one of two locations:

- On non-partitioned systems, the SMW template is `/opt/xt-images/templates/default/etc/opt/cray/nodehealth/nodehealth.conf`.
- On partitioned systems, the SMW template is `/opt/xt-images/templates/default-pN/etc/opt/cray/nodehealth/nodehealth.conf`, where `pN` is the partition number.

In either case, after modifying the `nodehealth.conf` file, it is necessary to remake the boot image for the compute node and reboot the node with the new boot image in order for the changes to take effect.

Each CLE release package also includes an example NHC configuration file, `/opt/cray/nodehealth/default/etc/nodehealth.conf.example`. The `nodehealth.conf.example` file is a copy of the `/etc/opt/cray/nodehealth/nodehealth.conf` file provided for an initial installation.

IMPORTANT: The `/etc/opt/cray/nodehealth/nodehealth.conf` file is not overwritten during a CLE upgrade if the file already exists. This preserves the site-specific modifications previously made to the file. However, the system administrator should compare the `/etc/opt/cray/nodehealth/nodehealth.conf` file content with the `/opt/cray/nodehealth/default/etc/nodehealth.conf.example` file provided with each

release to identify any changes, and then update the `/etc/opt/cray/nodehealth/nodehealth.conf` file accordingly.

If the `/etc/opt/cray/nodehealth/nodehealth.conf` file does not exist, then the `/opt/cray/nodehealth/default/etc/nodehealth.conf.example` file is copied to the `/etc/opt/cray/nodehealth/nodehealth.conf` file.

To use an alternate NHC configuration file, use the `xtcleanup_after -f alt_NHCconfigurationfile` option to specify which NHC configuration file to use with the `xtcleanup_after` script. For additional information, see the `xtcleanup_after(8)` man page.

NHC can also be configured to automatically dump, reboot, or dump and reboot nodes that have failed tests. This is controlled by the `action` variable specified in the NHC configuration file that is used with each NHC test and the `/etc/opt/cray-xt-dumpd/dumpd.conf` configuration file. For additional information, see [Dump and Reboot Nodes Automatically](#), the `dumpd(8)` man page, and the `dumpd.conf` configuration file on the SMW.

Disable the NHC

To disable NHC entirely, set the value of the `nhcon` global variable in the `/etc/opt/cray/nodehealth/nodehealth.conf` file to `off`. Default is `on`.

Configure NHC to Use SSL

NHC is configured to use secure sockets layer (SSL) protocol by default. Although this setting is configurable, Cray recommends that all sites configure NHC to use SSL.

If the site requires authentication and authorization to protect access to compute nodes, an administrator can configure compute nodes to perform node health checking by using the `openssl` utility and secure sockets layer (SSL) protocol. SSL provides optional security functionality for NHC.

To enable the use of SSL, set the `NHC_SSL` setting in the `CLEinstall.conf` file to `yes`.

For more information about configuring NHC to use SSL, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*.

Configure Node Health Checker Tests

Edit the `/etc/opt/cray/nodehealth/nodehealth.conf` file to configure the NHC tests that will test CNL compute node functionality. All tests that are enabled will run when NHC is in either Normal Mode or in Suspect Mode. Tests run in parallel, independently of each other, except for the `Free Memory Check` test, which requires that the `Application Exited Check` test passes before the `Free Memory Check` test begins.

The `xtcheckhealth` binary runs the NHC tests; for information about the `xtcheckhealth` binary, see the `intro_NHC(8)` and `xtcheckhealth(8)` man pages.

The NHC tests are listed below. In the default NHC configuration file, each test that is enabled starts with an action of `admindown`, except for `Free Memory Check`, which starts with an action of `log`.

Also read important test usage information in [Guidance for the Accelerator Test](#) on page 125, [Guidance for the Application Exited Check and Apinit Ping Tests](#) on page 125, [Guidance for the Filesystem Test](#) on page 126, [Guidance for the Hugepages Test](#) on page 127, and [Guidance for the NHC Lustre File System Test](#) on page 127.

Accelerator	<p>Tests the health of any accelerators present on the node. It is an application set test and should not be run in the reservation set.</p> <p>The global accelerator test (<code>gat</code>) script detects the type of accelerator(s) present on the node and then launches a test specific to the accelerator type. The test fails if it is unable to run successfully on the accelerator, or if the amount of allocated memory on the accelerator exceeds the amount specified using the <code>gat -m</code> argument.</p> <p>Default: enabled</p>
Application Exited Check	<p>Verifies that any remaining processes from the most recent application have terminated. It is an application set test and should not be run in the reservation set because an application is not associated with a reservation cancellation.</p> <p>The <code>Application Exited Check</code> test checks locally on the compute node for processes running under the ID of the application (APID). If processes are running, NHC waits a period of time (defined in the configuration file) to determine if the application processes exit properly. If the process does not exit within that time, this test fails.</p> <p>Default: enabled</p>
Apinit Log and Core File Recovery	<p>A plugin script to copy <code>apinit</code> core dump and log files to a login/service node. It is an application set test.</p> <p>Default: not enabled. <code>Apinit Log and Core File Recovery</code> should not be enabled until a destination directory is determined and specified in the NHC configuration file.</p>
Apinit Ping	<p>Verifies that the ALPS daemon is running on the compute node and is responsive. It is an application set test.</p> <p>The <code>Apinit Ping</code> test queries the status of the <code>apinit</code> daemon locally on each compute node; if the <code>apinit</code> daemon does not respond to the query, then this test fails.</p> <p>Default: enabled</p>
Free Memory Check	<p>Examines how much memory is consumed on a compute node while applications are not running. Use it only as a reservation test because an application within a reservation may leave data for another application in a reservation. If run in the application set, <code>Free Memory Check</code> could consider data that was intentionally left for the next application to be leaked memory and mark the node <code>admindown</code>. Run the <code>Free Memory Check</code> only after the <code>Reservation</code> test passes successfully.</p> <p>Default: enabled (action is <code>log</code> only)</p>
Filesystem	<p>Ensures that the compute node is able to perform simple I/O to the specified file system. It is configured as an application set test in the default configuration, but it can be run in the reservation set. For a file system that is mounted read-write, the test performs a series of operations on the file system to verify the I/O. A file is created, written, flushed, synced, and deleted. If a mount point is not explicitly specified, the mount point(s) from the compute node <code>/etc/fstabs</code> file will be used and a <code>Filesystem</code> test will be created for each mount point found in the file. If a mount point is explicitly specified, then only that file system will be checked. An administrator can specify multiple <code>Filesystem</code> tests by placing multiple <code>Filesystem</code> lines in the configuration file. For example, one line could specify the implicit <code>Filesystem</code> test, and the next line could specify a specific file system that does not appear in <code>/etc/fstab</code>. This could continue for any and all file systems.</p>

When enabling the `Filesystem` test, an administrator can place an optional line (such as, `Excluding: FileSystem-foo`) in the `/etc/opt/cray/nodehealth/nodehealth.conf` configuration file to list mount points that should not be tested by the `Filesystem` test. This allows intentionally excluding specific mount points even though they appear in the `fstab` file. This action prevents NHC from setting nodes to `admindown` because of errors on relatively benign file systems. Explicitly specified mount points cannot be excluded in this fashion; if they should not be checked, then they should simply not be specified.

The `Filesystem` test creates its temporary files in a subdirectory (`.nodehealth.fstest`) of the file system root. An error message is written to the console when the `unlink` of a file created by this test fails.

Default: enabled

Hugepages

Calculates the amount of memory available in a specified page size with respect to a percentage of `/proc/boot_freemem`. It is a reservation set test.

This test will continue to check until either the memory clears up or the time-out is reached. The default time-out is 300 seconds.

Default: disabled

Sigcont Plugin

Sends a `SIGCONT` signal to the processes of the current APID. It is an application set test.

Default: disabled

Plugin

Allows scripts and executables not built into NHC to be run, provided they are accessible on the compute node. For information about writing a plugin test, see *Writing a Node Health Checker (NHC) Plugin Test (S-0023)*.

Default: disabled so that local configuration settings may be used

ugni_nhc_plugins

Tests the User level Gemini Network Interface (uGNI) on compute nodes. It is a reservation set test and an application set test. By extension, testing the uGNI interface also tests the proper operation of parts of the network interface card (NIC). The test sends a datagram packet out to the node's NIC and back again.

Reservation

checks for the existence of the `/proc/reservations/rid` directory, where `rid` is the reservation ID. It is a reservation set test, and should not be run in the application set.

If this directory still exists, the test will attempt to end the reservation and then wait for the specified `timeout` value for the directory to disappear. If the test fails and Suspect Mode is enabled, NHC enters Suspect Mode. In Suspect Mode, `Reservation` continues running, repeatedly requesting that the kernel clean up the reservation, until the test passes or until Suspect Mode times out. If the directory does not disappear in that time, the test prints information to the console and exits with a failure.

Default: enabled with a `timeout` value of 300 seconds

CCM plugin

validates the cleanup of a cluster compatibility mode (CCM) environment at the end of a reservation. It is a reservation set test, and it will not run if it is misconfigured as an application test.

This test runs on a compute node only when `/var/crayccm` is detected. The test removes the `/var/lib/{empty,debus}` directories, unmounts CCM mount points if they still exist, and unmounts `/dsl/dev/random` and `/dsl/dev/pts`. If the unmounts are

successful, the test removes the `/var/crayccm`, `/var/lib/rpcbind`, and `/var/spool/{PBS,torque}` directories.

The `CCM plugin` is not included in a site's NHC configuration file. Administrators must add the test to their configuration in order to use it. See the `/opt/cray/nodehealth/default/etc/nodehealth.conf.example` file for `CCM plugin` settings to copy into a site's NHC configuration file.

Individual tests may appear multiple times in the `/etc/opt/cray/nodehealth/nodehealth.conf` file, with different variable values. Every time a test is specified in the `/etc/opt/cray/nodehealth/nodehealth.conf` file, NHC will run that test. This means if the same line is specified five times, NHC will try to run that same test five times. This functionality is mainly used in the case of the `Plugin` test, allowing the administrator to specify as many additional tests as have been written for the site, or the `Filesystem` test, allowing the administrator to specify as many additional file systems as wanted. However, any test can be specified to run any number of times. Different parameters and test actions can be set for each test. For example, this could be used to set up hard limits and soft limits for the `Free Memory Check` test. Two `Free Memory Check` tests could be specified in the configuration file; the first test configured to only warn about small amounts of non-free memory, and the second test configured to `admindown` a node that has large amounts of non-free memory. See the `/etc/opt/cray/nodehealth/nodehealth.conf` file for configuration information.

Guidance for the Accelerator Test

This test uses the global accelerator test (`gat`) script (`/opt/cray/nodehealth/default/bin/gat.sh`) to first detect the accelerator type and then launch the test specific to that type of accelerator.

The `gat` script supports two arguments for NVIDIA GPUs:

`-m` *maximum_memory_size* Specify the *maximum_memory_size* as either a kilobyte value or a percentage of total memory. For example, `-m 100` specifies that no more than 100 kilobytes of memory can be allocated, while `-m 10%` specifies that no more than 10 percent of memory can be allocated.

In the default NHC configuration file, the specified memory size is 10%.

`-r` Perform a soft restart on the GPU and then rerun the test. In the default NHC configuration file, the `-r` argument is specified.

The `gat` script has the following options for Intel Xeon Phi:

`-M` *kilobytes* or `-M` *n%* This option works exactly as the `-m` option for the NVIDIA GPUs.

`-c` Specifies the minimum number of cores that must be active on the Xeon Phi for the test to pass. If `-c` is omitted, the minimum number of active cores required to pass the test is the total number of cores on the Xeon Phi.

Guidance for the Application Exited Check and Apinit Ping Tests

These two tests must be enabled and both tests must have their action set as `admindown` or `die`; otherwise, NHC runs the risk of allowing ALPS to enter a live-lock. Only specify the `die` action when the `advanced_features` control variable is turned off.

ALPS must guarantee two conditions about the nodes in a reservation before releasing that reservation:

- that ALPS is functioning on the nodes
- that the previous application has exited from the nodes

Either those two conditions are guaranteed or the nodes must be set to some state other than `up`. When either ALPS has guaranteed these two conditions about the nodes or the nodes have been set to some state other than `up`, then ALPS can release the reservation.

These NHC tests guarantee two conditions:

- `Apinit_ping` guarantees that ALPS is functioning on the nodes
- `Application_Exited_Check` guarantees that the previous application has exited from the nodes

If either test fails, then NHC sets the nodes to `suspect` state if Suspect Mode is enabled; otherwise, NHC sets the nodes to `admindown` or `unavail`. Therefore, either the nodes pass these tests or the nodes are no longer in the `up` state. In either case, ALPS is free to release the reservation and the live-lock is avoided. Note that this only happens if the two tests are enabled and their action is set as `admindown` or `die`. The `log` action does not suffice because it does not change the state of the nodes. If either test is disabled or has an action of `log`, then ALPS may live-lock. In this live-lock, ALPS will call NHC endlessly.

Guidance for the Filesystem Test

The NHC `Filesystem` test can take an explicit argument (the mount point of the file system) or no argument. If an argument is provided, the `Filesystem` test is referred to as an explicit `Filesystem` test. If no argument is given, the `Filesystem` test is referred to as an implicit `Filesystem` test.

The explicit `Filesystem` test will test the file system located at the specified mount point.

The implicit `Filesystem` test will test each file system listed in the `/etc/fstab` file on each compute node. The implicit `Filesystem` test is enabled by default in the NHC configuration file.

The `Filesystem` test will determine whether a file system is mounted read-only or read-write. If the file system is mounted read-write, then NHC will attempt to write to it. If it is mounted read-only, then NHC will attempt to read the directory entities `"."` and `".."` in the file system to guarantee, at a minimum, that the file system is readable.

Some file systems are mounted on the compute nodes as read-write file systems, while their underlying permissions are read-only. As an example, for an auto-mounted file system, the base mount-point may have read-only permissions; however, it could be mounted as read-write. It would be mounted as read-write, so that the auto-mounted sub-mount-points could be mounted as read-write. The read-only permissions prevent tampering with the base mount-point. In a case such as this, the `Filesystem` test would see that the base mount-point had been mounted as a read-write file system. The `Filesystem` test would try to write to this file system, but the write would fail due to the read-only permissions. Because the write fails, the `Filesystem` test would fail, and NHC would incorrectly decide that the compute node is unhealthy because it could not write to this file system. For this reason, file systems that are mounted on compute nodes as read-write file systems, but are in reality read-only file systems, should be excluded from the implicit `Filesystem` test.

The administrator can exclude tests by adding an "Excluding: *file system mount point*" entry in the NHC configuration file. See the NHC configuration file for further details and an example.

A file system is deemed a critical file system if it is needed to run applications. All systems will likely need at least one shared file system for reading and writing input and output data. Such a file system would be a critical file system. File systems that are not needed to run applications or read and write data would be deemed as noncritical file systems. The administrator must determine the criticality of each file system.

Cray recommends the following:

- Exclude noncritical file systems from the implicit `Filesystem` test. See the NHC configuration file for further details and an example.
- If there are critical file systems that do not appear in the `/etc/fstab` file on the compute nodes (such file systems would not be tested by the implicit `Filesystem` test), these critical file systems should be checked through explicit `Filesystem` tests. Add explicit `Filesystem` tests to the NHC configuration file by providing the mount point of the file system as the final argument to the `Filesystem` test. See the NHC configuration file for further details and an example.
- If a file system that is mounted as read-write but it has read-only permissions, exclude it from the implicit `Filesystem` test. NHC does not support such file systems.

Guidance for the Hugepages Test

The `Hugepages` test runs the `hugepages_check` command, which supports two arguments:

- t *threshold*** Use this argument to specify the *threshold* as a percentage of `/proc/boot_freemem`. If this test is enabled and this argument is not supplied, the default of `-t 90` is used.
- s *size*** Specify the hugepage size. The valid sizes are 2, 4, 8, 16, 32, 64, 128, 256, and 512. If this test is enabled and this argument is not supplied, the default of `-s 2` is used.

Guidance for the NHC Lustre File System Test

The Lustre file system has its own hard time-out value that determines the maximum time that a Lustre recovery will last. This time-out value is called `RECOVERY_TIME_HARD`, and it is located in the file system's `fs_defs` file. The default value for the `RECOVERY_TIME_HARD` is 15 minutes.

IMPORTANT: The time-out value for the NHC `Lustre` file system test should be twice the `RECOVERY_TIME_HARD` value.

The default in the NHC configuration file is 30 minutes, which is twice the default `RECOVERY_TIME_HARD`. If the value of `RECOVERY_TIME_HARD` is changed, the time-out value of the NHC `Lustre` file system test must also change correspondingly.

The NHC time-out value is specified on the following line in the NHC configuration file:

```
# Lustre: <warning time-out> <test time-out> <restart delay>
Lustre: 900 1800 60
```

Further, the overall time-out value of NHC's Suspect Mode is based on the maximum time-out value for all of the NHC tests. Invariably, the NHC `Lustre` file system test has the longest time-out value of all the NHC tests.

IMPORTANT: If the NHC `Lustre` file system test time-out value is changed, then the time-out value for Suspect Mode must also be changed. The time-out value for Suspect Mode is set by the `suspectend` variable in the NHC configuration file. The guidance for setting the value of `suspectend` is that it should be the maximum time-out value, plus an additional buffer. In the default case, `suspectend` was set to 35 minutes -- 30 minutes for the Lustre test, plus an additional 5 minute buffer. For more information about the `suspectend` variable, see [NHC Suspect Mode](#).

NHC Control Variables

The following variables in `/etc/opt/cray/nodehealth/nodehealth.conf` affect the fundamental behavior of NHC.

nhcon	Turning off <code>nhcon</code> disables NHC entirely. Default: <code>on</code>
advanced_features	If set to <code>on</code> , this variable allows multiple instances of NHC to run simultaneously. This variable must be <code>on</code> to use CNCU and reservation sets. Default: <code>on</code>
dumpdon	If set to <code>off</code> , NHC will not request any dumps or reboots from <code>dumpd</code> . This is a quick way to turn off dump and reboot requests from NHC. The <code>dump</code> , <code>reboot</code> , and <code>dumpreboot</code> actions do not function properly when this variable is <code>off</code> . Default: <code>on</code>
anyapid	Turning <code>anyapid</code> on specifies that NHC should look for any <code>apid</code> in <code>/dev/cpuset</code> while running the Application Exited Check and print stack traces for processes that are found. Default: <code>off</code>

Global Configuration Variables that Affect all NHC Tests

The following global configuration variables may be set in the `/etc/opt/cray/nodehealth/nodehealth.conf` file to alter the behavior of all NHC tests. The global configuration variables are case-insensitive.

Runtests: <i>Frequency</i>	Determines how frequently NHC tests are run on the compute nodes. <i>Frequency</i> may be either <code>errors</code> or <code>always</code> . When the value <code>errors</code> is specified, the NHC tests are run only when an application terminates with a non-zero error code or terminates abnormally. When the value <code>always</code> is specified, the NHC tests are run after every application termination. If the <code>Runtests</code> global variable is not specified, the implicit default is <code>errors</code> . This variable applies only to tests in the application set; reservations do not terminate abnormally.
Connecttime: <i>TimeoutSeconds</i>	Specifies the amount of time, in seconds, that NHC waits for a node to respond to requests for the TCP connection to be established. If Suspect Mode is disabled and a particular node does not respond after <code>connecttime</code> has elapsed, then the node is marked <code>admindown</code> . If Suspect Mode is enabled and a particular node does not respond after <code>connecttime</code> has elapsed, then the node is marked <code>suspect</code> . NHC will then attempt to contact the node with a frequency established by the <code>recheckfreq</code> variable. If the <code>Connecttime</code> global variable is not specified, then the implicit default TCP time-out value is used. NHC will not enforce time-out on the connections if none is specified. The <code>Connecttime: TimeoutSeconds</code> value provided in the default NHC configuration file is 60 seconds.
cache_hosts: <i>[on off]</i>	Turn this variable on when NHC initialization is slow. If NHC startup time is satisfactory, the preferred setting is <code>cache_hosts: off</code> . Default: <code>off</code> .

The following global variables control the interaction of NHC and `dumpd`, the SMW daemon that initiates automatic dump and reboot of nodes.

maxdumps: <i>MaximumNodes</i>	Specifies the number of nodes that fail with the <code>dump</code> or <code>dumpreboot</code> action that will be dumped. For example, if NHC was checking on 10 nodes that all failed tests with the <code>dump</code> or <code>dumpreboot</code> actions, only the number of nodes specified by <code>maxdumps</code> would be dumped, instead of all of them. The default value is 1. To disable dumps of failed nodes with <code>dump</code> or <code>dumpreboot</code> actions, set <code>maxdumps: 0</code> .
downaction: <i>action</i>	Specifies the action NHC takes when it encounters a down node. Valid actions are <code>log</code> , <code>dump</code> , <code>reboot</code> , and <code>dumpreboot</code> . The default action is <code>log</code> .
downdumps: <i>number_dumps</i>	Specifies the maximum number of dumps that NHC will dump for a given APID, assuming that the <code>downaction</code> variable is either <code>dump</code> or <code>dumpreboot</code> . These dumps are in addition to any dumps that occur because of NHC test failures. The default value is 1.

The following global variables control the interaction between NHC, ALPS, and the SDB.

alps_recheck_max: <i>number of seconds</i>	NHC will attempt to verify its view of the states of the nodes with the ALPS view. If NHC is unable to contact ALPS, this variable controls the maximum delay between rechecks. Default value: 10 seconds
alps_sync_timeout: <i>number of seconds</i>	If NHC is unable to contact ALPS to verify the states of the nodes, this variable controls the length of time before NHC gives up and aborts. Default value: 1200 seconds
alps_warn_time: <i>number of seconds</i>	If NHC is unable to contact ALPS to verify the states of the nodes, this variable controls how often warnings are issued. Default value: 120 seconds
sdb_recheck_max: <i>number of seconds</i>	NHC will contact the SDB to query for the states of the nodes. If NHC is unable to contact the SDB, this variable controls the maximum delay between rechecks. Default value: 10 seconds
sdb_warn_time: <i>number of seconds</i>	If NHC is unable to contact the SDB, this variable controls how often warnings are issued. Default value: 120 seconds
node_no_contact_warn_time: <i>number of seconds</i>	If NHC is unable to contact a specific node, this variable controls how often warnings are issued. Default value: 600 seconds

The following global variable controls NHC's use of node states.

unhealthy_state: <i>swdown</i>	When a node is deemed unhealthy, it is normally set to <code>admindown</code> . This variable permits a different state to be chosen instead. Default: not set
--	---

unhealthy_state: When a node is going to be rebooted, it is normally set to `Unavail`. This variable permits a different state to be chosen instead.

rebootq

Default: not set

Standard Variables that Affect Individual NHC Tests

The following variables are used with each NHC test; set each variable for each test. All variables are case-insensitive. Each NHC test has values supplied for these variables in the default NHC configuration file.

Specific NHC tests require additional variables, which are defined in the `nodehealth` configuration file.

action Specifies the action to perform if the compute node fails the given NHC test. *action* may have one of the following values:

log Logs the failure to the system console log. The `log` action will not cause a compute node's state to be set to `admindown`.

IMPORTANT: Tests that have an action of `Log` do not run in Suspect Mode. If using plugin scripts with an action of `Log`, the script will only be run once, in Normal Mode. This makes log collecting and various other maintenance tasks easier to code.

admindown Sets the compute node's state to `admindown` (no more applications will be scheduled on that node) and logs the failure to the system console log.

If Suspect Mode is enabled, the node will first be set to `suspect` state, and if the test continues to fail, the node will be set to `admindown` at the end of Suspect Mode.

die Halts the compute node so that no processes can run on it, sets the compute node's state to `admindown`, and logs the failure to the system console log. (The `die` action is the equivalent of a kernel panic.) This action is good for catching bugs because the state of the processes is preserved and can be dumped at a later time.

If the `advanced_features` variable is enabled, `die` is not allowed.

Each subsequent action includes the actions that preceded it; for example, the `die` action encompasses the `admindown` and `log` actions.

If NHC is running in Normal Mode and cannot contact a compute node, and if Suspect Mode is not enabled, NHC will set the compute node's state to `admindown`.

The following actions control the NHC and `dumpd` interaction.

dump Sets the compute node's state to `admindown` and requests a dump from the SMW, in accordance with the `maxdumps` configuration variable.

reboot Sets the compute node's state to `unavail` and requests a reboot from the SMW. The `unavail` state is used rather than the `admindown` state when nodes are to be rebooted because a node that is set to `admindown` and subsequently rebooted stays in the `admindown` state. The `unavail` state does not have this limitation.

dumpreboot Sets the compute node's state to `unavail` and requests a dump and reboot from the SMW.

The following actions control the NHC and `dumpd` interaction.

- warntime** Specifies the amount of elapsed test time, in seconds, before `xtcheckhealth` logs a warning message to the console file. This allows an administrator to take corrective action, if necessary, before the `timeout` is reached.
- timeout** Specifies the total time, in seconds, that a test should run before an error is returned by `xtcheckhealth` and the specified `action` is taken.
- restartdelay** Valid only when NHC is running in Suspect Mode. Specifies how long NHC will wait, in seconds, to restart the test after the test fails. The minimum restart delay is one second.
- sets** Indicates when to run a test. The default NHC configuration specifies to run specific tests after application completion and to run an alternate group of tests at reservation end. When ALPS calls NHC at the end of the application, tests marked with `Sets: Application` are run. By default, these tests are: `Filesystem`, `Accelerator`, `ugni_nhc_plugins`, `Application Exited Check`, `Apinit Ping Test`, and `Apinit Log and Core File Recovery`. At the end of the reservation, ALPS calls tests marked `Sets: Reservation`. By default, these are: `Free Memory Check`, `ugni_nhc_plugins`, `Reservation`, and `Hugepages Check`.

If no set is specified for a test, it will default to `Application`, and run when ALPS calls NHC at the end of the application. If NHC is launched manually, using the `xtcheckhealth` command, and the `-m sets` argument is not specified on the command line, then `xtcheckhealth` defaults to running the `Application` set.

If a test is marked `Sets: All`, it will always run, regardless of how NHC is invoked.

NHC Suspect Mode

Upon entry into Suspect Mode, NHC immediately allows healthy nodes to be returned to the resource pool. Suspect Mode allows the remaining nodes, which are all in `suspect` state, an opportunity to return to healthiness. If the nodes do not return to healthiness by the end of the Suspect Mode (determined by the `suspectend` global variable; see below), their states are set to `admindown`. For more information about how Suspect Mode functions, see the `intro_NHC(8)` man page.

IMPORTANT: Suspect Mode is enabled in the default `/etc/opt/cray/nodehealth/nodehealth.conf` configuration file. Cray recommends that sites run NHC with Suspect Mode enabled.

If enabled, the default NHC configuration file uses the following Suspect Mode variables:

- suspectenable:** Enables Suspect Mode; valid values are `y` and `n`.
Default: `y`
- suspectbegin:** Sets the Suspect Mode timer. Suspect Mode starts after the number of seconds indicated by `suspectbegin` have expired.
Default: `180`
- suspectend:** Suspect Mode ends after the number of seconds indicated by `suspectend` have expired. This timer only starts after NHC has entered Suspect Mode.
Default: `2100`

Considerations when evaluating shortening the length of Suspect Mode:

- The length of Suspect Mode can be shortened if there are no external file systems, such as Lustre, for NHC to check.
- Cray recommends that the length of Suspect Mode be at least a few seconds longer than the longest time-out value for any of the NHC tests. For example, if the `Filesystem` test had the longest time-out value at 900 seconds, then the length of Suspect Mode should be at least 905 seconds.
- The longer the Suspect Mode, the longer nodes have to recover from any unhealthy situations. Setting the length of Suspect Mode too short reduces this recovery time and increases the likelihood of the nodes being marked `admindown` prematurely.

recheckfreq: Suspect Mode rechecks the health of the nodes in `suspect` state at a frequency specified by `recheckfreq`. This value is in seconds.

For a detailed description about NHC actions during the recheck process, see the `intro_NHC(8)` man page.

Default: 300

NHC Messages

NHC messages may be found on the SMW in `/var/opt/cray/log/sessionid/nhc-YYYYMMDD` with '`<node_health:M.m>`' in the message, where *M* is the major and *m* is the minor NHC revision number. All NHC messages are visible in the console file.

NHC prints a summary message per node at the end of Normal Mode and Suspect Mode when at least one test has failed on a node. For example:

```
<node_health:3.1> APID:100 (xtnhc) FAILURES: The following tests have failed in
normal mode:
<node_health:3.1> APID:100 (xtnhc) FAILURES: (Admindown) Apinit_Ping
<node_health:3.1> APID:100 (xtnhc) FAILURES: (Admindown) Plugin /example/plugin
<node_health:3.1> APID:100 (xtnhc) FAILURES: (Log Only ) Filesystem_Test on /
mydir
<node_health:3.1> APID:100 (xtnhc) FAILURES: (Admindown) Free_Memory_Check
<node_health:3.1> APID:100 (xtnhc) FAILURES: End of list of 5 failed test(s)
```

The `xtcheckhealth` error and warning messages include node IDs and application IDs and are written to the console file on the SMW; for example:

```
[2010-04-05 23:07:09][c1-0c2s0n0]<node_health:3.0> APID:2773749
(check_apid) WARNING: Failure: File /dev/cpuset/2773749/tasks exists and is not
empty. \
The following processes are running under expired APID
2773749:
[2010-04-05 23:07:09][c1-0c2s0n1]<node_health:3.0> APID:2773749
(check_apid) WARNING: Pid: 300 Name: (marys_program) State: D
```

The `xtcleanup_after` script writes its normal launch information to the `/var/log/xtcheckhealth_log` file, which resides on the login nodes. The `xtcleanup_after` launch information includes the time that `xtcleanup_after` was launched and the time `xtcleanup_after` called `xtcheckhealth`.

The `xtcleanup_after` script writes error output (launch failure information) to the `/var/log/xtcheckhealth_log` file, to the console file on the SMW, and to the `syslog`.

Example `xtcleanup_after` output follows:

```
Thu Apr 22 17:48:18 CDT 2010 <node_health> (xtcleanup_after)
/opt/cray/nodehealth/3.0-1.0000.20840.30.8.ss/bin/xtcheckhealth -a 10515
-e 1 /tmp/apsysLVNqO9 /etc/opt/cray/nodehealth/nodehealth.conf
```

If a Login Node Crashes While `xtcheckhealth` Binaries are Monitoring Nodes

If a login node crashes while `xtcheckhealth` binaries on that login node are monitoring compute nodes that are in `suspect` state, those `xtcheckhealth` binaries will die when the login node crashes. When the login node that crashed is rebooted, a recovery action takes place. When the login node boots, the `node_health_recovery` binary starts up. This script checks for all compute nodes that are in `suspect` state and were last set to `suspect` state by this login node. The script then determines the APID of the application that was running on each of these compute nodes at the time of the crash. The script then launches an `xtcheckhealth` binary to monitor each of these compute nodes. One `xtcheckhealth` binary is launched per compute node monitored.

If the `Application_Exited_Check` test is enabled in the configuration file (default), `xtcheckhealth` is launched with this APID to test for processes that may have been left behind by that application. Otherwise, NHC does not run the `Application_Exited_Check` test and will not check for leftover processes, but will run any other NHC tests that are enabled in the configuration file.

Nodes are changed from `suspect` state to `up` or `admindown`, depending upon whether they fail any health checks. No system administrator intervention should be necessary.

NHC automatically recovers the nodes in `suspect` state when the crashed login node is rebooted because the recovery feature runs on the rebooted login node. If the crashed login node is not rebooted, then manual intervention is required to rescue the nodes from `suspect` state. This manual recovery can commence as soon as the login node has crashed. To recover from a login node crash during the case in which a login node will not be rebooted, the `nhc_recovery` binary is provided to help release the compute nodes owned by the crashed login node; see [Recover from a Login Node Crash when a Login Node will not be Rebooted](#). Also, see the `nhc_recovery(8)` man page for a description of the `nhc_recovery` binary usage.

Recover from a Login Node Crash when a Login Node will not be Rebooted

1. Create a file, `nodelistfile`, that contains a list of the nodes in the system that are currently in Suspect Mode. The file must be a list of NIDs, one per line; do not include a blank line at the end of the file.
2. List all of the `suspect` nodes in the system and the login nodes to which they belong.

```
smw:~# nhc_recovery -d nodelistfile
```

3. Parse the `nhc_recovery` output for the NID of the login node that crashed, creating a file, `computenodes`, that contains all of the compute nodes owned by the crashed login node.
4. Use the `computenodes` file to create `nodelist` files containing nodes that share the same APID (to determine the nodes from the crashed login node). For example, the files can be named `nodelistfile-APID1`, `nodelistfile-APID2`, `nodelistfile-APID3`, etc.

5. Release all of the `suspect` compute nodes owned by the crashed login node.

```
smw:~# nhc_recovery -r computenodes
```

All of these compute nodes are released in the database, but they are all still in `suspect` state.

6. Determine what to do with these `suspect` nodes from the following three options:
 - (Cray recommends this option) Rerun NHC on a non-crashed login node to recover the nodes listed in step 4 on page 133. Invoke NHC for each `nodelist-APID` file. Supply the APID that corresponds to the `nodelistfile`; an iteration count of 0 (zero), which is the value normally supplied to NHC by ALPS; and an application exit code of 1 as the APID argument. An exit code of 1 ensures that NHC will run regardless of the value of the `runtests` variable (`always` or `errors`) in the NHC configuration file. For example:


```
smw:~# xtcleanup_after -s nodelist-APID1 APID1 0 1
smw:~# xtcleanup_after -s nodelist-APID2 APID2 0 1
smw:~# xtcleanup_after -s nodelist-APID3 APID3 0 1
.
.
.
```
 - Set the `suspect` nodes to `admindown` and determine their fate by further analysis.
 - Set the `suspect` nodes back to `up`, keeping in mind that they were in Suspect Mode for a reason.

Configure Fast Compute Node Reboot

Fast Compute Node Reboot (FCNR) is an option of the `xtbootsys` command that reboots compute nodes without bouncing them, reducing the time needed for reboots. FCNR addresses a specific problem: memory fragmentation reaches a level that prevents allocation of huge pages. Do not use FCNR as a general method of rebooting nodes.

The intended means of calling FCNR is through the configurations of NHC and the SMW daemon `dumpd`. The `xtbootsys` command syntax for use of FCNR specifies the `--fast` option:

```
# xtbootsys --reboot --fast [node_list]
```

The following list describes what happens when a node boots using FCNR:

- `xtbootsys` calls `xtcli shutdown` to cleanly shut down the node.
- After the shutdown completes, the fast reboot daemon is loaded outside of the kernel so that it is not overwritten when the new kernel loads. This daemon waits for a signal from the boot node.
- `xtbootsys` waits for the `Fast reboot daemon ready` message in the console log.
- `xtbootsys` calls `xtcli boot` to load the new kernel into the node.
- The boot node signals the fast reboot daemon to jump into newly loaded kernel, starting the node boot.

Limitations

A node and the system in general must be healthy to use FCNR. A node in the `admindown` state can be rebooted successfully using FCNR. Accelerator nodes cannot be rebooted using FCNR.

FCNR might not succeed under the following circumstances:

- A file server is so loaded that the node cannot unmount its file systems within 20 to 30 seconds.

- NHC detects a problem with the node, such as a hung process.
- The console log is being flooded.

Enabling FCNR

In the NHC configuration file, `/etc/opt/cray/nodehealth/nodehealth.conf`:

- Under the NHC control variables section, ensure that `dumpd` is set to `on`.
- Ensure that the Hugepages Check is uncommented, which enables it.

In the `dumpd` configuration file, `/etc/opt/cray-xt-dumpd/dumpd.conf`:

- Under the reboot action definition, specify the `--fast` option for the `xtbootsys` command.

It is not possible to specify different reboot methods (fast reboot versus the conventional bounce) for the various NHC tests. The reboot action defined in the `dumpd` configuration file applies to all reboot actions specified for NHC tests.

Boot-node Failover

The boot node is integral to the operation of a Cray system. The following services run on the boot node:

- NFS shared root (read-only)
- NFS persistent `/var` (read-write)
- Boot node daemon, `bnd`
- Hardware supervisory system (HSS) and system database (SDB) synchronization daemon, `xtdbsyncd`
- ALPS daemons `apbridge`, `apres`, and `apwatch`

CLE provides functionality to create a standby boot node that automatically acts as a backup in the event of primary node failure. Failover allows the system to keep running without an interrupt to the system or the services run on the boot node; however, it does not provide a failback capability.

A virtual network is configured for the boot and SDB nodes to support failover for these nodes. The virtual network is configured by default, regardless of the boot or SDB node failover configuration on the system. The `CLEinstall` program provides the capability to change the default virtual network configuration, however, the default values are acceptable in most cases. For more information, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)* or the `CLEinstall.conf(5)` man page.

Sequence of Events

When the primary boot node is booted, the backup boot node also begins to boot. However, the backup boot node makes a call to the `rca-helper` utility before it mounts its root file system, causing the backup boot node to be suspended until a primary boot-node failure event is detected.

The `rca-helper` daemon running on the backup boot node waits for a primary boot-node failure event, `ec_node_failed`. When the heartbeat of the primary boot node stops, the blade controller begins the heartbeat checking algorithm to determine if the primary boot node has failed. When the blade controller determines that the primary boot node has failed, it sends an `ec_heartbeat_stop` event to set the alert flag for the primary node. The primary boot node is halted through `STONITH`. Setting the `alert` flag on the node triggers the HSS state manager on the SMW to send out the `ec_node_failed` event.

When the `rca-helper` daemon running on the backup boot node receives an `ec_node_failed` event alerting it that the primary boot node has failed, it allows the boot process of the backup boot node to continue. Any remaining boot actions occur on the backup boot node. Booting of the backup boot node takes approximately two minutes.

Each service node runs a failover manager daemon (`fomd`). When a service node's `fomd` receives the `ec_node_failed` event, it takes appropriate action. The `fomd` process updates the ARP cache entry for the boot node virtual IP address to reference the backup boot node.

The purpose of this implementation of boot-node failover is to ensure that the system continues running, not to guarantee that every job will continue running. Therefore, note the following:

- During the time the primary boot node has failed, any service node that tries to access its root file system will be I/O blocked until the backup boot node is online, at which time the request will be satisfied and the operation will resume. In general, this means if an application is running on a service node, it can continue to run if the application is in memory and does not need to access disk. If it attempts to access disk for any reason, it will be blocked until the backup boot node is online.
- Applications running on compute nodes are affected only if they cause a service node to access its root file system, in which case the service node function would be blocked until the backup boot node is online.

Requirements

The following is a list of requirements for configuring a system for boot-node failover:

- The backup boot node requires a Host Bus Adapter (HBA) card to communicate with the RAID.
 - IMPORTANT:** The backup boot node must be configured in the same zone as the primary boot node.
- The boot RAID host port must be able to see the desired LUNs; for DDN, use the host port mapping; for NetApp (formerly LSI and Engenio), use SANshare in the SANtricity® Storage Manager.
- The backup boot node also requires a Gigabit Ethernet card connected through a Gigabit Ethernet switch to the same port on the SMW as the primary boot node (typically port 4 of the SMW quad Ethernet card).
- STONITH capability must be enabled on the blade or module of the primary boot node in order to use the boot node failover feature. STONITH is a per blade setting and not a per node setting. Ensure that the primary boot node is located on a separate blade from services with conflicting STONITH requirements such as Lustre.

Configure Boot-node Failover

If boot-node failover was configured during the CLE software installation or upgrade (as documented in *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*), this procedure is not needed.

TIP: Use the `nid2nic` or `rtr --system-map` commands to translate between node or NIC IDs and physical ID names.

1. As `crayadm` on the SMW, halt the primary and alternate boot nodes.



WARNING: Verify that the system is shut down before invoking the `xtcli halt` command.

```
crayadm@smw:~> xtcli halt primary_id, backup_id
```

2. Update the default boot configuration used by the boot manager based on the system setup:

- Default:

```
crayadm@smw:~> xtcli boot_cfg update -b primary_id, backup_id -i /bootimagedir/
bootimage
```

- If using /raw0:

```
crayadm@smw:~> xtcli boot_cfg update -i /raw0
```

- If using partitions:

```
crayadm@smw:~> xtcli part_cfg update pN -b primary_id,backup_id -i /bootimagedir/
bootimage
```

- If using partitions and /raw0

```
crayadm@smw:~> xtcli part_cfg update pN -i /raw0
```

3. Update the CLEinstall.conf file to designate the primary and backup boot nodes so the file has the correct settings for the next upgrade.

4. Boot the boot node.

5. Determine the current STONITH setting on the primary boot node. If the system is partitioned, invoke these commands with the `--partition pn` option. The STONITH capability must be enabled on the blade of the primary boot node in order to use the boot-node failover feature.



CAUTION: STONITH is a per blade setting, not a per node setting. Ensure that the primary boot node is not assigned to a blade that hosts services with conflicting STONITH requirements such as Lustre.

For this example, the primary boot node is `c0-0c0s0n1` located on blade `c0-0c0s0`:

```
crayadm@smw:~> xtdaemonconfig c0-0c0s0 | grep stonith
c0-0c0s0: stonith=false
```

6. Enable STONITH on the primary boot node if necessary.

```
crayadm@smw:~> xtdaemonconfig c0-0c0s0 stonith=true
c0-0c0s0: stonith=true
The expected response was received.
```

7. The STONITH setting does not survive a power cycle. To maintain the STONITH setting for the primary boot node add the following line to the boot automation file:

```
# boot bootnode:
lappend actions {crms_exec "xtdaemonconfig c0-0c0s0 stonith=true"}
```

8. Boot the system.

Disable Boot-node Failover

For the examples in this procedure, the primary boot node is `c0-0c0s0n1` and the backup boot node is `c2-0c1s7n1`.

1. Halt the primary and backup boot nodes.

```
crayadm@smw:~> xtcli halt c0-0c0s0n1,c2-0c1s7n1
```

2. Update the default boot configuration.

```
crayadm@smw:~> xtcli boot_cfg update -b c0-0c0s0n1,c0-0c0s0n1
```

3. Update the HSS daemon.

```
crayadm@smw:~> xtdaemonconfig c0-0c0s0 stonith=false
```

Compute Node Failover Manager

The compute node failover manager daemon (`cnfomd`) facilitates communication from the compute nodes to the backup boot or SDB node in the event of a primary boot or SDB node failure. When a node failed event from the primary boot or SDB node is detected, `cnfomd` updates the ARP cache entries for the boot or SDB node virtual IP address to point to the backup node. The daemon runs on the compute nodes and is similar to the failover manager daemon (`fomd`) on the service nodes. If both boot and SDB node failover are disabled, the `cnfomd` process exits immediately after start up.

This functionality is included in the `cray-rca-compute` RPM and is installed by default.

Configure SDB Node Failover

Critical services such as the Application Level Placement Scheduler (ALPS) and Lustre rely on the SDB and will fail if the SDB node is unavailable. The CLE release provides functionality to create a standby SDB node that automatically acts as a backup in the event of primary node failure. When a secondary (backup) SDB node is configured, SDB node failover occurs automatically when the primary SDB node fails.

The CLE implementation of SDB node failover includes installation configuration parameters that facilitate automatic configuration, a `chkconfig` service called `sdbfailover`, and a `sdbfailover.conf` configuration file for defining site-specific commands to invoke on the backup SDB node.

The backup SDB node uses `/etc` files that are class or node specialized for the primary SDB node and not for the backup node itself; the `/etc` files for the backup node will be identical to those that existed on the primary SDB node. The backup SDB node also mounts the NID-specific persistent `/var` file system of the primary SDB node. For example, if the SDB NID is 270 and the backup SDB NID is 273 then `/snv/270/var` gets mounted on the backup SDB node when it takes over instead of `/snv/273/var`.

The following list summarizes requirements to implement SDB node failover on a Cray system.

- Designate a service node to be the alternate or backup SDB node. The backup SDB node requires a Host Bus Adapter (HBA) card to communicate with the RAID. This backup node is dedicated and cannot be used for other service I/O functions.
- Enable the STONITH capability on the blade or module of the primary SDB node in order to use the SDB node failover feature. STONITH is a per blade setting and not a per node setting. Ensure that the primary SDB node is located on a separate blade from services with conflicting STONITH requirements, such as Lustre.
- Enable SDB node failover by setting the `sdbnode_failover` parameter to `yes` in the `CLEinstall.conf` file prior to running the `CLEinstall` program.

When this parameter is used to configure SDB node failover, the `CLEinstall` program will verify and turn on `chkconfig` services and associated configuration files for `sdbfailover`.

- Specify the primary and backup SDB nodes in the boot configuration by using the `xtcli` command with the `boot_cfg update -d` options. For more information, see the `xtcli(8)` man page.
- (Optional) Populate `/etc/opt/cray/sdb/sdbfailover.conf` with site-specific commands.

When a failover occurs, the backup SDB node invokes all commands listed in the `/etc/opt/cray/sdb/sdbfailover.conf` file. Include commands in this file that are normally invoked during system start-up through boot automation scripts. In a SDB node failover situation, these commands must be invoked on the new (backup) SDB node. For example, include commands to start batch system software (if not started through `chkconfig`) or commands to add a route to an external license server.

If at any time the system is reconfigured to use a different primary SDB node, STONITH must be enabled for the new SDB node and disabled for the previous node.

For procedures to configure SDB node failover during a CLE software installation, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*.

Create Logical Machines for Cray XE/XK Systems

Configure a logical machine (sometimes known as a *system partition*) with the `xtcli part_cfg` command. Partition IDs are predefined as `p0` to `p31`. The default partition `p0` is reserved for the complete system and is no longer a valid ID once a system has been partitioned.

Create Routable Logical Machines

A routable logical machine is generally one that is logically a cube. The topology class of the system indicates how the system is physically cabled together, which in turn, determines the logical structure of the system. It is easiest to describe the routing based on physical location. Because it is impossible to route around some types of failures without a torus in the z-dimension, do not divide the system in a way that breaks the z-dimension torus.

Topology Class 0

These are the smallest systems. A topology class 0 system can contain one to nine chassis in up to three cabinets. Each chassis has its y- and z-dimensions looped back on itself. The chassis are connected in the x-dimension.

To partition the system, break up the configuration in the x-dimension by grouping a number of chassis together. Thus, to define the partitions, it is necessary to know the order in which the chassis are cabled together. Table 5

shows the order of the chassis. The last chassis in the list is cabled back to the first chassis in the list to complete the torus.

Table 5. Topology 0 Chassis Layout

Number of Chassis	Order of Chassis in x-Dimension
1	c0-0c0
2	c0-0c0, c0-0c1
3	c0-0c0, c0-0c1, c0-0c2
4	c0-0c0, c0-0c1, c0-0c2, c1-0c1
5	c0-0c0, c0-0c1, c0-0c2, c1-0c1, c1-0c0
6	c0-0c0, c0-0c1, c0-0c2, c1-0c2, c1-0c1, c1-0c0
7	c0-0c0, c0-0c1, c0-0c2, c1-0c2, c1-0c1, c2-0c0, c1-0c0
8	c0-0c0, c0-0c1, c0-0c2, c1-0c2, c1-0c1, c2-0c1, c2-0c0, c1-0c0
9	c0-0c0, c0-0c1, c0-0c2, c1-0c2, c1-0c1, c2-0c2, c2-0c1, c2-0c0, c1-0c0

To partition the system on a cabinet basis, the administrator must take the particular configuration and the logical chassis ordering shown in THE TABLE into account. For example, for a three-cabinet (nine-chassis) topology class 0 system, the administrator can partition the system on a cabinet basis as follows:

c0-0, c1-0 and c2-0

Or:

c0-0 and c1-0, c2-0

Cabinet c1-0 cannot be a partition on its own because the three chassis are not all directly connected together. Cabinets c0-0 and c2-0 can each be independent partitions because all three chassis for each of these cabinets are directly connected together.

Topology Class 1

Class 1 topology systems contain a single row of cabinets. Generally, systems have 4 to 15 cabinets. The three chassis in each cabinet are cabled together in the y-dimension. The z-dimension is looped back on itself within the chassis. The cabinets are then cabled together in the x-dimension.

To create a torus in the x-dimension, the cabinets are cabled in an interleaved fashion. This means that cabinet 0 in the row is cabled to cabinet 2, which is cabled to 4, and so on to the end of the row. At this point, the highest-numbered even cabinet is cabled to the highest-numbered odd cabinet. Then the odd cabinets are cabled together, coming back down the row to cabinet 1. To complete the torus, cabinet 1 is cabled to cabinet 0.

To partition this system, an administrator can:

- Group together a consecutive number of even (or odd) cabinets. For example, create two logical machines, one with all the even cabinets and another with the odd cabinets.
- Group together consecutive cabinets on each end of the row. For example, partition a 12-cabinet system with cabinets 0-5 in one partition and cabinets 6-11 in another.

- Group a combination of cabinets. For example, for a 12-cabinet system, define three logical machines containing cabinets 0-5; 6,8,10; and 7,9,11, respectively.

Topology Class 2

Topology class 2 systems are configured with two equal-sized rows of cabinets. The chassis within the cabinet are cabled together in the y-dimension. Corresponding cabinets in each row are cabled together in the z-dimension. That is, they are cabled together by pairing up chassis within the cabinets, and then cabling them together. The chassis are paired chassis0-chassis2, chassis1-chassis1, and chassis2-chassis0. The x-dimension within each row is cabled the same interleaved fashion as is topology class 1.

To partition a topology class 2 system, keep pairs of corresponding cabinets together and do not break the z-dimension. Thus, topology class 2 can be partitioned in the same way as topology class 1. The logical machine includes the cabinets from both rows.

Topology Class 3

Topology class 3 systems contain multiple equal-sized rows of cabinets. These can be cabled in two ways:

- The y-dimension is a torus.

There must be an even number of rows in this configuration.

- The y-dimension is a mesh.

This configuration can have any number of rows, typically three or more. The y-dimension is cabled between the rows. The z-dimension cables the three chassis within a cabinet together. The x-dimension is cabled down each row, in the same configuration as topology classes 1 and 2.

There are many ways to create a logical machine for a topology class 3 system. Make sure that all partitions are rectangular with respect to the cabinets. The administrator must also account for x-dimension cabinet interleaving. Rows are more complicated to divide when the y-dimension is a torus, especially for systems with row counts greater than four. A subset of the number of rows can be taken to make a partition. Taking corresponding cabinets from all rows leaves the y-dimension torus intact, which in general helps performance.

A two-row Gemini topology class 3 system is not supported because it does not lend itself well to cycle-free fault tolerant routing (because of the Y dimension of size 4).

Configure a Logical Machine

The logical machine can have one of three states:

- `EMPTY` - not configured
- `DISABLED` - configured but not activated
- `ENABLED` - configured and activated

When a partition is defined, its state changes to `DISABLED`. Undefined partitions are `EMPTY` by default.

The `xtcli part_cfg` command

Use the `xtcli part_cfg` command with the `part_cmd` option (add in the following example) to identify the operation to be performed and the `part_option` (`-m`, `-b`, `-d` and `-i`) to specify the characteristics of the logical machine. The boot image may be a raw device, such as `/raw0`, or a file.

Create a logical machine with a boot node and SDB node specifying the boot image path

- When using a file for the boot image, the same file must be on both the SMW and the `bootroot` at the same path.
- For the logical machine to be bootable, both the boot node and SDB node IDs must be specified.

```
crayadm@smw:~> xtcli part_cfg add p2 -m c0-0,c0-1,c0-2,c0-3 \
-b c0-0c0s0n0 -d c0-0c0s2n1 -i /bootimagedir/bootimage
```

To watch HSS events on the specified partition, execute the `xtconsumer -p partition_name` command.

To display the console text of the specified partition, execute the `xtconsole -p partition_name` command.

For more information, see the `xtcli_part(8)`, `xtconsole(8)`, and `xtconsumer(8)` man pages.

Boot a Logical Machine

The `xtbootsys --partition pN` option enables the administrator to indicate which logical machine (partition) to boot. If a partition name is not specified, the default partition `p0` (component name for the entire system) is booted. Alternatively, if a partition name is not specified and the `CRMS_PARTITION` environment variable is defined, this variable is used as the default partition name. Valid values are in the form `pN`, where *N* ranges from 0 to 31.

`xtbootsys` manages a link from `/var/opt/cray/log/partition-current` to the current `sessionid` directory for that partition, allowing changes to `/var/opt/cray/log/p1-current`, for example.

Update the Boot Configuration

The HSS `xtcli boot_cfg` command allows the administrator to specify the primary and backup boot nodes and the primary and backup SDB nodes for `s0` or `p0` (the entire system).

For a partitioned system, use `xtcli part_cfg` to manage boot configurations for partitions.

For more information, see the `xtcli_boot(8)` and `xtcli_part(8)` man pages.

For this example, update the boot configuration using the boot image `/bootimagedir/bootimage`, primary boot node (for example, `c0-0c0s0n1`), backup boot node, primary SDB node, and the backup SDB node:

```
crayadm@smw:~> xtcli boot_cfg update -b primaryboot_id,backupboot_id \
-d primarySDB_id,backupSDB_id -i /bootimagedir/bootimage
```

Modify Boot Automation Files

The boot automation files should be located in `/opt/cray/hss/default/etc` on the SMW. There are several automation files; for example, `auto.generic.cnf` and `auto.min.cnf`.

For boot automation scripts, the Lustre file system should start up before the compute nodes. The system administrator can also boot the system or shut down the system using both user-defined and built-in procedures in the `auto.xtshutdown` file. For related procedures, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*.

If boot automation files are used, see the `xtbootsys(8)` man page, which provides detailed information about boot automation files, including descriptions of using the `xtbootsys crms_boot_loadfile` and `xtbootsys crms_boot_sdb_loadfile` automation file procedures.

Change the System Software Version to be Booted

Release switching enables the administrator to change between versions and releases of the CLE software that are installed concurrently on the system.

The operating system must be rebooted in order to switch CLE releases on a Cray system. A release change cannot occur while the mainframe is running. Reboot is required each time the version changes; but, the SMW does not need to be rebooted.

Minor release switching allows the administrator to select one of the CLE software versions that are installed within a single system set and have the same base operating system release (e.g., switching from 5.2UP01 back to 5.2UP00). Switching is achieved by modifying sets of symbolic links in the file system to refer to the requested release.

Major release switching requires a separate set of disk partitions for each major operating system (e.g., switching from 4.2UP02 to 5.1UP00). Each system set provides a complete set of all file system and boot images, thus making it possible to switch easily between two or more different versions of the CLE system software. Each system set can be an alternative location for an installation or upgrade of the Cray system. System sets are defined in the `/etc/sysset.conf` file on the SMW.

If multiple versions of the software are installed and no version is chosen, the most recently installed is used.

Invoke a Minor Release Switch Within a System Set

The `xtrelswitch` command performs release switching by manipulating symbolic links in the file system and by setting the default version of modulefiles that are loaded at login. `xtrelswitch` uses a release version that is provided either in the `/etc/opt/cray/release/xtrelease` file or by the `xtrel` boot parameter. If the latter is not provided, the former is used. The `xtrelswitch` command is not intended to be invoked interactively; rather it is called by other scripts as part of the boot sequence. Specifically, when the boot node is booted, this command is invoked to switch the components in the boot node and shared root file systems. The `xtrelswitch` command does not support switching between major release levels, for example from CLE 4.0 to CLE 5.0.

To accomplish minor release switching, set the `bootimage_xtrel` parameter to `yes` in the `CLEinstall.conf` installation configuration file. This will include the release version in the boot image parameters file. If switching between minor levels is routine, it may be more convenient to use a *bootimage* in `/bootimagedir` (the boot image must be in the same path for both the SMW and the boot root), instead of the updating the `BOOT_IMAGE` disk partition.

For additional information, see the `xtrelswitch(8)` man page.

Invoke a Major Release Switch Using Separate System Sets

When system sets are used to change the Cray software booted on a Cray system, an entirely different file system is booted. The switched components include:

- The boot node root file system
- The shared-root file system
- The disk partition containing the SDB
- The `syslog`, `ufs`, and persistent `/var` file systems

Booting a system set requires:

- The `/etc/sysset.conf` file that describes the available system sets.
- Choosing which boot image will be used for the next boot. Each system set label has at least one `BOOT_IMAGE`.
- Activating a boot image for the chosen system set label.

The `CLEinstall` program installs or upgrades a system set to a set of disk partitions on the Boot RAID. For more information about the `CLEinstall` program and the `/etc/sysset.conf` file, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)* and the `sysset.conf(5)` man page.

Boot a System Set

1. Choose which system set in the `/etc/sysset.conf` file is to be used for the boot. For example:

```
LABEL:BLUE
DESCRIPTION:BLUE system with production
```

2. For the chosen system set, there is at least one `BOOT_IMAGE` in the `/etc/sysset.conf` file. Look at the `/etc/sysset.conf` file to determine which boot image is associated with which raw device. For this example, get the `SMWdevice` entry for `BOOT_IMAGE0` for the chosen system set:

```
# function      SMWdevice      host      hostdevice      mountpoint      shared
BOOT_IMAGE0    /dev/disk/by-id/scsi-3600a0b800026e1400000192c4b66eb70-part2
boot /dev/disk/by-id/scsi-3600a0b800026e1400000192c4b66eb70-part2 /raw0 no
```

3. Set the next boot to use the boot image `BOOT_IMAGE0` from the `BLUE` system set, which is the `/dev/disk/by-id/scsi-3600a0b800026e1400000192c4b66eb70-part2` disk partition. There will be a link from `/raw0` to `/dev/disk/by-id/scsi-3600a0b800026e1400000192c4b66eb70-part2`.

```
smw:~ # xtcli part_cfg update pN -i /raw0
```

Or, if working with a partitioned system, `pN`:

```
smw:~ # xtcli part_cfg update pN -i /raw0
```

Change the Service Database (SDB)

The SDB, which is a MySQL database, contains the XTAdmin system database. The XTAdmin database contains both persistent and nonpersistent tables. The `processor` and `service_processor` tables are nonpersistent and are created from the HSS data at boot time. The XTAdmin database tables track system configuration information. The SDB makes the system configuration information available to the Application Level Placement Scheduler (ALPS), which interacts with individual compute nodes running CNL.

Cray provides commands (see [Update SDB Tables](#) on page 147) to examine values in the SDB tables and update them when the system configuration changes.



CAUTION: Do not use MySQL commands to change table values directly. Doing so can leave the database in an inconsistent state.

Accounts that access MySQL by default contain a `.my.cnf` file in their home directories.

Service Database Tables

The SDB tables described belong to the XTAdmin database.

Table 6. Service Database Tables

Table Name	Function
<code>attributes</code>	Stores compute node attribute information
<code>lustre_failover</code>	Updates the database when a node's Lustre failover configuration changes
<code>lustre_service</code>	Updates the database when a node's Lustre service configuration changes
<code>filesystem</code>	Updates the database when a Lustre file system's configuration changes
<code>gpus</code>	Stores accelerator module (GPU) information
<code>processor</code>	Stores master list of processing elements and their status
<code>segment</code>	For nodes with multiple NUMA nodes, stores attribute information about the compute node and its associated NUMA nodes
<code>service_cmd</code>	Stores characteristics of a service
<code>service_config</code>	Stores processing element services that the resiliency communication agent (RCA) starts
<code>service_processor</code>	Stores nodes and classes (boot, login, server, I/O, or network)
<code>version</code>	Stores the database schema version

View the SDB with MySQL Commands

SDB is configured as part of the system installation, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*. Follow this procedure to examine the SDB.



CAUTION: Use MySQL commands to examine tables, but do not use them to change table values directly. Doing so can leave the database in an inconsistent state.

1. Log on to the SDB node as user `crayadm`.
2. Enter the MySQL shell.

```
crayadm@sdb:~> mysql -u basic -p
Enter password: *****
mysql> show databases;
+-----+
| Database |
+-----+
| XTAdmin  |
+-----+
1 row in set (0.04 sec)
```

3. Select the `XTAdmin` database.

```
mysql> use XTAdmin;
Database changed
```

4. Display the tables in the `XTAdmin` database.

```
mysql> show tables;
+-----+
| Tables_in_XTAdmin |
+-----+
| attributes         |
| filesystem         |
| gpus               |
| lustre_failover   |
| lustre_service    |
| processor          |
| segment           |
| service_cmd       |
| service_config    |
| service_processor |
| version           |
+-----+
10 rows in set (0.00 sec)
```

5. Display the format of the `service_processor` table.

```
mysql> describe service_processor;
+-----+-----+-----+-----+-----+-----+
| Field          | Type                | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| processor_id  | int(10) unsigned   |      | PRI | 0        |       |
| service_type  | varchar(64)        | YES  |     | NULL     |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

6. Display the contents of all fields in the `service_processor` table.

```
mysql> select * from XTAdmin.service_processor;
+-----+-----+-----+-----+-----+-----+
|
```

```

| processor_id | service_type |
+-----+-----+
|          0 | service      |
|          3 | service      |
|          4 | service      |
|          7 | service      |
|          8 | service      |
|         11 | service      |
|         12 | service      |
|         15 | service      |
|         16 | service      |
|         19 | service      |
|         20 | service      |
|         23 | service      |
|         24 | service      |
|         27 | service      |
+-----+-----+
14 rows in set (0.00 sec)

```

7. Display `processor_id` values from the `processor` table.

```

mysql> select processor_id from processor;
+-----+
| processor_id |
+-----+
|          0 |
|          3 |
|          4 |
|          7 |
|          8 |
|         103 |
|         104 |
|         107 |
|
|         ...
|         192 |
|         195 |
+-----+
162 rows in set (0.00 sec)

```

8. Exit the MySQL shell.

```
mysql> quit
```

Update SDB Tables

The CLE command pairs shown enable the system administrator to update tables in the SDB. One command converts the data into an ASCII text file to edit; the other writes the data back into the database file.

Table 7. Service Database Update Commands

Get Command	Put Command	Table Accessed	Reason to Use	Default File
<code>xtdb2proc</code>	<code>xtproc2db</code>	<code>processor</code>	Updates the database when a	<code>./processor</code>

Get Command	Put Command	Table Accessed	Reason to Use	Default File
			node is taken out of service	
<code>xtdb2attr</code>	<code>xtattr2db</code>	<code>attributes</code>	Updates the database when node attributes change	<code>./attribute</code>
<code>xtdb2nodeclasses</code>	<code>xtnodeclasses2db</code>	<code>service_processor</code>	Updates the database when a node's class changes	<code>./node_classes</code>
<code>xtdb2segment</code>	<code>xtsegment2db</code>	<code>segment</code>	For nodes with multiple NUMA nodes, updates the database when attribute information about node changes	<code>./segment</code>
<code>xtdb2servcmd</code>	<code>xtservcmd2db</code>	<code>service_cmd</code>	Updates the database when characteristics of a service change	<code>./serv_cmd</code>
<code>xtdb2servconfig</code>	<code>xtservconfig2db</code>	<code>service_config</code>	Updates the database when services change	<code>./serv_config</code>
<code>xtdb2etchosts</code>	<code>none</code>	<code>processor</code>	Manages IP mapping for service nodes	<code>none</code>
<code>xtdb2lustrefailover</code>	<code>xtlustrefailover2db</code>	<code>lustre_failover</code>	Updates the database when a node's Lustre failover state changes	<code>./lustre_failover</code>
<code>xtdb2lustreserv</code>	<code>xtlustreserv2db</code>	<code>lustre_service</code>	Updates the database when a file system's failover process is changed	<code>./lustre_serv</code>
<code>xtdb2filesystem</code>	<code>xtfilesystem2db</code>	<code>filesystem</code>	Updates the database when a file system's status changes	<code>./filesystem</code>
<code>xtdb2gpus</code>	<code>xtgpus2db</code>	<code>gpus</code>	Updates the database when attributes about the accelerators change	<code>./gpus</code>
<code>xtprocadmin</code>	<code>none</code>	<code>processor</code>	Displays or sets the current value of processor flags and	<code>none</code>

Get Command	Put Command	Table Accessed	Reason to Use	Default File
			node attributes in the service database (SDB). The batch scheduler and ALPS are impacted by changes to these flags and attributes.	
<code>xtservconfig</code>	<code>none</code>	<code>service_config</code>	Adds, removes, or modifies service configuration in the SDB <code>service_config</code> table	<code>none</code>

Change Nodes and Classes

The `service_processor` table tracks node IDs (NIDs) and their classes. The table is populated from the `/etc/opt/cray/sdb/node_classes` file on the boot node every time the system boots. Change this file to update the database when the classes of nodes change, for example, when adding login nodes. If changes are made to `/etc/opt/cray/sdb/node_classes`, the same changes must be made to the node class settings in `CLEinstall.conf` before performing an update or upgrade installation; otherwise, the install utility will complain about the inconsistency.

The `xtnodeclasses2db` command inserts the node-class list into the database. It does not make any changes to the shared root. To change the shared root, invoke the `xtnce` command.

For more information, see the `xtdb2nodeclasses(8)`, `xtnodeclasses2db(8)`, and `xtnce(8)` man pages.

Update the SDB when Services Change

The `service_config` table of the SDB maintains a list of the services to be configured on service nodes. Update this table when services are changed, for example, when adding the `PBS-MOM` service.

For more information, see the `xtservconfig(8)` man page.

1. Use the `xtservconfig` command to determine the services that are available in the `service_config` table. The `xtservconfig` command can be executed from any service node but is normally run from the boot node.

```
boot:~ # xtservconfig avail
SERVICE-COMMAND START STOP
SERVICE-COMMAND START STOP
NTP /opt/cray/rca/default/etc/fom+ /opt/cray/rca/default/etc/fom+
PBS-MOM /opt/cray/rca/default/etc/fom+ /opt/cray/rca/default/etc/fom+
PBS-SCHED /opt/cray/rca/default/etc/fom+ /opt/cray/rca/default/etc/fom+
PBS-SERV /opt/cray/rca/default/etc/fom+ /opt/cray/rca/default/etc/fom+
```

2. Use the `xtservconfig` command to modify the services that run on each node. `root` user privileges are required to make a change using the `xtservconfig` command. For this example, add the `PBS-MOM` service:

```
boot:~ # xtservconfig -n 12 add PBS-MOM
```

3. Reboot the node or send a `SIGHUP` signal on the affected node to activate the change.

- a. Log on to the affected node as root user.

```
boot:~ # ssh root@nid00012
```

- b. Send a `SIGHUP` signal.

```
nid00012:~ # killall -HUP fomd
```

This causes the failover manager to read the database.

Export Lustre with NFSv3

Cray supports exporting a direct-attached Lustre file system with NFSv3 (NFSv4 support is deferred at this time). This feature allows hosts that are external to the Cray system to access direct-attached Lustre file systems that would otherwise not be available and it provides access to data stored on a Lustre file system to hosts that do not support a Lustre client. A service node Lustre client (with external connectivity such as Ethernet or InfiniBand™) can mount the Lustre file system and export that mounted file system via standard NFS methods.

1. Enter `xtopview` for the node view of the service node Lustre client.
For this example, the service node is NID 8:

```
boot:~ # xtopview -n nid
```

2. Check the file specialization of the `/etc/exports` file.

```
node/8:/ # xthowspec /etc/exports
node/8:/etc/exports:node
```

3. Optional: Specialize `/etc/exports` if it is not already specialized.

```
node/8:/ # xtspec -n nid /etc/exports
```

4. Edit the `/etc/exports` file and add an entry for the mounted Lustre file system to be exported via NFS. Specifying `insecure` gives the best interoperability with the universe of potential client systems. The `fsid=value` setting is required to successfully export a Lustre file system. This value is a 32-bit integer assigned by the administrator that identifies the file system to NFS. For more information on NFS mount options, see the `mount(8)` and `nfs(5)` man pages.

To identify the mounted Lustre file systems on a node use the following command:

```
crayadm@nid:~> mount -t lustre
23@gni1:/lus_hera on /lus/nid00023 type lustre (rw,relatime,flock)
```

```
node/8:/ # vi /etc/exports
```

```
/lus/nid00023 *(rw,insecure,no_root_squash,no_subtree_check,fsid=value)
```

- Repeat steps 2 on page 150 through 3 on page 150 for the `/etc/sysconfig/nfs` file so that it is node specialized.
- Edit the `/etc/sysconfig/nfs` file and change the following parameter to disable NFSv4.

IMPORTANT: NFSv4 is not supported for exporting Lustre at this time.

```
node/8:/ # vi /etc/sysconfig/nfs
```

```
# Enable NFSv4 support (yes/no)
#
NFS4_SUPPORT="no"
```

- Configure the `nfsserver` and `rpcbind` services to start at boot with the `chkconfig` command.

```
node/8:/ # chkconfig nfsserver on
nfsserver on
nid:/ # chkconfig rpcbind on
rpcbind on
```

- Restart the `nfsserver` and `rpcbind` services to pick up the configuration changes made earlier.

```
node/8:/ # service nfsserver restart
Shutting down kernel based NFS server: nfsd           done
Starting kernel based NFS server: mountd statd nfsd sm-notify done
node/8:/ # service rpcbind restart
Shutting down rpcbind                               done
Starting rpcbind                                     done
```

- Exit `xtopview`.

Configure the NFS client to Mount the Exported Lustre File System

Depending on the site's client system, the configuration may be different. This procedure contains general information that will help configure the client system to properly mount the exported Lustre file system. Consult the client system documentation for specific configuration instructions.

- As `root`, verify that the `nfs` client service is started at boot.
- Add a line to the `/etc/fstab` file to mount the exported file system. The list below describes various recommended file system mount options. For more information on NFS mount options, see the `mount(8)` and `nfs(5)` man pages.

```
server@network:/filesystem /client/mount/point lustre file_system_options 0 0
```

Recommended file system mount options:

```
rsize=1048576,wsiz=1048576
```

Set the read and write buffer sizes from the server at 1MiB. These options match the NFS read/write transaction to the Lustre filesystem block size, which reduces cache/buffer thrashing on the service node providing the NFS server functionality.

```
soft,intr
```

Use a soft interruptible mount request.

async	Use asynchronous NFS I/O. Once the NFS server has acknowledged receipt of an operation, let the NFS client move along even though the physical write to disk on the NFS server has not been confirmed. For sites that need end-to-end write-commit validation, set this option to <code>sync</code> instead.
proto=tcp	Force use of TCP transport; this makes the larger <code>rsize/wsize</code> operations more efficient. This option reduces the potential for UDP retransmit occurrences, which improves end-to-end performance.
relatime,timeo=600,local_lock=none	Lock and time stamp handling, transaction timeout at 10 minutes.
nfsvers=3	Use NFSv3 specifically. NFSv4 is not supported at this time.

3. Mount the file system manually or reboot the client to verify that it mounts correctly at boot.

Enable File-locking for Lustre Clients

To enable file-locking for all Linux clients when mounting the Lustre file system on service nodes or on compute nodes, use the `flock` option for mount. For more information, see the Lustre documentation at <http://wiki.whamcloud.com/display/PUB/Documentation>.

Sample mount line from compute node `/etc/fstab`

```
4@gni:136@gni:/filesystem /lus/nid00004 lustre rw,flock 0 0
```

Back Up and Restore Lustre Failover Tables

To minimize the potential impact of an event that creates data corruption in the SDB database, Cray recommends creating a manual backup of the Lustre tables that can be restored after a reinitialization of the SDB database.

Manually Back Up Lustre Failover Tables

1. Log on to the boot node as `root`.

2. Back up the `lustre_service` table.

```
boot# mysqldump -h sdb XTAdmin lustre_service > /var/tmp/lustre_service.sql
```

3. Back up the `lustre_failover` table.

```
boot# mysqldump -h sdb XTAdmin lustre_failover > /var/tmp/lustre_failover.sql
```

Manually Restore Lustre Failover Tables

4. Log on to the boot node as `root`.
5. After the SDB database is recreated, restore the `lustre_service` table.

```
boot# mysqldump -h sdb XTAdmin < /var/tmp/lustre_service.sql
```

6. Restore the `lustre_failover` table.

```
boot# mysqldump -h sdb XTAdmin < /var/tmp/lustre_failover.sql
```

Node Attributes

Users can control the selection of the compute nodes on which to run their applications and can select nodes on the basis of desired characteristics (*node attributes*). This allows a placement scheduler to schedule jobs based on the node attributes.

A user invokes the `cselect` command to specify node-selection criteria. The `cselect` script uses these selection criteria to query the table of node attributes in the SDB and returns a node list to the user based on the results of the query. When launching the application, the user includes the node list using the `aprun -L node_list` option as described on the `aprun(1)` man page. The ALPS placement scheduler allocates nodes based on this list.

To meet specific user needs, the administrator can modify the `cselect` script. For additional information about the `cselect` script, see the `cselect(1)` man page.

Initially Set Node Attributes

In order for users to select desired node attributes, the system administrator must first set the characteristics of individual compute nodes. Node attribute information is written to the `/etc/opt/cray/sdb/attributes` data file and loaded into the `attributes` table in the SDB when the SDB is booted.

Generate the `/etc/opt/cray/sdb/attributes` File

Data for the `/etc/opt/cray/sdb/attributes` file comes from two other files: the `/etc/opt/cray/sdb/attr.xthwinv.xml` file, which contains information to generate the hardware attributes for each node, and the `/etc/opt/cray/sdb/attr.defaults` file, which allows administrators to set values for specific nodes (or all nodes if a `DEFAULT` is specified). The `xtprocdadmin(8)` man page includes a description of the attributes fields used by these two files.

`/etc/opt/cray/sdb/attr.xthwinv.xml` This file is created by `CLEinstall` and automatically regenerated by `xtbootsys` at each boot through the `xthwinv -x` command.

To manually generate the `/etc/opt/cray/sdb/attr.xthwinv.xml` file, invoke the `xthwinv -x` command on the System Management Workstation (SMW) through the boot node, redirecting the output to the `/etc/opt/cray/sdb/attr.xthwinv.xml` file on the boot node; for example:

```
boot:~ # ssh smw xthwinv -x s0 > /etc/opt/cray/sdb/
attr.xthwinv.xml
```

For additional information, see the `xthwinv(8)` man page.

IMPORTANT: If there are blades powered down when a software upgrade is scheduled, see the `CLEinstall(8)` man page for instructions on using the `--xthwinvxmlfile` option during the upgrade process.

`/etc/opt/cray/sdb/attr.defaults`

This file can be used to set an attribute value on any node, but it is primarily used for assigning labels to nodes.

```
label0
label1
label2
label3
```

Each label is a string of up to 32 characters; the string cannot contain any spaces or shell-sensitive characters. These labels can be applied to all nodes or to a given set of nodes.



CAUTION: Do not attempt to set hardware attributes (memory size, clock speed, and cores) in the `attr.defaults` file because the values will be overwritten by those already specified in the `/etc/opt/cray/sdb/attr.xthwinv.xml` file.

To create the `attr.defaults` file, copy the example file provided in `/opt/cray/sdb/default/etc/attr.defaults.example`. Edit the file to modify the existing attribute settings and create site-specific attributes as needed. If `CLEinstall` ran previously, `attr.defaults` was already copied and exists in that location.

In addition to the attributes in the `/etc/opt/cray/sdb/attr.defaults` file, there are two keywords that allow the administrator to describe the node or set of nodes to which attributes are assigned. For global default-attribute values that apply to the entire system, the line that specifies an attribute must begin with the `DEFAULT:` keyword. For example:

```
DEFAULT:  osclass=2
```

The `nodeid` keyword assigns attributes to a specific node or set of nodes and overrides a default setting. For values that apply only to certain nodes, the line that specifies the attributes must begin with `nodeid=[RANGE]`, where `RANGE` is a comma-separated list of nodes and ranges that have the form `m, n` or `m-n`. For example:

```
nodeid=234,245-248 label3='GREEN'
```

Use node attribute labels to assign nodes to user groups

For this example, uses labels to assign groups of compute nodes to specific user groups without the need to partition the system:

```
nodeid=101-500 label0=physicsdept
nodeid=501-1000 label1=csdept
nodeid=50-100,1001 label2=biologydept
```

Display the Format of the SDB attributes Table

When the SDB boots, it reads the `/etc/opt/cray/sdb/attributes` file and loads it into the SDB `attributes` table.

To display the format of the `attributes` SDB table, use the `mysql` command:

```
crayadm@login:~> mysql -e "desc attributes;" -h sdb XTAdmin
+-----+-----+-----+-----+-----+-----+
| Field      | Type                | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| nodeid     | int(32) unsigned    | NO   | PRI | 0        |       |
| archtype   | int(4) unsigned     | NO   |     | 2        |       |
| osclass    | int(4) unsigned     | NO   |     | 2        |       |
| coremask   | int(4) unsigned     | NO   |     | 1        |       |
| availmem   | int(32) unsigned    | NO   |     | 0        |       |
| pagesz12   | int(32) unsigned    | NO   |     | 12       |       |
| clockmhz   | int(32) unsigned    | YES  |     | NULL     |       |
| label0     | varchar(32)         | YES  |     | NULL     |       |
| label1     | varchar(32)         | YES  |     | NULL     |       |
| label2     | varchar(32)         | YES  |     | NULL     |       |
| label3     | varchar(32)         | YES  |     | NULL     |       |
| numcores   | int(4) unsigned     | NO   |     | 1        |       |
| sockets    | int(4) unsigned     | NO   |     | 1        |       |
| dies       | int(4) unsigned     | NO   |     | 1        |       |
+-----+-----+-----+-----+-----+-----+
```

The service database command pair `xtdb2attr` and `xtattr2db` enables the system administrator to update the `attributes` table in the SDB. For additional information about updating SDB tables using command pairs, see [Update SDB Tables](#) on page 147.

View and Temporarily Set Node Attributes

Use the `xtprocadm` command to view current node attributes. The `xtprocadm -A` option lists all attributes of selected nodes. The `xtprocadm -a attr1,attr2` option lists selected attributes of selected nodes.

An administrator can use the `xtprocadm -a attr=value` command to temporarily set certain site-specific attributes. Using the `xtprocadm -a attr=value` command to set certain site-specific attributes is not persistent across reboots. Attribute settings that are intended to be persistent across reboots (such as labels) must be specified in the `attr.defaults` file.

NOTE: For compute nodes, `xtprocadm` changes to attributes require restarting the `apbridge` daemon on the boot node in order for ALPS to detect changes that the `xtprocadm` command has made to the SDB. Restarting the other ALPS components (for example, on the SDB node or on the login node if they

are separate nodes) is not necessary. To restart `apbridge`, log into the boot node as `root` and execute the following command:

```
boot:~ # /etc/init.d/alps restart
```

For example, the following command creates a new `label1` attribute value for the compute node whose NID is 350. The `xtprocadmin` command must be executed by `root` from a service node and the SDB must be running:

```
boot:~ # xtprocadmin -n 350 -a label1=eedept
```

Connected				
NID	(HEX)	NODENAME	TYPE	LABEL1
350	0x15e	c1-0c1s0n0	compute	eedept

Then restart the `apbridge` daemon on the boot node in order for ALPS to detect changes that the `xtprocadmin` command has made to the SDB.

```
boot:~ # /etc/init.d/alps restart
```

The XTAdmin Database segment Table

The XTAdmin database contains a `segment` table that supports the memory affinity optimization tools for applications and CPU affinity options for all Cray compute nodes. The CPU affinity options apply to all Cray multicore compute nodes.

The `segment` table is similar to the `attributes` table but differs in that a node may have multiple segments associated with it; the `attributes` table provides summary information for each node.

In order to address the application launch and placement requirements for compute nodes with two or more NUMA nodes, the Application Level Placement Scheduler (ALPS) requires additional information that characterizes the intranode topology of the system. This data is stored in the `segment` table of the XTAdmin database and acquired by `apbridge` when ALPS is started, in much the same way that node attribute data is acquired.

The `segment` table contains the following fields:

- node_id** The node identifier that maps to the `nodeid` field of the `attributes` table and `processor_id` field of the `processor` table.
- socket_id** Contains a unique ordinal for each processor socket.
- die_id** Contains a unique ordinal for each processor die; with this release, `die_id` is 0 in the `segment` table and is otherwise unused (reserved for future use).
- numcores** The number of integer cores per node; in systems with accelerators this only applies to the host processor (CPU).
- coremask** The processor core mask. The `coremask` has a bit set for each core of a CPU. 24-core nodes will have a value of 16777215 (hex 0xFFFFFFFF).
`coremask` is deprecated and will be removed in a future release.
- mempgs** Represents the amount of memory available, in Megabytes, to a single segment.

The `/etc/sysconfig/xt` file contains `SDBSEG` field, which specifies the location of the `segment` table file; by default, `SDBSEG=/etc/opt/cray/sdb/segment`.

To update the `segment` table, use the following service database commands:

- `xtdb2segment`, which converts the data into an ASCII text file that can be edited
- `xtsegment2db`, which writes the data back into the database file

For more information, see the `xtdb2segment(8)` and `xtsegment2db(8)` man pages.

After manually updating the `segment` table, log on to any login node or the SDB node as root and execute the `apmgr resync` command to request that ALPS reevaluate the configuration node segment information and update its information.

If ALPS or any portion of the feature fails in relation to segment scheduling, ALPS reverts to the standard scheduling procedure.

Network File System (NFS)

The Network File System (NFS) version 4 distributed file system protocol is supported. NFS is enabled by default on the `boot`, `sdb`, and `ufs` service nodes but is not enabled on compute nodes. Support for NFSv4 is included as part of the SLES software.

The CLE installation tool supports NFS tuning via `/etc/sysconfig/nfs` and `/etc/init.d/nfsserver` on the boot node. The `nfs_mountd_num_threads` and `use_kernel_nfsd_number` parameters in the `CLEinstall.conf` installation configuration file control an NFS `mountd` tuning parameter that is added to `/etc/sysconfig/nfs` and used by `/etc/init.d/nfsserver` to configure the number of `mountd` threads on the boot node. By default, NFS `mountd` behavior is a single thread. For a larger Cray system (greater than 50 service I/O nodes), contact a Cray service representative for assistance changing the default setting.

To enable the `nfsserver` service on all service nodes, set the `CLEinstall.conf nfsserver` parameter to `yes`. The default setting is `no`.

Configure Ethernet Link Aggregation (Bonding, Channel Bonding)

Linux Ethernet link aggregation is generally used to increase aggregate bandwidth by combining multiple Ethernet channels into a single virtual channel. Bonding can also be used to increase the availability of a link by utilizing other interfaces in the bond when one of the links in that bond fails.

For more information, see the Linux documentation file `/usr/src/linux/Documentation/networking/bonding.txt` installed on the system.

This procedure configures an I/O service node bonding interface.

1. On the boot node, run the `xtopview` command for the node that needs the bonding interface configured. For example, to access node 2, type the following:

```
boot:~ # xtopview -n 2
```

2. Create and specialize the following files:

- /etc/sysconfig/network/ifcfg-bond0
- /etc/sysconfig/network/ifcfg-eth0
- /etc/sysconfig/network/ifcfg-eth1

```
node/2:/ #touch /etc/sysconfig/network/ifcfg-bond0
node/2:/ #xtspec -n 2 /etc/sysconfig/network/ifcfg-bond0
node/2:/ #touch /etc/sysconfig/network/ifcfg-eth0
node/2:/ #xtspec -n 2 /etc/sysconfig/network/ifcfg-eth0
node/2:/ #touch /etc/sysconfig/network/ifcfg-eth1
node/2:/ #xtspec -n 2 /etc/sysconfig/network/ifcfg-eth1
```

3. Edit the previously created files to include the site-specific network settings.

```
node/2:/ #vi /etc/sysconfig/network/ifcfg-bond0
```

```
BOOTPROTO="static"
BROADCAST="10.0.2.255"
IPADDR="10.0.2.10"
NETMASK="255.255.0.0"
NETWORK="10.0.2.0"
REMOTE_IPADDR=""
STARTMODE="onboot"
BONDING_MASTER="yes"
BONDING_MODULE_OPTS="mode=active-backup primary=eth1"
BONDING_SLAVE0="eth0"
BONDING_SLAVE1="eth1"
```

```
node/2:/ #vi /etc/sysconfig/network/ifcfg-eth0
```

```
BOOTPROTO='static'
STARTMODE='onboot'
MASTER=bond0
SLAVE=yes
REMOTE_IPADDR=''
IPV6INIT=no
```

```
node/2:/ #vi /etc/sysconfig/network/ifcfg-eth1
```

```
BOOTPROTO='static'
STARTMODE='onboot'
MASTER=bond0
SLAVE=yes
REMOTE_IPADDR=''
IPV6INIT=no
```

4. Exit from xtopview

Configure a Virtual Local Area Network (VLAN) Interface

This procedure configures an 802.1Q standard VLAN.

1. On the boot node, run the `xtopview` command for the node that needs the VLAN configured. For example, to access node 2, type the following:

```
boot:~ # xtopview -n 2
```

2. Create and specialize a file named `/etc/sysconfig/network/ifcfg-vlanN`, where *N* is the VLAN ID. This example creates and specializes the `vlan2` file:

```
node/2:/ # touch /etc/sysconfig/network/ifcfg-vlan2
node/2:/ # xtspec -n 2 /etc/sysconfig/network/ifcfg-vlan2
```

3. Edit the `/etc/sysconfig/network/ifcfg-vlanN` file to include the usual network settings. It must also include variable `ETHERDEVICE`, which provides the real interface for the VLAN. The real interface will be set up automatically; it does not require a configuration file. For additional information, see the `ifcfg-vlan(5)` man page.

This example sets up `vlan2` on top of `eth0`:

```
ifcfg-vlan2
STARTMODE=onboot
ETHERDEVICE=eth0
IPADDR=192.168.3.27/24
```

An interface named `vlan2` will be created when the system boots.

4. Exit from `xtopview`.

Configure Realm-specific IP Addressing (RSIP)

Realm-Specific Internet Protocol (RSIP) enables internal client nodes, such as compute nodes, to reach external IP networking resources. Support for RSIP is available with CLE on systems that have CNL compute nodes. RSIP for IPv4 TCP and User Datagram Protocol (UDP) transport protocols are supported. Internet Protocol Security (IPSec) and IPv6 protocols are not supported.

RSIP is composed of two main components: RSIP clients and RSIP servers or gateways. The system administrator configures RSIP and select servers using RSIP parameters in `CLEinstall.conf`. By default, when RSIP is enabled, all CNL compute nodes are configured to be RSIP clients.

On a Cray system, RSIP servers must be service nodes with an external IP interface such as a 10-GbE network interface card (NIC). Multiple RSIP servers can be configured using multiple service nodes, but only one RSIP daemon (`rsipd`) and one external interface are allowed per service node. Cray requires configuring RSIP servers as dedicated network nodes.



CAUTION: Cray recommends not configuring login nodes or service nodes that provide Lustre or batch services as RSIP servers. Failure to set up an RSIP server as a dedicated network node can disrupt network functionality. That said, this may not be possible on small systems that cannot afford to dedicate a whole node as a login node. In this case, modifications are provided in [Configure a Login Node as an RSIP Server](#) on page 163.

The performance impact of configuring RSIP is negligible; very little noise is generated by the RSIP client. RSIP clients will issue a lease refresh message request/response pair once an hour, at a rate of 10 clients per second, but otherwise are largely silent.

To configure RSIP for a Cray system, first determine which service nodes and associated Ethernet devices will be used to provide RSIP services. Optionally, determine if service nodes with no external IP interfaces (isolated service nodes) will be configured to act as RSIP clients. In this case, modified steps are provided to configure a login node as a RSIP server.

Enhancements to the default RSIP configuration require a detailed analysis of specific site configuration and requirements. Contact a Cray representative for assistance in changing the default RSIP configuration.

The following terms are used throughout the RSIP procedures and are important to understand:

External (externally routable)	In this context, network space outside of the Cray system.
Internal (internally routable)	In this context, network space inside of the Cray system.
Gateway	A node running the RSIP daemon that acts as the router between the external and internal network space. A gateway node must have an external facing NIC and an internal facing NIC.

Install and Configure RSIP Using CLEinstall

The CLEinstall program can be configured to automatically install RSIP either during a system software upgrade or as a separate event. In either case, it is necessary to update the compute node boot image and restart the Cray system before RSIP is functional.

When the following RSIP-specific parameters are set in the CLEinstall.conf file, CLEinstall will load the RSIP RPM, modify rsipd.conf and invoke the appropriate xtrsipcfg commands to configure RSIP for the system.

rsip_nodes A space-separated list that specifies the NIDs of nodes identified as RSIP servers.

rsip_interfaces A space-separated list that specifies the IP interface for each RSIP server node. List the interfaces in the same order specified in the `rsip_nodes` parameter.

rsip_servicenode_clients A space-separated integer list that specifies service nodes defined as RSIP clients.



CAUTION: Do not configure service nodes with external network connections as RSIP clients. Configuring a network node as an RSIP client will disrupt network functionality. Service nodes with external network connections will route all non-local traffic into the RSIP tunnel and IP may not function as desired.

CNL_rsip=yes Enable the RSIP client on CNL compute nodes. Optionally, edit the `/var/opt/cray/install/shell_bootimage_LABEL.sh` script and set `CNL_RSIP=y`.

If configuring RSIP for the first time during an installation or upgrade of the CLE system software, follow RSIP-specific instructions in *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*. If configuring RSIP as a separate event, follow [Install, Configure, and Start RSIP Clients and Servers](#) to complete a default RSIP configuration and setup. To add isolated service nodes as RSIP clients to an already configured RSIP service, follow [Add Isolated Service Nodes as RSIP Clients](#).

For additional information about configuring RSIP, see the `xtrsipcfg(8)`, `rsipd(8)`, and `rsipd.conf(5)` man pages.

Install, Configure, and Start RSIP Clients and Servers

1. Edit `CLEinstall.conf` and add RSIP configuration details.

For this example, configure nodes 16 and 20 as RSIP servers with an external interface named `eth0`, node 64 as an RSIP server with an external interface named `eth1`, and node 0 as a service node RSIP client:

```
smw:~ # vi /home/crayadm/install.xtrelease/CLEinstall.conf
```

```
rsip_nodes=16 20 64
rsip_interfaces=eth0 eth0 eth1
rsip_servicenode_clients=0
CNL_rsip=yes
```

2. Invoke `CLEinstall` on the SMW specifying the `xtrelease` that is currently installed on the system set being used and located in the `CLEmedia` directory.

```
smw:~ # /home/crayadm/install.xtrelease/CLEinstall --upgrade \
--label=system_set_label --XTrelease=xtrelease \
--configfile=/home/crayadm/install.xtrelease/CLEinstall.conf \
--CLEmedia=/home/crayadm/install.xtrelease
```

3. Enter `y` to proceed when prompted to update the boot root and again for the shared root.

```
*** Do you wish to continue? (y/n) --> y
```

Upon completion, `CLEinstall` lists suggested commands to finish the installation. Those commands are also described here. For more information about running the `CLEinstall` program, see *Installing and Configuring Cray Linux Environment (CLE) Software (S-2444)*.

4. Rebuild the boot image using `shell_bootimage_LABEL.sh`, `xtbootimg` and `xtcli` commands. Suggested commands are included in output from `CLEinstall` and `shell_bootimage_LABEL.sh`.
5. Run the `shell_post_install.sh` script on the SMW to unmount the boot root and shared root file systems and perform other cleanup as necessary.

```
smw:~# /var/opt/cray/install/shell_post_install.sh /bootroot0 /sharedroot0
```

6. Optional: If configuring a service node RSIP client, edit the boot automation file to start the RSIP client. On the isolated service node, invoke a `modprobe` of the `krsip` module with an IP argument pointing to the HSN IP address of an RSIP server node, and specify the number of ports requested.

For this example, the IP address of the RSIP server is `10.128.0.17`, the isolated service node is `nid00000`, and 32 ports are requested:

```
crayadm@smw:~> vi /opt/cray/etc/auto.xthostname
```

After the line or lines that start the RSIP servers, add:

```
# RSIP client startup
lappend actions { crms_exec_via_bootnode "nid00000" "root" "modprobe
krsip ip=10.128.0.17 use_rsip_local_ports=1 num_ports=32" }
```

7. Boot the Cray system.

```
crayadm@smw:~> xtbootsys -a auto.xthostname
```



CAUTION: RSIP clients on the compute nodes make connections to the RSIP server(s) during system boot. Initiation of these connections is staggered at a rate of 10 clients per second; during that time, connectivity over RSIP tunnels will be unreliable. Avoid using RSIP services for several minutes following a system boot; larger systems will require more time for connections to complete.

8. Test RSIP functionality by logging on to an RSIP client node (compute node) and pinging the IP address of the SMW or other host external to the Cray system.

For this example, `nid00074` is a compute node and `10.3.1.1` is a valid external IP address:

```
crayadm@smw:~> ssh root@nid00074
# ping 10.3.1.1
10.3.1.1 is alive!
```

Add Isolated Service Nodes as RSIP Clients

A system administrator can configure service nodes that are isolated from the network as RSIP clients. This procedure assumes that RSIP is already configured and functional on the Cray system. If RSIP is not installed and configured on the system, follow [Install, Configure, and Start RSIP Clients and Servers](#), which includes an optional step to configure isolated service nodes as RSIP clients.



WARNING: Do not configure service nodes with external network connections as RSIP clients. Configuring a network node as an RSIP client will disrupt network functionality. Service nodes with external network connections will route all non-local traffic into the RSIP tunnel and IP may not function as desired.

1. Select one RSIP server to provide access for the isolated service node. For this example, the RSIP server is `nid00016`.
2. Log on to the boot node and invoke `xtopview` in the node view for the selected RSIP server.

```
boot:~ # xtopview -n 16
```

3. Edit the `rsipd.conf` file and modify `max_clients` to add an additional client for each isolated service node being configured:
For this example, change the originally configured 300 RSIP clients (compute nodes) to 301.

```
node/16:/ # vi /etc/opt/cray/rsipd/rsipd.conf
```

```
max_clients 301
```

4. Load the RSIP client on the node. On the isolated service node, invoke a `modprobe` of the `krsip` module with an IP argument pointing to the HSN IP address of the RSIP server node selected in [1](#) on page 162. For this example, the IP address of the RSIP server is `10.128.0.17`, the isolated service node is `nid00023`, and 32 ports are requested:

```
boot:~ # ssh nid00023
nid00023:~ # modprobe krsip ip=10.128.0.17 use_rsip_local_ports=1 \
num_ports=32
```

5. Edit the boot automation file on the SMW to start the RSIP client. For this example, use the configuration values from the previous steps:

```
crayadm@smw:~> vi /opt/cray/etc/auto.xthostname
```

```
# RSIP client startup
lappend actions { crms_exec_via_bootnode "nid00023" "root" "modprobe krsip
ip=10.128.0.17 use_rsip_local_ports=1 num_ports=32" }
```

Configure a Login Node as an RSIP Server

Cray recommends not configuring login nodes as RSIP servers, because the RSIP server can interfere with IP traffic on the login node. It is understandable that sites with smaller systems may need a login node to do double duty; and for that reason, a workaround is provided. This **should not** be applied to service nodes that provide Lustre or batch services.

Follow the procedure to [Configure RSIP Gateway Address Pools](#) on page 163 with the following variances:

- Add a secondary IP address to `eth0` on the login node.
- Use this secondary IP address in the `rsipd.conf` file **and not** the login node's primary IP address.

Configure RSIP Gateway Address Pools

Prerequisites

- RSIP is installed and working (without pools) on the system
- Root privilege is available on both the gateway and the boot nodes (these can be the same)
- One or more unused, externally-addressable IP is available
- Ability to restart the RSIP daemon on the gateway
- Ability to reboot the system

There are multiple reasons for using RSIP gateway address pools, including:

- Relieving port exhaustion due to heavy traffic
- Addition of new nodes
- A dedicated network node is not available to use as the RSIP gateway

These instructions assume a single gateway, but it is possible to configure pools for multiple gateways. Each gateway has its own configuration file and its own pool of IP addresses. Simply repeat these directions for each gateway.

1. Edit a copy of the current `rsipd.conf` configuration file. A read-only copy is available on the SMW at `/etc/opt/cray/rsipd/rsipd.conf`.

```
smw:~# vi rsipd.conf
```

2. Modify the `pool` option.

This option defines the addresses the RSIP daemon can assign to the RSIP clients. These addresses must be externally routable. The `pool` option accepts an individual IP address or a dash-separated range of addresses (e.g., `198.51.100.1-198.51.100.10`). More than one instance of `pool` is allowed, making it possible to configure non-contiguous IPs in the pool.

IMPORTANT:

For this example, define two non-contiguous IP address ranges and one single IP address:

```
pool 198.51.100.1-198.51.100.10
pool 198.51.100.12-198.51.100.20
pool 198.51.100.250
```

There may already be a `pool` entry with the gateway's external address in the configuration file, but it is unlikely because that address will be added to the `pool` automatically if no other addresses are specified. Add `pool` entries for the site's external addresses. Depending on the site's use case, it may not be desirable to include the external NIC's primary IP.

3. Modify the `port_range` option.

This option sets which ports the gateway can assign to clients. If the gateway's external IP is in the `pool`, then ports used by the kernel cannot be used in the `port_range`. This is because the range specified will be used for all IPs in the pool. `port_range` is also specified as a dash-separated range, but unlike the `pool`, only one instance of `port_range` is allowed and, therefore, only one contiguous range is possible.

For this example, the range of ports assignable to clients is 8192-60000:

```
port_range 8192-60000
```

There should be a `port_range` entry already in the file, although it may be commented out. If the range listed is not correct for the system, change it to a range that is appropriate. If a change is made, the OS port range may also need to be moved. If uncertain how to do this, consult the documentation for installing RSIP. Upon start, the RSIP daemon will warn if the port range overlaps the kernel port range. Currently the test for this is somewhat limited and may still warn even if the NIC's primary IP is not in the pool. In that case, it is safe to ignore the warning.

4. Save the modified configuration file.
5. Log on to the boot node and invoke `xtopview`.

For a single gateway:

```
smw:~# ssh root@boot
boot:~# xtopview
```

For multiple gateways:

```
boot:~# xtopview -n node
```

- Copy the configuration file to the shared root.

The exact location on the shared root will vary, but one common location is `/.shared/base/default/etc/opt/cray/rsipd/`. If uncertain, contact the administrator that originally installed the site's RSIP instance. In a node-specific shared root (i.e., for multiple gateways), `/etc/opt/cray/rsipd/` should symbolically link to the correct location in most cases.

```
default:// # cp rsip.conf /.shared/base/default/etc/opt/cray/rsipd/
```

- Exit `xtopview` and the boot node.

```
default:// # exit
boot:~# exit
```

- Edit a copy of the current `krsip.conf` to change the number of ports requested by each client now that more ports available. This step is necessary because `xtrsipcfg` currently does not support address pools.

```
smw:~# vi krsip.conf
```

The format of entries in `krsip.conf` is:

```
nid server_ip alt_server_ip port_count delay
```

- Modify the `port_count` entries.

This column indicates the number of ports the client should request under RSAP-IP mode (the default and currently only supported mode for Cray RSIP). At the current time, Cray's RSIP implementation only supports one IP/port-range allocation per client node, therefore, it is desirable that this is as high a number as possible because more cannot be requested later. The current implementation allows for a maximum of only 255 ports if multiple assign requests are not configured. (For further information, see [Configure Multiple RSAP-IP Assign Requests](#).) When setting up RSIP without pools, this would normally be calculated by `xtrsipcfg`. To calculate the value for pools, use the following formula, where `ports_per_client` is the value used for `port_count`.

```
clients_per_server = clients / (servers * ips_per_server)
ports_per_client = port_range / clients_per_server
```

Note that `port_count` can be different for every node. This means that some nodes can have more ports than others, or clients connecting to one gateway can have a different amount of ports than a group connecting to a different gateway, depending on the size of the groups.

- Install `krsip.conf` as `compute/etc/krsip.conf` in the boot image. Follow site-specific procedures to build a new image and reboot the machine.

Currently, the `krsip` client cannot change its options without being restarted, and it cannot be unloaded. This means that the machine must be rebooted, and the new config file must be installed in the boot image.

- Add the pool addresses as secondary addresses to the external interface on the gateway node. This must be done for each address in the pool.

IMPORTANT: To define pools that are persistent across reboots, these entries can be added to the boot script.

The command has the following format, where *pool_ip* is an IP from the pool, *mask* is the network mask, *broadcast_addr* is the broadcast address on the interface, and *interface* is the name of the external interface.

```
ip addr add pool_ip/mask brd broadcast_addr dev interface
```

In this example, *broadcast_addr* is set to +, indicating that the broadcast address is automatically calculated based upon the netmask and IP being assigned.

```
# ip addr add 198.51.100.1/24 brd + dev eth0
```

12. If the number of ports requested was changed, then the system must be rebooted. Otherwise, simply restart the RSIP daemon on the gateway.

```
# /etc/init.d/rsipd restart
```

13. Wait a few minutes for all client nodes to get new registrations, then verify the registrations.

```
# /etc/init.d/rsipd dump
```

The RSIP registration table is displayed, showing nodes with registration for IPs in the pool.

RSIP Gateway Address Pool Example

Scenario: A machine with 2000 client nodes exists with only one service node available to use as the gateway. Configure 255 ports for each client node.

Assumptions:

- The gateway has a primary external IP of 198.51.100.1/24 on eth0
- Spare external IPs exist in the range 198.51.100.2–198.51.100.20
- The NICs primary IP will not be used in the pool (allowing the full port range of 0-65535)

Determine the number of IPs needed

Rearrange the `ports_per_client` equation:

```
ports_per_client = port_range / clients_per_server
```

to

```
ips_per_server = clients * ports_per_client / port_range / servers
```

Plug in the known values:

```
ips_per_server = 2000 * 255 / 65535 / 1 = 7.78
```

Round up to 8 IPs per server, which is less than the number of IPs available.

Modify the configuration files

In `rsipd.conf`:

```
pool 198.51.100.2-198.51.100.9
port_range 0-65535
```

And in `krsip.conf`

```
3 198.51.101.254 198.51.101.253 255 2
```

If `xtrsipcfg` is used to generate `krsip.conf`, then only the `port_count` field (255 in the example above) needs to change. Otherwise, the entire file must be created by hand, which is strongly discouraged.

Add secondary IPs to the gateway

Log on to the gateway as `root`.

```
# ip addr add 198.51.100.2/24 brd + dev eth0
# ip addr add 198.51.100.3/24 brd + dev eth0
# ip addr add 198.51.100.4/24 brd + dev eth0
# ip addr add 198.51.100.5/24 brd + dev eth0
# ip addr add 198.51.100.6/24 brd + dev eth0
```

Configure Multiple RSAP-IP Assign Requests

Prerequisites

- RSIP is installed on the Cray system and working with gateway address pools
- Root access is available on both the gateway and boot nodes (these can be the same)
- Sufficient IPs exist or can be added to the pool to support the desired number of ports per client
- Ability to restart the RSIP daemon on the gateway
- Ability to reboot the system

Why use multiple requests?

Due to a protocol limitation, each RSAP-IP mode assign request is limited to requesting 255 ports. The protocol, however, allows multiple assignments to the same client; therefore, if more than 255 ports per client are needed, multiple assign requests are needed.

On Cray systems, the `xtrsipcfg` utility is used to configure the RSIP gateway daemon. `xtrsipcfg` does not yet support configuring the RSIP client to make multiple assignment requests; therefore, sites wanting to use this feature must configure it manually.

Limitations

Currently the RSIP implementation only supports a single contiguous port range for a given client, even though this range might be made up of multiple allocations. This means that all allocations given to a client must come from the same IP.

There is no guarantee that a client will receive the number of ports specified in `krsip.conf` if there are not enough ports available on the gateway. If the gateway is not configured with enough IPs, the gateway will give out as many ports as it can on a first come, first served basis

1. Edit a copy of the current `rsipd.conf` configuration file. A read-only copy is available on the SMW at `/etc/opt/cray/rsipd/rsipd.conf`.

```
smw:~# vi rsipd.conf
```

2. Search for the `max_ports` entry, which may or may not already exist.

```
max_ports 768
```

`max_ports` sets the maximum number of ports that can be assigned to a single client. It is an integer value that can only be defined once, and it will apply to all RSAP-IP mode clients using this gateway.

3. Calculate the new value of `max_ports` using the following formula, where `ports_per_client` is the value used for `max_ports`.

```
clients_per_server = clients / (servers * ips_per_server)
ports_per_client = port_range / clients_per_server
```

4. Use the calculated value to define `max_ports` in the `rsipd.conf` file. If an existing entry exists, it **must be** replaced with the new value.
5. Save the modified configuration file.
6. Log on to the boot node and invoke `xtopview`.

For a single gateway:

```
smw:~# ssh root@boot
boot:~# xtopview
```

For multiple gateways:

```
boot:~# xtopview -n node
```

7. Copy the configuration file to the shared root.

The exact location on the shared root will vary, but one common location is `/.shared/base/default/etc/opt/cray/rsipd/`. If uncertain, contact the administrator that originally installed the site's RSIP instance. In a node-specific shared root (i.e., for multiple gateways), `/etc/opt/cray/rsipd/` should symbolically link to the correct location in most cases.

```
default:// # cp rsip.conf /.shared/base/default/etc/opt/cray/rsipd/
```

8. Exit `xtopview` and the boot node.

```
default:// # exit
boot:~# exit
```

9. Edit a copy of the current `krsip.conf` to change the number of ports the client should request under RSAP-IP mode (the default and currently only supported mode for Cray RSIP). This step must be done manually because `xtrsipcfg` currently does not support multiple assign requests or pools.

```
smw:~# vi krsip.conf
```

The format of entries in `krsip.conf` is:

```
nid server_ip alt_server_ip port_count delay
```

10. Modify the `port_count` entries.

This column indicates the number of ports the client should request under RSAP-IP mode (the default and currently only supported mode for Cray RSIP). This value should match the value of `max_ports` set in the `rsipd.conf` file.

Note that `port_count` can be different for every node. This allows some nodes to have more ports than others, or clients connecting to one gateway to have a different amount of ports than a group connecting to a different gateway, depending on the size of the groups.

11. Install `krsip.conf` as `compute/etc/krsip.conf` in the boot image. Follow site-specific procedures to build a new image and reboot the machine.

Currently, the `krsip` client cannot change its options without being restarted, and it cannot be unloaded. This means that the machine must be rebooted, and the new config file must be installed in the boot image.

12. Verify that multiple allocations have been made by executing the following from the gateway node:

```
node:~# /etc/init.d/rsipd dump
```

The output should display that each client has more than one allocation. Note that the time it takes for all clients to get their allocations varies depending on the number of clients per gateway.

Configure RSIP to use RSA-IP Mode

Prerequisites

- RSIP is installed on the Cray system and working with gateway address pools
- Root access is available on both the gateway and boot nodes (these can be the same)
- Sufficient external IPs exist (one for each RSA-IP client plus any IPs needed to cover the needs of any RSAP-IP clients that will be running)
- Ability to reboot the system

Realm Specific Address IP (RSA-IP) mode differs from the default Realm Specific Address/Port IP (RSAP-IP) mode in that clients operating in RSA-IP mode are allocated an entire IP by the gateway instead of a port range on an IP. No other client will be assigned the same IP or ports from the same IP.

1. Back up the current configuration files. On the gateway node, the `rsipd.conf` file is located at `/etc/opt/cray/rsipd/rsipd.conf`. If there are multiple gateway nodes, backup the `rsipd.conf` file on each of them as they are unique. The `krsip.conf` file is present on any compute node in `/etc/krsip.conf` and is identical across all compute nodes.
2. Generate the necessary client and server configuration files using a modified version of `xtrsipcfg` that supports RSA-IP mode; where `template_path` is the directory path for the installed operating system template to be modified. Consult the `xtrsipcfg` man page for a complete description of the options.

```
smw:~ # xtrsipcfg_v2 --format=cfg -b --template=template_path [-p=partition]
```

The tool will ask a series of questions to determine what values to place in the config files, including questions about configuring some nodes as RSA-IP clients. A new `rsipd.conf` and `krsip.conf` are created upon completion.

3. Log on to the boot node and invoke `xtopview`.

For a single gateway:

```
smw:~# ssh root@boot
boot:~# xtopview
```

For multiple gateways:

```
boot:~# xtopview -n node
```

4. Copy the configuration file to the shared root.

The exact location on the shared root will vary, but one common location is `/.shared/base/default/etc/opt/cray/rsipd/`. If uncertain, contact the administrator that originally installed the site's RSIP instance. In a node-specific shared root (i.e., for multiple gateways), `/etc/opt/cray/rsipd/` should symbolically link to the correct location in most cases.

```
default:// # cp rsipd.conf /.shared/base/default/etc/opt/cray/rsipd/
```

5. Exit `xtopview` and the boot node.

```
default:// # exit
boot:~# exit
```

6. Install `krsip.conf` as `compute/etc/krsip.conf` in the boot image. Follow site-specific procedures to build a new image and reboot the machine.

Currently, the `krsip` client cannot change its options without being restarted, and it cannot be unloaded. This means that the machine must be rebooted, and the new config file must be installed in the boot image.

7. Reboot the system in order for the changes to take effect.

IP Routes for CNL Nodes in the `/etc/routes` File

To provide route entries for compute nodes, edit the `/etc/routes` file in the compute node template image on the SMW. This provides a simple mechanism for the administrator to configure routing access from compute nodes to login and network nodes using external IP destinations without having to traverse RSIP tunnels. This mechanism is not intended to be used for general-purpose routing of internal HSN IP traffic. It is intended only to provide IP routes for compute nodes that need to reach external IP addresses or external networks. A new `/etc/routes` file is created in the compute images and is examined during startup. Non-comment, non-blank lines are passed to the `route add` command. The empty template file provided contains comments describing the syntax.

Update the System Configuration After a Blade Change

When a blade is changed in a Cray system, the configuration of the system must be updated. Updating the system is required after:

- Adding additional blades to the system
- Removing a blade from the system configuration
- Changing a blade from a compute blade to a service blade
- Changing a blade from a service blade to a compute blade

The system configuration can be updated when the system is not booted (see [Update the System Configuration after Hardware Changes when the System is not Booted](#) on page 172) or while the system is booted (see [Update the System Configuration While the System Is Booted](#)).

Update the System Configuration after Adding a Blade when the System is not Booted

1. Update the `CLEinstall.conf` file with the blade changes.
2. Execute the `CLEinstall` command, including the `--xthwinxmlfile` option, to update the system and prepare a boot image. The `CLEinstall --xthwinxmlfile` option orders the `CLEinstall` program to use previously captured configuration information. If using a partition and not the entire machine, use `pN` instead of `s0`.

```
smw:~ # ./CLEinstall --label=LABEL --upgrade --XTrelease=5.2.14 \
--xthwinxmlfile=/home/crayadm/install.5.2.14/xthwinv.s0.xml \
--configfile=/home/crayadm/install.5.2.14/CLEinstall.conf \
--CLEmedia=/home/crayadm/install.5.2.14
```

3. Run the `shell_bootimage_LABEL.sh` script, where `LABEL` is the system set label specified in `/etc/sysset.conf` for this boot image. Specify the `-c` option to automatically create and set the boot image for the next boot. For information about additional options accepted by this script, use the `-h` option to display a help message.

```
smw:~ # /var/opt/cray/install/shell_bootimage_LABEL.sh -c
```

4. Run the `shell_post_install.sh` script on the SMW to unmount the boot root and shared root file systems and perform other cleanup as needed.

```
smw:~ # /var/opt/cray/install/shell_post_install.sh /bootroot0 /sharedroot0
```

5. If adding a compute blade into the system configuration, boot the system normally.
6. If adding a service blade into the system configuration, complete the following steps:
 - a. Boot the boot node.
 - b. If the login node(s) or RSIP node(s) have changed, edit `/etc/sysconfig/network/ifcfg-eth0`.
 - c. Boot the SDB node and all other service nodes.
 - d. Create a backup copy of the `boot:/root/.ssh/known_hosts` file.
 - e. Delete the `boot:/root/.ssh/known_hosts` file.

- f. Update ssh keys for new service nodes by running the `/var/opt/cray/install/shell_ssh.sh` script, which creates a new `known_hosts` file. These keys are used for ssh commands to the blades, including `pdsh` commands that are called by `xtshutdown` and `/etc/init.d/lustre` and possibly by actions specified in the boot automation file on the SMW.

```
boot:~ # /var/opt/cray/install/shell_ssh.sh
```

- g. Configure the new nodes to support the role for which they were added.
- h. Update SMW boot automation files if new service nodes have been added or removed that are providing services (such as ALPS on login nodes) that are explicitly started by `hostname` in the boot automation file.
- i. Complete booting the system (or reboot using the boot automation file).

Update the System Configuration after Hardware Changes when the System is not Booted

IMPORTANT: After hardware changes are made when the system is **not** booted, follow these procedures to update the system configuration.

When blades are changed that have the same blade type and Processor Daughter Card (PDC) type, `xtdiscover` does not need to be executed. If these blades are changed or if cabling changes are made and `xtdiscover` does not have to be executed, the `xtbounce --linktune` command, which forces `xtbounce` to do full tuning on the system, must still be executed. For more information, see the `xtbounce(8)` man page.

1. Update the system configuration to reflect the changed blade configuration. If the blade or PDC type is different:

```
smw:~ # xtdiscover
```

2. If the blade or PDC type is different, execute the `xtzap --blade` command.

```
smw:~ # xtzap --blade blade_name
```

3. Execute the `xtbounce --linktune=all` command to tune PCIe and HSN links on the system. If using a partition and not the entire machine, use `pN` instead of `s0`.

```
smw:~ # xtbounce --linktune=all s0
```

4. Capture the system configuration for `CLEinstall` by executing the following `xthwinv` commands. (The `CLEinstall --xthwinvxmlfile` option will order the `CLEinstall` program to use this captured configuration information.) If using a partition and not the entire machine, use `pN` instead of `s0`. For additional information about the `CLEinstall` and `xthwinv` commands, see the `CLEinstall(8)` and `xthwinv(8)` man pages.

```
smw:~ # xthwinv -x s0 > /home/crayadm/install.5.2.14/xthwinv.s0.xml
```

Update the System Configuration While the System Is Booted

To change the system configuration physically while the system is booted, use the `xtwarmswap` command to remove or add one or more blades or to remove or add a high-speed network (HSN) cable.



CAUTION: When reserving nodes for maintenance, an `admindown` of any node in use by a current batch job can cause a subsequent `aprun` in the job to fail. Instead, it is recommended that a batch subsystem be used to first reserve nodes for maintenance, and then verify that a node is not in use by a batch job prior to setting a node to `admindown`. Contact a Cray service representative to reserve nodes for maintenance.

The `xtwarmswap` command runs on the SMW and coordinates with the `xtnlrd` daemon to take the necessary steps to perform warm swap operations. For additional information, see the `xtwarmswap(8)` man page.

Reuse One or More Previously-failed HSN Links

Before failed links can be reintegrated into the HSN configuration, the alert flags on the LCBs that are to be reused must be cleared. The `xtnlrd` daemon will automatically do this when told to perform a warm swap `sync` operation.

Execute an `xtwarmswap -s LCB_names -p partition_name` to clear the alert flags on the specified LCBs and to reroute the HSN using those links.

Note that by specifying `all` on the `xtwarmswap -s` option, all LCBs with alerts will be reused and, hence, their alert flags will be cleared. Alternatively, by specifying `none` on the `xtwarmswap -s` option, no LCB alerts will be cleared and the system will be rerouted without any changes to the configuration.

The `xtwarmswap` command results in `xtnlrd` performing the same link recovery steps as for a failed link, but with two differences: no alert flags are set, and an `init_new_links` and a `reset_new_links` step are performed to initialize both ends of any links to be used, before new routes are asserted into the Aries™ routing tables.

The elapsed time for the warm swap synchronization operation is typically about 30 seconds.

Reuse One or More Previously-failed Blades, Mezzanines, or Cabinets

Failed blades have alert flags set on the ASICs and the LCBs. These alert flags must be cleared before the blades, Mezzanines, or cabinets can be reused.

Before previously-failed blades can be reused, the alert flags on their Gemini ASICs and on the link endpoints (LCBs) must be cleared. This is done automatically by `xtnlrd` when told to perform a warm swap `add` operation.

Perform an `xtwarmswap --add` operation to bring the blades back into the HSN configuration. Doing so cold starts the blades and initializes the links to the blades.

1. Ensure that blades/Mezzanines/cabinets have power.
2. Ensure that an `xtalive` command to all required blades succeeds.
3. Add the blade(s) to the HSN by executing the `xtwarmswap --add blade_ID,...` command. Note that this command automatically executes a `mini-xtdiscover` command after the warm swap steps have completed

successfully. No manual invocation of `xtdiscover`, which gets the new hardware attributes from the added blades, is necessary.

Because the `xtwarmswap --add` command cold starts the added blades, the time to return the blades back to service is about 10 minutes for the cold start and about 60 seconds for the link-recovery handling.

4. Boot the nodes on the blade(s) by executing the `xtcli boot CNL0 blade_ID,...` command on the SMW.

Add or Remove a High-speed Network Cable from Service

To specify a high-speed network (HSN) cable to add or remove from service, use the `xtwarmswap --add-cable cable` command or the `xtwarmswap --remove-cable cable` command, respectively.

These options are not supported if more than a single active partition exists in the system. Also, the `-p|--partition` option must not be specified when using these options. The routing of the Cray HSN will be updated to route around the removed cable. In addition, the `--linktune` option must not be specified when using the `--remove-cable` option.

Remove a Compute Blade from Service While the System is Running

A compute blade can be physically removed for maintenance or replacement while the system is running; however, the applications using the nodes on the blade to be removed must be allowed to drain, or be killed beforehand.

Verify that the proposed system configuration is routable prior to starting this warm swap procedure. Doing this in advance of idling the nodes on the blades to be removed provides assurance that a valid set of nodes is being taken out of service before affecting the system. Log on to the SMW as `crayadm` and execute the following command, where `pN` is the partition from which the blades are being removed:

```
smw:~> rtr -s --id test --remove=blade_ID,blade_ID,blade_ID,... pN
```



CAUTION: This procedure warm swaps a compute blade from service while the system is running. Do not warm swap service blades, unless the blade is an I/O base blade (IBB) that has InfiniBand cards and is an LNET blade. Before attempting to warm swap any service blade, it is advisable to consult with a Cray service representative.

1. Log on to the login node as `root`.
2. Execute the following command to mark the nodes on the compute blade as `admindown`. This tells ALPS not to launch new applications onto them. (This command may also be executed from the boot node as user `root`.)

```
login:~ # xtproadmin -k s admindown -n blade_ID
```

The arguments to the `-n` option should be the NID values for the nodes on the blade being removed, as shown by executing `xtproadmin | grep bladename`.

3. From the login node, execute the `apstat -n` command to determine if any applications are running on the node marked `admindown`. This example shows that `apid 1301` is running on NID 6:

```
login:~ # apstat -n
NID Arch State HW Rv Pl PgSz Avl Conf Placed PEs Apids
 6 XT UP I 8 8 8 4K 4096000 4096000 4096000 8 1301
 7 XT UP I 8 - - 4K 4096000 0 0 0
 8 XT UP I 8 - - 4K 4096000 0 0 0
```

4. Wait until the applications using the nodes on the blade finish or use the `apkill apid` command to kill the application.
5. Log on to the SMW as `crayadm`.
6. Execute the `xtcli halt blade_ID` command to halt the blade.

```
smw:~> xtcli halt blade_ID
```

7. Execute the `xtwarmswap --remove blade_ID` command to remove the compute blade from service. The routing of the Cray HSN will be updated to route around the removed blade.

The `--remove` stage of the `xtwarmswap` process uses the ASIC resiliency infrastructure and takes about 60 seconds to complete.

```
smw:~> xtwarmswap --remove blade_ID
```

AOC cables are disabled if they are detected to be unused; the most likely case being the removal of a blade from an active system while the cabling stays in place

8. Execute the `xtcli power down blade_ID` command, which helps to identify which blade to pull (all lights are off on the blade).

```
smw:~> xtcli power down blade_ID
```

9. Physically remove the blade, if desired. To complete this step, see the hardware maintenance and replacement procedures documentation for the Cray system, or contact a Cray Service representative.



CAUTION: If a blade cannot be reinstalled in the empty slot within 2 minutes, install a filler blade assembly in the empty slot; failure to do so can cause other blades in the system to overheat.

Return a Compute Blade into Service

After a blade has been repaired or when a replacement blade is available, use the following procedure to return the blade into service.

1. Physically insert the blade into the slot. To complete this step, see the hardware maintenance and replacement procedures documentation for the Cray system, or contact a Cray Service representative.
2. On the SMW, execute the `xtcli power up blade_ID` command.

```
smw:~> xtcli power up blade_ID
```

3. Ensure that the blade is ready by entering the following command, and wait until the command returns the correct response:

```
smw:~> xtalive blade_ID
The expected response was received.
```

4. Verify the status of the blade controller to ensure that its "Comp state" is "up" and that there are no flags set.

```
smw:~> xtcli status -t 10 blade_ID
```

5. Bounce the blade.

```
smw:~> xtbounce blade_ID
```

6. If a replacement blade is installed, enter the following command on the SMW to reflash the firmware on the replacement L0:

```
smw:~> xtflash -t 10 blade_ID
```

7. Enter the following command on the SMW to check the PIC firmware level on the blade:

```
smw:~> xtcheckpic -s blade_ID
```

The command returns a list of PICs that are not the latest version, as in this example.

```
!!INCORRECT Gemini Mezzanine PIC : c0-0c0s3 7:0x60 version=0x14
```

An administrator can update the firmware on the node and Gemini mezzanine PICs by issuing the `fm` command on the SMW.

An external PIC programmer (referred to as the MPLAB) must be used to update all other PICs (refer to the Cray Customer Service Best Practices WIKI at <http://service-new.us.cray.com/wiki/XT-PICs>). If it is not possible to update the PIC firmware immediately, update it during the next maintenance period that does not impact system availability.

8. If any node or Gemini mezzanine PIC is not at the correct level, complete the following steps:

- a. Disable SEDC.

In the `sedc_srv.ini` file, change `INT:startup_action = 1` to `INT:startup_action = 0`. The `sedc_srv.ini` file can be located at `/opt/cray/etc/sedc_srv.ini` or `/opt/cray/hss/default/etc/sedc_srv.ini`.

Then execute the following kill command:

```
smw:~> kill -1 `pidof sedc_manager`
```

- b. Program the affected PICs:

```
smw:~> fm -N -t proc blade_ID (node PIC)
smw:~> fm -N -t mezz blade_ID (mezzanine PIC)
```

If the Gemini mezzanine PIC command fails, enter the following command to use the low-speed PIC programming path:

```
smw:~> fm -N -t mezz -L blade_ID (mezzanine PIC)
```

- c. Reboot the blade:

```
smw:~> xtlogin blade_ID
(Compute) c0-0c0s2 / # reboot
```

The L0 requires approximately two minutes to reboot.

- d. Verify the reboot is complete:

```
smw:~> xtalive blade_ID
```

Reenter the command until it returns the correct response: The expected response was received.

- e. Enable SEDC.

In the `sedc_srv.ini` file, change `INT:startup_action = 0` to `INT:startup_action = 1`.

Then execute the following kill command:

```
smw:~> kill -1 `pidof sedc_manager`
```

9. Execute the `xtbounce --linkdown blade_ID` command to prepare the blade for the warm swap (takes down all HSN links on the blade).

```
smw:~> xtbounce --linkdown blade_ID
```

10. Add the blade(s) to the HSN by executing the `xtwarmswap --add blade_ID,...` command. This command activates routing on the newly installed blade and automatically executes a `mini-xtdiscover` command once the warm swap steps have completed successfully. No additional manual invocation of `xtdiscover`, which gets the new hardware attributes from the added blades, is necessary.

```
smw:~> xtwarmswap --add blade_ID
```

Because the `xtwarmswap --add` command cold starts the added blades, the time to return the blades back to service is about 10 minutes for the cold start and about 60 seconds for the link-recovery handling.

11. Boot the nodes on the blade(s) by executing the `xtcli boot CNL0 blade_ID,...` command on the SMW.

```
smw:~> xtcli boot CNL0 blade_ID
```

12. As `root` on the login node, execute the following command to mark the nodes on the compute blade as `up`. This tells ALPS that new applications may be launched onto those nodes. (This command may also be executed from the boot node as user `root`.)

```
login:~ # xtproadmin -k s up -n blade_ID
```

13. Verify that the blade is up.

```
login:~ # xtproadmin | grep blade_ID
```

Cray Lightweight Log Management (LLM) System

The Cray Lightweight Log Management (LLM) system is the log infrastructure for Cray systems and must be enabled for systems to successfully log events. At a high level, a library is used to deliver messages to `rsyslog` utilizing the RFC 5424 protocol; `rsyslog` transports those messages to the SMW and places the messages into log files.

The LLM system relies on the `sessionid` that is generated by `xtbootsys`. Therefore, systems must always be booted using `xtbootsys`. If the site has multi-part boot procedures or uses manual procedures, have the process started by an `xtbootsys` session. That session can be effectively empty -- it is only needed to initiate a boot `sessionid`. Subsequent `xtbootsys` calls can then use `--session last` or manual processes.

By default, LLM has a log trimming mechanism enabled called `xttrim`.

IMPORTANT: Do not use the `xtgetsyslog` command because it is not compatible with LLM. For additional information, see [Manage Log Files Using CLE and HSS Commands](#) on page 81.

For further information, see the `intro_LLM(8)` and `intro_LLM_logfiles(5)` man page.

Configure LLM

The LLM system is intended to work as a turnkey system. In most cases little or no configuration is required.

Both the SMW and CLE software installation tools allow certain LLM settings to be defined. The installation tool enforces any of these settings in the LLM configuration file, which means that these settings will not be accidentally lost during an upgrade due to changes in the `llm.conf` file that is distributed with the LLM software.

NOTE: The `llm.conf` file may be automatically replaced during the upgrade process. The previous version will be renamed `llm.conf.rpmsave`. Settings made in the `SMWinstall.conf` and `CLEinstall.conf` files will be enforced in the new `llm.conf` configuration file. It is important to review the resulting `llm.conf` file after a software update to ensure all settings are as intended.

The most critical parameter is `LLM` in the `SMWinstall.conf` and `CLEinstall.conf` files. This parameter must be set to `LLM=yes`, which ensures that `enabled=yes` is set in the `llm.conf` file. The `LLM=yes` setting is the only one needed for basic LLM operation.

IMPORTANT: It is recommended that the minimal possible configuration be performed in order to achieve the needed results. Set or change items only if needed. If a setting must be changed, change it in the `SMWinstall.conf` and/or `CLEinstall.conf` files if they provide a mechanism to change that value. Make direct changes to the `llm.conf` configuration file (`/etc/opt/cray/llm/llm.conf`) only if the `SMWinstall.conf` and `CLEinstall.conf` files do not provide a mechanism to change that value.



CAUTION: The `rsyslog.conf` configuration file is not intended to be modified locally. Configurable settings can be found in the `llm.conf` configuration file. Modifications outside of those provided by `llm.conf` may cause failure in other Cray software components, such as `xtumpsys`. If the provided configuration does not have the needed functionality, it is recommended that the logs be forwarded to another host where custom file processing can be performed without risk to critical software components.

For a description of all LLM configuration settings, see the `llm.conf(5)` man page or the `llm.conf` configuration file.

State Manager LLM Logging

The log data from the State Manager is written to `/var/opt/cray/log/sm-yyyyymmdd`. The default setting for the State Manager is to enable LLM logging. If LLM or `craylog` failures occur, State Manager logging is not disrupted. Logging then reverts to behavior that is very similar to legacy State Manager logging, which is also used when State Manager LLM logging is turned off.

To disable LLM logging for the State Manager, add the `-L` option to the `/opt/cray/hss/default/bin/rsms` script entry:

```
sm=(/opt/cray/hss/default/bin/state_manager sm "-L n")
```

Boot Manager LLM Logging

The log data from the Boot Manager is written to `/var/opt/cray/log/bm-yyyymmdd`. If the `-L` command line option is used with the `bootmanager` command or if LLM is not enabled, Boot Manager reverts to legacy logging, which writes log data to `/var/opt/cray/log/bm.out`. This is a less satisfactory logging method because each Boot Manager restart creates a new log and moves the previous log to `bm.out.1`. A third restart can possibly cause recent log data to be lost.

LLM Configuration Tips

- If the RAID controllers do not have IP addresses that begin with 10.1.0., be sure to specify the `llm_raid_ip=` option in the `SMWinstall.conf` file. Failure to set this correctly will result in the RAID logs not going into their specified location of `/var/opt/cray/log/raid-yyyymmdd`.
- Verify that the `/var/opt/cray/log` directory is on a different disk than the root hard drive. The subdirectories of `/var/opt/cray` should be links to a different disk.
- If an active log file is deleted while `rsyslog` is running, `rsyslog` will continue to write to the file handle even though there is no longer an entry for the file in the directory table of contents. Once `rsyslog` exits, all references to that file handle are gone, so the contents will be lost. To delete a currently open log file, the suggested approach is to rename or remove the file and then `hup rsyslog (/etc/init.d/cray-syslog hup)` to tell it to reopen files.
- Ensure that the site log host system can handle the log files load. Otherwise, the messages will back up on the SMW and cause unexpected behavior.

Change the Location to Log syslog-ng Information

Syslog messages from the service partition are only present on the SMW in `/var/opt/cray/log/sessionid`. The log system does not support customization of the configuration files, but if custom log formats are desired, configure the log system to forward all system logs from the SMW to the site-provided log server. See the `intro_llm(8)` and `CLEinstall.conf(5)` man pages for more information.

Manage System Services

Synchronize Time of Day on Compute Node Clocks with the Clock on the Boot Node

A network time protocol (NTP) client, `ntpclient`, is available to install on compute nodes. By default, `ntpclient` is not installed. When installed, the time of day on compute node clocks is synchronized with the clock on the boot node.

Without this feature, compute node clocks will drift apart over time, as much as 18 seconds a day. When `ntpclient` is installed on the compute nodes, the clocks drift apart for a four-hour calibration period and then slowly converge on the time reported by the boot node.

The standard Cray system configuration includes an NTP daemon (`ntpd`) on the boot node that synchronizes with the clock on the SMW. Additionally, the service nodes run `ntpd` to synchronize with the boot node.

To install the `ntpclient` RPM in the compute node boot image, edit the `shell_bootimage_LABEL.sh` script and specify `CNL_NTPCLIENT=y`, and then update the CNL boot image. Optionally, the administrator can enable this feature as part of a CLE software upgrade by setting `CNL_ntpclient=yes` in the `CLEinstall.conf` file before the `CLEinstall` program is run.

On compute nodes, the computational overhead for `ntpclient` is negligible and a small increase (800K) to the memory footprint will be incurred. Minimal network overhead for the boot node is required to process NTP requests. For each compute node on the system, the boot node will send and receive one packet every 15 minutes. Even on very large Cray systems, the boot node will process fewer than 25 transactions a second to support `ntpclient` requests.

Add, Start, Restart, or Limit Services

Add, Start or Restart Services

Services can be added to the service nodes by using standard Linux mechanisms, such as executing the `chkconfig` command while in the `xtopview` utility on the boot node or executing `/etc/init.d/servicename start|stop|restart` (which starts, stops, or restarts a service immediately) on the service node. This is the recommended approach for most services.

Limit Services

A system administrator can set up a firewall to limit services that are running on the system. Cray has enabled the Linux kernel to provide this capability. Use the `iptables` command to set up, maintain, and inspect tables that contain rules to filter IP packets.

For more information about `iptables`, see the `iptables(8)` man page.

Add and Start a Service Using RCA

Services may also be added by using the Resiliency Communication Agent (RCA). Configuration with RCA is indicated if a service requires extra resiliency. The RCA monitors the service and restarts it in case of failure.

1. Modify the `service_cmd` table of the Service database (SDB) to include new service information (see [Update the SDB when Services Change](#) on page 149).

```
smw:~# umount /media/cdrom
```

2. Send a `SIGHUP` signal to the failover manager to reread the database.

Indicate Nodes on Which the Service Will Be Started

Use the `xtservconfig` command to indicate the node or nodes on which the service will be started. The `xtservconfig` command can be executed from any service node but is normally run from the boot node. Only user `root` can make changes using the `xtservconfig` command.

After a new service is configured, the node must be rebooted or a `SIGHUP` signal must be sent to the service on the affected node.

Adding the PBS-MOM service for a specific node

For this example, add the PBS-MOM service for node 5:

```
boot:~ # xtservconfig -n 5 add PBS-MOM
```

Force the fcmd to update its configuration information about a new or updated service on a node

Remember to log on to the affected node as user `root`.

```
# killall -HUP fcmd
```

The `killall -HUP fcmd` command causes the failover manager to read the database.

Effect a change for a new or updated service on a group of nodes

For this example, effect a change for login nodes 1 through 9:

```
boot:~ # pdsh -w login[1-9] "killall -HUP fcmd"
```

Create a Snapshot of /var

The `/var` directory on a Cray system can be configured either as persistent or nonpersistent (see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*). In the latter case, the `/var` directory is volatile, and its initial contents are rebuilt at boot time from a skeleton archive, `/.shared/var-skel.tgz`.

The advantage of using a nonpersistent `/var` directory is ease of management. Each time the system is rebooted, the `/var` directory is freshly recreated from the central skeleton file; therefore, accumulation of files and potential corruption of files with the `/var` directory is much less of a concern. However, because the contents of `/var` are not saved, if there is a need to update the initial contents of the `/var` directory (e.g., when a new package requires a directory), the skeleton archive must be updated.

The `xtpkgvar` command creates a compressed tar file with a skeleton snapshot of the `/var` directory. To add files to the directory, make changes in the `xtopview` shell to the `/var` directory and take a snapshot of it with the `xtpkgvar` command.



CAUTION: Use the `xtpkgvar` command only when configuring the shared-root file system. The `xtpkgvar` command is used by the `CLEinstall` utility.

For more information, see the `xtpkgvar(8)` man page.

Prevent Login Node Hangs

If all available processes on a login node are in use, the node can hang or become nearly unresponsive. A hang of this type can be identified primarily by the presence of `cannot fork` error messages, but it is also associated with an unusually large number of processes running concurrently, the machine taking several minutes to make a prompt available, or never making a prompt available. In the case of an overwhelming number of total processes, it is often a large number of the same process overwhelming the system, which indicates a `fork()` system call error in that particular program.

This problem can be prevented by making a few changes to configuration files in `/etc` on the shared root of the login node. These configurations set up the `ulimit` built-in and the Linux Pluggable Authentication Module (PAM) to enforce limits on resources as specified in the configuration files. There are two types of limits that can be specified, a soft limit and a hard limit. Users receive a warning when they reach the soft limit specified for a resource, but they can temporarily increase this limit up to the hard limit using the `ulimit` command. The hard limit can never be exceeded by a normal user. Because of the shared root location of the configuration files, the changes must be made from the boot node using the `xtopview` tool.

1. Log on to the boot node and invoke `xtopview` for the login nodes, where `login` is the class name for login nodes.

```
boot:~ # xtopview -c login
```

2. Edit the `/etc/security/limits.conf` file and add the following lines, where `soft_lim_num` and `hard_lim_num` are the number of processes at which the hard and soft limits are to be enforced. The `*` represents "apply to all users" but can also be configured to apply specific limits by user or group (see the `limits.conf` file for further options).

```
class/login:/ # vi /etc/security/limits.conf
```

```
* soft nproc      soft_lim_num
* hard nproc      hard_lim_num
```

3. Verify that the following line is included in the appropriate PAM configuration files for any authentication methods for which limits are to be enforced; the PAM configuration files are located in the `/etc/pam.d/` directory. For example, to enforce limits for users connecting through `ssh`, add the `pam_limits.so` line to the file `/etc/pam.d/sshd`. Other applicable authentication methods to also include are `su` in the file `/etc/pam.d/su` and local logins in `/etc/pam.d/login`.

```
session    required    pam_limits.so
```

For more information about the Pluggable Authentication Module (PAM), see the `PAM(8)` man page.

4. Exit to the boot node; the changes are effective immediately on login nodes.

```
class/login:/ # exit
```

5. Log on to a login node to test that the limits are in place. which should return the number specified as the soft limit for the number of processes available to a user, for example:

For more information about using the `ulimit` command, see the `ulimit(P)` man page.

```
boot:~ # ssh login
login:~ # ulimit -u
```

The number specified as the soft limit for the number of processes available to a user is displayed.

Archive the Shared-root Using `xtoparchive`

If a full backup copy cannot be made, Cray recommends making a backup copy of the `.shared` root structure before making significant changes to the shared root.

The `xtoparchive` command performs operations on an archive of shared-root configuration files. Run the `xtoparchive` command on the boot node using the `xtopview` utility in the default view. The archive is a text-based file similar to a tar file and is specified using the required `archivefile` command-line argument. The `xtoparchive` command is intended for configuration files only. Binary files will not be archived. If a binary file is contained within a specification file list, it will be skipped and a warning will be issued.

1. Log on to the boot node and invoke `xtopview`.
2. Add the files specified in `specfile` to the archive file `archivefile`.

```
default/~/ # xtoparchive -a -f specfile archivefile
```

To archive any specialized files that have changed, invoke the `archive_etc.sh` script. This can occur while the system is booted or from the boot root and shared root in a system set that is not booted. The `archive_etc.sh` script uses the `xtoprdump` and `xtoparchive` commands to generate an archive of specialized files on the shared root.

3. Exit `xtopview`.

For more information about archiving and upgrading specialized files, see the `shared_root(5)`, `xtoparchive(8)`, `xtopco(8)`, `xtoprdump(8)`, and `xtoprlog(8)` man pages.

Use Linux Utilities to Back Up Limited Shared-root Configuration Data

Cray recommends that if a full backup copy cannot be made, make a backup copy of the `.shared` root structure before making significant changes to the shared root.

This procedure uses the Linux `tar(1)` command and must be run from the boot node.

1. Change to the shared root directory that is to be backed up.

```
boot:/rr # cd /rr/current
```

2. Create a tar file for the directory.

```
boot:/rr # tar czf /rr/dot_shared-20120929.tgz .shared
```

3. Change to the `/rr` directory.

```
boot:/rr # cd /rr
```

4. Verify that the file exists.

```
boot:/rr # ls -al dot_shared-20120929.tgz
-rw-r--r--  1 root  root    7049675 Sep 29 14:21
dot_shared-20050929.tgz
boot:/rr #
```

Back Up the Boot Root and Shared Root

Before backing up the boot root and shared root, consider the following issues.

- User `root` must perform this task.
- Do not have file systems mounted on the SMW and the Cray system at the same time.
- File system device names may be different at the site.
- If the backup file systems have not been used yet, `mkfs`s may need to be run prior to beginning the backup.
- File systems should be quiescent and clean (`fsck`) to get an optimal dump and restore.

Back up the boot root and the shared root by using the `xthotbackup` command or by using the Linux `dump` and `restore` commands.

Use the xthotbackup Command to Back Up Boot Root and Shared Root

Execute the `xthotbackup` command on the SMW to create a bootable backup. The `xthotbackup` command must be executed with `root` privileges. The system set labels in `/etc/sysset.conf` define disk partitions for backup and source system sets which are used by `xthotbackup` to generate the appropriate `dump` and `restore` commands. The entire contents of the disk partitions defined in a source system set are copied to the corresponding disk partitions in the backup system set. The backup and source system sets must be configured with identical partitions. (Follow the steps provided on the `xthotbackup(8)` man page in the Initial Setup section to set up identical system sets.) The disk partitions in the backup system set are formatted prior to the `dump` and `restore` commands.

IMPORTANT: The `xthotbackup` utility can also work with Logical Volume Manager (LVM) volumes, but this requires extra configuration before LVM snapshots can be created. For more information on LVM configuration and `xthotbackup` use with LVM, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)* and the `xthotbackup(8)` man page.

By default, `xthotbackup` forces file system checks by using `fsck -f`, unless the `xthotbackup -n` option is specified. All `fsck` activity is done in parallel (by default, `xthotbackup` uses the `fsck -p` option to check all file systems in parallel), unless the `xthotbackup -l` option is specified.

Load the `cray-install-tools` module to access the `xthotbackup` utility and the `xthotbackup(8)` man page.



WARNING: Do not use the `xthotbackup` command when either the source or destination system set is booted. Running `xthotbackup` with a booted system set or partition could cause data corruption.

Use the xthotbackup command to create a bootable backup system set

For this example, dump all of the partitions from the source label, `BLUE`, to the backup label, `GREEN`, and then make them bootable.

```
smw:~ # xthotbackup -a -b BLUE GREEN
```

The `xthotbackup` command can also be used to copy selected file systems from source to the backup system set.

Use the xthotbackup command to copy selected file systems from source to the backup system set

For this example, dump only the `SDB` and `SYSLOG` partitions in the system set labelled `BLUE` to the system set labelled `GREEN`.

```
smw:~ # xthotbackup -f SDB,SYSLOG BLUE GREEN
```

Back Up the Boot Root and Shared Root Using dump and restore

1. Verify that the Cray system is halted.
2. Open a `root` session.

```
crayadm@smw:~> su -
```

3. Mount the boot root to the SMW.

```
smw:~ # mount /dev/sda1 /bootroot0
```

4. Mount the backup boot root to the SMW.

```
smw:~ # mount /dev/sdb1 /bootroot1
```

5. Change directories to the backup boot root.

```
smw:~ # cd /bootroot1
```

6. Dump and restore boot root to the backup boot root.

```
smw:/bootroot1 # dump -0 -f - /bootroot0 | restore -rf -
```

7. When the dump is complete, unmount both boot-root file systems.

```
smw:/bootroot1 # cd /  
smw:/ # umount /bootroot0 /bootroot1
```

8. Mount the shared root to the SMW.

```
smw:/ # mount /dev/sdc6 /sharedroot0
```

9. Mount the backup shared root to the SMW.

```
smw:/ # mount /dev/sdg6 /sharedroot1
```

10. Change directories to the backup shared root.

```
smw:/ # cd /sharedroot1
```

11. Dump and restore shared root to the backup shared root.

```
smw:/sharedroot1 # dump -0 -f - /sharedroot0 | restore -rf -
```

12. When the dump is complete, unmount both shared root file systems.

```
smw:/sharedroot1 # cd /  
smw:/ # umount /sharedroot0 /sharedroot1
```

13. Exit the root session.

```
smw:~ # exit
```

Restore the HSS Database

When the `xtdiscover` command is executed, it automatically makes a backup copy of partition information, the entire HSS database, and the `/etc/hosts` file. In the event that the HSS database is lost or corrupted for any reason, such as a disk failure, the HSS database can be restored.

Execute these commands on the SMW.

```
smw:~> rsms stop
smw:~> mysql -uhssds -phssds < /home/crayadm/hss_db_backup/database_backup_file
smw:~> cp /home/crayadm/hss_db_backup/hosts_backup_file /etc/hosts
smw:~> rsms start
```

Recover from Service Database Failure

If there are problems with the SDB, for example, if commands like `xtprocdadmin` do not work, restart the service-node daemons.

```
sdb:~ # /etc/init.d/sdb restart
```

Commands in this file stop and restart MySQL.

Database Server Failover

The SDB uses dual-ported local RAID to store files.

If SDB node failover is configured, one service processor acts as the primary SDB server. If the primary server daemon dies, or the node on which it is running dies, the secondary (backup) SDB server that connects to the same RAID storage starts automatically. IP failover directs all new TCP/IP connections to the server, which now becomes the primary SDB server. Connections to the failed server are ended, and an error is reported to the client.

Rebuild Corrupted SDB Tables

The boot process creates all SDB tables except the accounting and boot tables. A small amount of corruption within an SDB table can be manually changed, using the tools in [Service Database Update Commands](#) on page 147, such that a reboot is not necessary. If the database table cannot be recovered, as a last resort reboot the system.

How to Handle Single-node Failures

A single-node failure is visible when using the `xtnodestat` command.

The `syslog` can be parsed to look for failures.

If problems with a node are suspected, invoke the `xtcli status` command. Nodes that have failed show an `alert` status. Jobs are not scheduled on the node as long as the alert is set. If problems persist, consult a service representative.

To see cabinet status, use the System Environmental Data Collections (SEDC); see *Using and Configuring System Environment Data Collections (SEDC) (S-2491)*.

For more information, see the `xtnodestat(1)`, `xtcli(8)`, and `xtsedcvier(8)` man pages.

Increase the Boot Manager Timeout Value

On systems of 4,000 nodes or larger, the time that elapses until the boot manager receives all responses to the boot requests can be greater than the default 60-second time-out value. This is due, in large part, to the amount of other event traffic that occurs as each compute node generates its console output.

To avoid this problem, change the `boot_timeout` value in the `/opt/cray/hss/default/etc/bm.ini` file on the SMW to increase the the default time-out value; for example:

Increase the `boot_timeout` value

For systems of 4,000 to 7,000 nodes, change the `boot_timeout` line to:

```
boot_timeout 120
```

For systems larger than 7,000 nodes, change the `boot_timeout` line to:

```
boot_timeout 180
```

The Application Level Placement Scheduler (ALPS)

ALPS (Application Level Placement Scheduler) is the Cray supported mechanism for placing and launching applications on compute nodes. ALPS provides application placement, launch, and management functionality and cooperates closely with third-party batch systems for application scheduling across Cray systems. The third-party batch systems make policy and scheduling decisions, while ALPS provides a mechanism to place and launch the applications contained within batch jobs.

ALPS application placement and launch functionality is only for applications executing on compute nodes. ALPS does not provide placement or launch functionality on service nodes.

ALPS performs the following functions:

- Assigns application IDs
- Manages compute node resources by satisfying reservation requests from third-party batch systems
- Provides a configurable node selection algorithm for placing applications (See the `ALPS_NIDORDER` configuration parameter in [The `/etc/sysconfig/alps` Configuration File](#) for more information.)
- Launches applications
- Delivers signals to applications
- Returns `stdout` and `stderr` from applications
- Provides application placement and reservation information
- Supports batch and interactive workloads
- Supports huge pages functionality for CNL applications
- Provides an XML interface for third-party batch-system communication
- Provides launch assistance to debuggers, such as TotalView
- Supports application placement of nonuniform numbers of processing elements (PEs) per node, allowing full use of all compute node resources on mixed-node machines
- Works with the CLE Node Health software to perform application cleanup following the non-orderly exit of an application (see [When an Application Terminates](#)) and reservation cleanup following each batch job. For additional information about the CLE Node Health software, see [Configure the Node Health Checker \(NHC\)](#) on page 120.

ALPS Architecture

The ALPS architecture includes the following clients and daemons, each intended to fulfill a specific set of responsibilities as they relate to application and system resource management. The ALPS components use TCP/IP sockets and User Datagram Protocol (UDP) datagrams to communicate with each other. The `apinit` daemon executes on compute nodes. All other ALPS components execute on service nodes (login, SDB, and boot nodes).

ALPS Clients

The ALPS clients provide the user interface to ALPS and application management. They are separated into the following distinct areas of functionality: application submission, application and reservation status, application signaling, administrator interface to ALPS, and batch system integration.

- `aprun`: Application submission
- `apstat`: Application placement and reservation status
- `apkill`: Application signaling
- `apmgr`: Collection of functions usually used by the system administrator in exceptional circumstances to manage ALPS
- `apbasil`: Workload manager interface

For detailed descriptions, see the client-specific section and man page.

ALPS Daemons

ALPS daemons provide support for application submission, placement, execution, and cleanup on the system.

- `apsys`: Client local privileged contact
- `apinit`: Application management on compute nodes
- `apsched`: Reservations and placement decisions
- `apbridge`: System data collection
- `apwatch`: Event monitoring
- `apres`: ALPS database event watcher restart daemon

For detailed descriptions, see the daemon-specific section and man page.

ALPS Log Files

Each of the ALPS daemons writes information to its log file in `/usr/opt/cray/alps/log` on whichever node that runs the daemon. The name of the log file consists of the daemon name appended with the month and day, such as `apsched0302`.

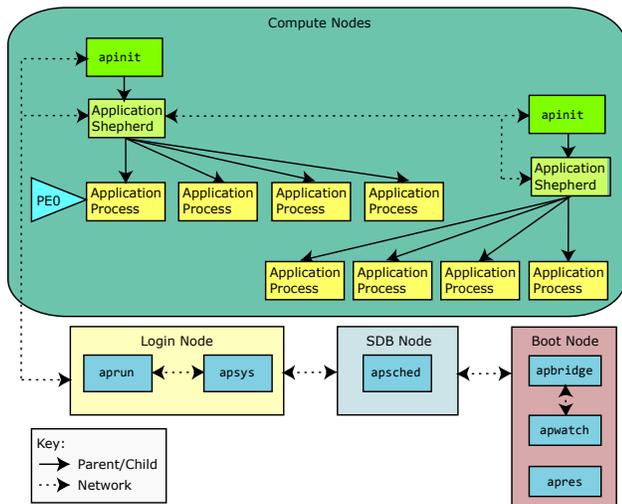
The `apinit` log file is in the `/usr/opt/cray/alps/log` directory on each compute node and also has a node ID appended to it, such as `apinit0302.00206`. Because this directory is in memory, the `apinit` log file is lost when a compute node is rebooted.

Each system has one `apbridge` daemon, one `apwatch` daemon, and one `apres` daemon, all of which must execute on the same node. By default, this is the boot node. These three daemons write to one log file on that node. The log file name format is `apbridgemmd`, for example, `apbridge1027`.

The ALPS Process

ALPS uses memory-mapped files to consolidate and distribute data efficiently, reducing the demand on the daemons that maintain these files by allowing clients and other daemons direct access to data they require.

Figure 4. ALPS Process



ALPS Clients

The aprun Client

The `aprun` client is used for application submission. Specifically, a user executes the `aprun` command to run a program across one or more compute nodes. The `aprun` client serves as the local representative of the application and is the primary interface between the user and an application running on compute nodes. The `aprun` client parses command-line arguments to determine the application resource requirements. These requirements are submitted locally to `apsys`, which forwards them to `apsched` for application placement.

After the application has an assigned placement list of compute nodes, `aprun` provides application-launch information to the `apinit` daemon on the first compute node in the placement list. The `aprun` client also provides user identity and environment information to `apinit` so that the user's login node session can be replicated for the application on the assigned set of compute nodes. This information includes the `aprun` current working directory, which must be accessible from the compute nodes.

The `aprun` client forwards `stdin` data to `apinit`, which is delivered to the first processing element (PE) of the application. Application `stdout` and `stderr` data is sent from `apinit` to `aprun` on the login node.

The `aprun` client catches certain signals (see the `aprun(8)` man page) and forwards the signal information to `apinit` for delivery to the application. Any signal that cannot be caught and that terminates `aprun` causes `apinit` to terminate the application.

IMPORTANT: Do not suspend `aprun`. It is the local representative of the application that is running on compute nodes. If `aprun` is suspended, the application cannot communicate with ALPS, such as sending exit notification to `aprun` that the application has completed.

For more information about using the `aprun` command, see the `aprun(8)` man page.

The apstat Client

The `apstat` client reports on application placement and reservation information. It reflects the state of `apsched` placement decisions. The `apstat` client does not have dynamic run-time information about an application, so the

`apstat` display does not imply anything about the running state of an application. The `apstat` display indicates statically that an application was placed and that the `aprun` claim against the reserved resources has not yet been released.

If no application ID (*apid*) is specified when executing the `apstat` command, the `apstat` command displays a brief overview of all applications.

For detailed information about this status information, see the `apstat(1)` man page.

The `apkill` Client

The `apkill` client is used for application signaling. It parses the command-line arguments and sends signal information to its local `apsys` daemon. The `apkill` command can be invoked on any login or service node and does not need to be on the same node as the `aprun` client for that application. Based upon the application ID, `apsys` finds the `aprun` client for that application and sends the signal to `aprun`, which sends signal information to `apinit` for delivery to the application.

The `apkill` client can send a signal only to a placed application, not a pending application.

For more information about the actions of this client, see the `apkill(1)` man page and the Linux `signal(7)` man page.

The `apmgr` Client

The `apmgr` command is a collection of ALPS-related functions for use by system administrators. These functions (subcommands) often require `root` permission and are usually used in exceptional circumstances to manage ALPS. The `apmgr` command is not typically installed on the boot node's file system; it is available on and is run from service nodes.

For information about using the `apmgr` subcommands, see the `apmgr(8)` man page.

The `apbasil` Client

The `apbasil` client is used for batch system integration. It is the interface between ALPS and the batch scheduling system. The `apbasil` client implements the Batch and Application Scheduler Interface Layer (BASIL). When a job is submitted to the batch system, the batch scheduler uses `apbasil` to obtain ALPS information about available and assigned compute node resources to determine whether sufficient compute node resources exist to run the batch job.

After the batch scheduler selects a batch job to run, the batch scheduler uses `apbasil` to submit a resource reservation request to the local `apsys` daemon. The `apsys` daemon forwards this reservation request to `apsched`. If the reservation-request resources are available, specific compute node resources are reserved at that time for the batch scheduler use only.

When the batch job is initiated, the prior confirmed reservation is bound to this particular batch job. Any `aprun` client invoked from within this batch job can claim compute node resources only from this confirmed reservation.

The batch system uses `apbasil` to cancel the confirmed reservation after the batch job terminates. The `apbasil` client again contacts the local `apsys` daemon to forward the cancel-reservation request to `apsched`. The compute node resources from that reservation are available for other use after the application has been released.

For additional information, see the `apbasil(1)` and `basil(7)` man pages.

ALPS Daemons

The apbridge Daemon

The `apbridge` daemon collects data about the hardware configuration from the service database (SDB) and sends it to the `apsched` daemon. It also works with the `apwatch` daemon to supply ongoing compute node status information to `apsched`. The `apbridge` daemon is the bridge from the architecture-independent ALPS software to the architecture-dependent specifics of the underlying system.

The `apbridge` daemon is not intended for direct use; it is only installed in the boot root and is invoked from within `/etc/init.d/alps`.

For more information, see the `apbridge(8)` man page.

The aptsched Daemon

The `apsched` daemon manages memory and processor resources of applications running on compute nodes. Only one instance of the ALPS scheduler can run across the entire system at a time.

When `apsched` receives a request for application placement from `aprun`, it either returns a message regarding placement or a message indicating why placement is not possible (errors in the request or temporarily unavailable resources). When an application terminates, an exit message is sent to `apsched`, and it releases the resources reserved for the application.

The `apsched` daemon writes a log file on the node on which `apsched` is executing. By default, this is the SDB node. The level of debug messages written by the `apsched` daemon is defined in the `/etc/opt/cray/alps/alps.conf` configuration file. To dynamically change the debug level, modify the `alps.conf` file and send a `SIGHUP` signal to `apsched` to read the `alps.conf` file.

For more information, see the `apsched(8)` man page.

The apsys Daemon

The `apsys` daemon provides a central privileged point of contact and coordination between ALPS components running on login and other service nodes. The `apsys` daemon receives incoming requests and forks child agent processes to delegate responsibilities and improve scalability and responsiveness. An `apsys` daemon executes on each login node and writes a log file on each login node.

Each `aprun` client has an `apsys` agent associated with it. Those two programs are on the same login node and communicate with each other over a persistent TCP/IP socket connection that lasts for the lifetime of the `aprun` client. The `apsys` daemon passes `aprun` messages to `apsched` over a transitory TCP/IP socket connection and returns the response to `aprun`.

An `apsys` agent is created to service `apbasil` and `apkill` messages. These programs communicate over transitory TCP/IP socket connections. The `apsys` agent handles the `apkill` message itself and forwards `apbasil` messages to `apsched`.

Each `apsys` agent maintains a separate agents file that is located in the ALPS shared directory. The file name format is `agents.nid`, for example, `/ufs/alps_shared/agents.40`. For information about defining the ALPS shared directory, see [ALPS Configuration Files](#).

`apsys` also provides a facility to system administrators to execute prologue and epilogue actions for each `aprun` execution. Administrators can specify a script or binary pathname in the `alps.conf` configuration file under the `apsys` block. The variables used to specify these pathnames are `prologPath` and `epilogPath`.

`prologTimeout` and `epilogTimeout` are used to enforce timeouts on prologue and epilogue scripts and executables.

For more information, see the `apsys(8)` man page.

Change Debug Message Level

The level of debug messages written by the `apsys` daemon is defined in the `/etc/opt/cray/alps/alps.conf` configuration file. To dynamically change the debug level, modify the `alps.conf` file and send a `SIGHUP` signal to `apsys` to read the `alps.conf` file.

The apwatch Daemon

The `apwatch` daemon waits for events and sends compute node status changes to `apbridge`, which sends it to `apsched`. The `apwatch` daemon is not intended for direct use; it is only installed in the boot root and is invoked from within `/etc/init.d/alps`.

For more information, see the `apwatch(8)` man page.

The apinit Daemon

The `apinit` daemon launches and manages new applications. A master `apinit` daemon resides on every compute node, initiates all new activity on that node, and writes a log file on the compute node. The `aprun` client connects to the `apinit` daemon on the first node of an application's allocated node set and sends a launch message containing all of the information the compute nodes need to launch and manage the new application.

The `apinit` daemon then forks a child process (referred to as the *apshepherd* or just *shepherd*) and transfers responsibility for managing the application on that node to that child. If the application requires more compute nodes, the shepherd process communicates to the `apinit` daemon on the next compute node, which forks another shepherd child process.

If the application is placed on more than one compute node, ALPS uses a TCP fan-out control tree network for application management messages to do binary transfer of the application when requested, and to handle application `stdin`, `stdout`, and `stderr` data. The root of the fan-out control tree is `aprun`. The width of the fan out is configured within the `/etc/opt/cray/alps/alps.conf` file and is `32`, by default.

The `apinit` daemon is under the control of RCA. If the `apinit` daemon fails, RCA restarts `apinit`. If RCA is unable to restart `apinit` after several attempts, ALPS is notified and the node is made unavailable (`DOWN`) for applications.

For more information, see the `apinit(8)` man page.

The apres Daemon

The ALPS `apres` event watcher restart daemon registers with the event router daemon to receive `ec_service_started` events. When the service type is the SDB (`RCA_SVCTYPE_SDBD`), ALPS updates its data to reflect the current values in the SDB. The `apres` daemon is invoked as part of the ALPS startup process on the boot node.

The `apres` daemon is not intended for direct use; it is only installed in the boot root and is invoked from within `/etc/init.d/alps`.

For more information, see the `apres(8)` man page.

ALPS Configuration Files

ALPS uses the following three files:

- `/etc/sysconfig/alps` configuration file
- `/etc/opt/cray/alps/alps.conf` configuration file
- `/etc/init.d/alps` file - starts and stops ALPS components and does not require customization

IMPORTANT: When configuring the RAID LUNs (logical units), verify that write caching is enabled on the LUN that contains the ALPS shared file system. For more information about RAID configuration, see the *Installing Cray System Management Workstation (SMW) Software (S-2480)* and the *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*.

The `/etc/sysconfig/alps` Configuration File

The `/etc/sysconfig/alps` file is in both the boot root and in the shared root. If the ALPS-related parameters are defined in the `CLEinstall.conf` file, after installation the parameters and settings are placed into the `/etc/sysconfig/alps` file.

If the ALPS-related parameters are not defined in the `CLEinstall.conf` file, they must be defined in the `/etc/sysconfig/alps` file (required parameters are indicated) and then the ALPS daemons must be started in order to use ALPS.

IMPORTANT: When changing parameter settings, update the `/etc/sysconfig/alps` file in both the boot root and in the shared root and restart the ALPS daemons on all service nodes.

ALPS_MASTER_NODE	(Required) Specifies the node name (<code>uname -n</code>) of the service node that runs <code>apsched</code> . Cray recommends that the SDB node be used as the <code>ALPS_MASTER_NODE</code> . For example: <code>ALPS_MASTER_NODE="sdb"</code>
ALPS_BRIDGE_NODE	(Required) Specifies the node name (<code>uname -n</code>) of the service node that runs <code>apbridge</code> . This is usually the boot node. Network connectivity between the SMW and the <code>ALPS_BRIDGE_NODE</code> parameter is required. (Such connectivity is guaranteed to exist from the boot node.) This default value is enforced in the <code>/etc/init.d/alps</code> file. For example: <code>ALPS_BRIDGE_NODE="boot"</code>
ALPS_MOUNT_SHARED_FS	Specifies if a separate file system is to be mounted at ALPS startup to hold control data; default is <code>no</code> . For configurations using multiple login nodes, a shared file system is required, and the shared file system must be mounted before ALPS is started. For example: <code>ALPS_MOUNT_SHARED_FS="no"</code>
ALPS_SHARED_DEV_NAME	Specifies the device to mount at ALPS start-up. If it is null and <code>ALPS_MOUNT_SHARED_FS</code> is <code>yes</code> , the device is determined by <code>/etc/fstab</code> . This parameter is not used unless <code>yes</code> is specified for <code>ALPS_MOUNT_SHARED_FS</code> . For example: <code>ALPS_SHARED_DEV_NAME="ufs:/ufs/alps_shared"</code>

-
- ALPS_SHARED_MOUNT_OPTIONS** Specifies the shared mount options. Set this parameter only if `ALPS_MOUNT_SHARED_FS` is `yes` and `ALPS_SHARED_DEV_NAME` is not null. For example: `ALPS_SHARED_MOUNT_OPTIONS="-t nfs -o tcp,rw"`
- ALPS_IP_PREFIX** (Deferred implementation) Use of this parameter has no effect. Specifies the first two octets for IP addresses on the high-speed network (HSN). These are internal addresses within the HSN. For example:
`ALPS_IP_PREFIX="10.128"`
- ALPS_NIDORDER** The `nid` ordering option for `apbridge` can be defined. Because this is a system-wide setting, Cray recommends that changing this option only when rebooting the system to ensure `apbridge` and `apsched` are restarted in the correct sequence. The valid values are:
- On** Leave `ALPS_NIDORDER` unset, or set this to `-On`, to assign order of nodes based solely on ascending numerical Node ID (NID) order. This is the default sorting order and the preferred option for Aries systems
 - Ox** Assigns node order by maximum dimension as the outer dimension; the smallest dimension will change most quickly. For example, a system with an "xyz" topology of 6x12x8 would have the *x* dimension changing most rapidly and the *y* dimension changing least rapidly, therefore this option would order them as (0,0,0), (1,0,0), (2,0,0), (3,0,0), (4,0,0), (5,0,0), (0,0,1), (1,0,1), (2,0,1) and so on.

This option often improves application performance over using the `-On` option. Applications that use a small percentage of the machine, especially on machines that are largely cubic in their dimensions, may not benefit from this configuration.

For max-major dimension ordering (minimum dimension varies fastest)
 - Oy** Specifies *y*-major dimension ordering, so that the *y* dimension always varies last. This is best used on Cray XE systems of more than three cabinets.
 - Or** This is the reverse of the `-Ox` ordering method; the order of nodes is assigned by the minimum dimension as the outer dimension. This is usually not recommended.
 - Of** For field ordering. (Uses `od_allocator_id` column.)
 - O2** Assigns order of nodes based on the "2x2x2" NID reordering method, which is a merger of the incidental small node packing of the simple NID number method and the inter-application interaction reduction of the "xyz" method. Cray recommends this option for Cray XE and Cray XK systems, as it produces improved bisection bandwidth of placements as compared to the `-Ox`, `-Oy`, `-Or`, or `-Of` ordering.
- APWATCH_LIBRARY_PATH** The `LD_LIBRARY_PATH` "add-on" needed for `apwatch`; it includes the path to the `gnet` and `glib` libraries and the `rsms` and `erd` libraries.
-

For example:

```
APWATCH_LIBRARY_PATH="/opt/gnet/lib:/opt/glib/lib:/opt/cray/
librmsevent.so: \
/opt/cray/libcray_event_router.so:/opt/gnome/lib64"
```

APWATCH_ERD

(Required) The host that has the event router daemon (ERD) running; typically, this is the host name of the SMW.

For example: APWATCH_ERD="smw"

ALPS_X2CONFIG

The directory where some Cray X2 configuration scripts are located. This is used only on Cray X2 systems and should be left unset on all other systems.

APEVENT_SYNC_SECS

The interval in seconds between `apevent sync` requests.

For example: APEVENT_SYNC_SECS="300"

A separate file system for control data is mounted at ALPS startup. This is assumed to be a mount point. Specify the device to mount at ALPS start-up using the parameter `ALPS_SHARED_DEV_NAME`. If it is null and `ALPS_MOUNT_SHARED_FS` is `yes`, the device is determined by `/etc/fstab`.

Sample `/etc/sysconfig/alps` configuration file

```
## Path:          System/ALPS
## Description:  ALPS options
## Type:         string
## Default:      alps
## ServiceReload: alps
## ServiceRestart: alps
#

## Type:         string
## Default:      ""
#
# Nodename (uname -n) of service node that will act as the ALPS
# master -- the node that will run apsched.  In most cases this
# will be the sdb node.  This parameter is required for all
# configurations.
#
ALPS_MASTER_NODE="sdb"

## Type:         string
## Default:      ALPS_MASTER_NODE or ""
#
# Nodename (uname -n) of service node that will act as the ALPS
# "bridge" -- the node that will run apbridge.  In most cases this
# will be the boot node.  This parameter is required for all
# configurations.  If no value is set, then the ALPS_MASTER_NODE
# will be used, but that will work only if there is network
# connectivity between the master node and the SMW.  (Such
# connectivity is guaranteed to exist from the boot node.)
# This default value is enforced when used - in the alps.init script,
# as in: ${ALPS_BRIDGE_NODE:=ALPS_MASTER_NODE}
# If this variable is read otherwise, the empty string will be found
# as
# the default value.
#
ALPS_BRIDGE_NODE="boot"
```

```

#
## Type:          yesno
## Default:       no
#
# Configurations that use multiple login nodes must use a shared
# filesystem. If "yes", the filesystem indicated by the
apsched:sharedDir
# variable in alps.conf will be mounted at ALPS start-up by the ALPS
start-up
# scripts. Note that /ufs mounts are brought up before ALPS is
launched, so
# if the alps.conf value is resident on /ufs, this option should be
"no".
# Otherwise, this script will have to mount the alps.conf value and
this option
# should be "yes".
#
ALPS_MOUNT_SHARED_FS="no"

#
# MOVED: ALPS_SHARED_DIR_PATH
#
# The value for the ALPS shared directory is now set within alps.conf
#
# alps.conf:
# apsched
# ...
# sharedDir /ufs/alps_shared
# ...
# /apsched
#

## Type:          string
## Default:       ""
## Example:       "ufs:/ufs/alps_shared"
#
# Device to mount at ALPS start-up. If it is null but
ALPS_MOUNT_SHARED_FS
# is "yes", then the device will be determined by /etc/fstab. This
# parameter is not used unless ALPS_MOUNT_SHARED_FS is "yes".
#
ALPS_SHARED_DEV_NAME=""

## Type:          string
## Default:       ""
## Example:       "-t nfs -o tcp,rw"
#
# This parameter is not used unless ALPS_MOUNT_SHARED_FS is "yes" and
# ALPS_SHARED_DEV_NAME is not null.
#
ALPS_SHARED_MOUNT_OPTIONS=""

## Type:          string
## Default:       "10.168"
## Example:       "10.168"
#
# The first two octets for IP addresses on the high-speed network.
# These are internal addresses within the HSN.
#
ALPS_IP_PREFIX="10.128"

```

```
## Type:      string
## Default:
## Example:   -On
#
# The nid ordering option for apbridge can be defined.
# The choices are:
#   -On or (leave unset) for numerical ordering, i.e. no special
order
#       This option (-On) is preferred for Aries systems
#   -Ox for max-major dimension ordering (minimum dimension varies
fastest)
#   -Oy for y-major dimension ordering (y dimension varies last)
#   -Or for reverse of max (i.e. min) ordering (usually not
recommended)
#   -Of for field ordering (uses od_allocator_id column)
#   -O2 for 2x2x2 ordering for older systems and certain irregular-
sized systems;
#       for most Gemini systems it uses an ordering that improves
further on
#       the 2x2x2 ordering; this (-O2) is recommended for Gemini
sites as
#       it improves bisection bandwidth of placements compared to -
Ox/y/r/n
#
ALPS_NIDORDER=""

## Type:      string
## Example:
## "/opt/gnet/lib:/opt/glib/lib:/opt/cray/librmssevent.so:/opt/cray/
libcray_event_router.so"
#
# The LD_LIBRARY_PATH "add-on" needed for APWATCH, it includes the
path
# to the gnet and glib libraries and the rsms and erd libraries.
#
APWATCH_LIBRARY_PATH="/opt/gnet/2.0.5/64/lib:/opt/glib/2.4.2/64/lib:/
opt/cray/lib64:/opt/gnome/lib64"

## Type:      string
## Default:   ""
## Example:   "whistler-smw"
#
# The host which has the Event Router Daemon running;
# typically this is the hostname of the smw.
# This parameter is required for all configurations.
#
APWATCH_ERD="smw"

## Type:      string
## Default:   "/opt/cray/etc"
## Example:   "/opt/cray/etc"
#
# The directory where some X2 config scripts are located;
# needs to be not empty only on systems w/an X2 component.
#
ALPS_X2CONFIG="/opt/cray/etc"

## Type:      integer
## Default:   "300"
## Example:   "60"
```

```
#
# The interval in seconds between apevent sync requests
#
APEVENT_SYNC_SECS="300"
```

The alps.conf Configuration File

The `/etc/opt/cray/alps/alps.conf` file is in the shared root and boot root and contains ALPS static configuration information used by the `apsched` and `apsys` daemons and the `aprun` and `apstat` commands. This file also contains application cleanup and reservation cleanup parameters.

The `alps.conf` file can be edited directly. Alternatively, the `apconf` utility provides scripts for querying, merging, and updating the contents of the `alps.conf`. For more information, see the `apconf(8)` man page.

NOTE: The parameter settings can be changed dynamically by modifying the `alps.conf` file and sending a `SIGHUP` signal to `apsched` or `apsys`, as applicable, to reread the `alps.conf` file.

Sample `/etc/opt/cray/alps/alps.conf` configuration file

```
#
# (c) 2015 Cray Inc. All Rights Reserved. Unpublished Proprietary
# Information. This unpublished work is protected to trade secret,
# copyright and other laws. Except as permitted by contract or
# express written permission of Cray Inc., no part of this work or
# its content may be used, reproduced or disclosed in any form.
#

# ALPS configuration file
# See the system admin guide for more information
# on possible settings and values

# Config values in this section affect all ALPS daemons
logging
# log method: 1 - traditional log files 2 - llm logging 3 - both
logMethod 1
/logging

apsched
alloc 0
bridge 1
fanout 32

# Debug level - a bitmask of what debug info to log:
# 0x0001 - General debug
# 0x0002 - Placement function details
# 0x0004 - sendMsg debug
# 0x0008 - Check listen port details
# 0x0010 - Recovery debug
# 0x0020 - XT optimized placement
# 0x0040 - Configuration debug
# 0x0080 - Context switch debug
# 0x0100 - Protection domain debug
# 0x0200 - Reserved for future use
# 0x0400 - GNI (Gemini/Aries) code debug
# 0x0800 - State refresh debug
# 0x1000 - Timings for reservations/appinfo file I/O
```

```

    debug          0
# Default CPU affinity: values cpu (default), none, numa, depth
#   cpuAffinity    cpu
# Default lustre cache flushing at app exit: values 0, 1
#   lustreFlush   1
# Default memory compaction at app exit: values 0, 1
#   memoryCompact 1
# Default app node share mode for cores and memory: values
exclusive, share
#   nodeShare     exclusive
# Max number of simultaneous reservations
#   maxResv       4096
# Default number of CPUs Per Compute Unit for batch/interactive
# (default is 0 on Gemini, 1 on Aries)
#   batchCPCU     0
#   interactiveCPCU 0
# If set, reservations will get all node resources
# (used for systems with hyperthreading)
#   resFullNode   1
# Maximum concurrent claims (apruns) allowed per reservation
#   claimsPerRes  1000

# Variables to control suspend/resume
# Enable suspend/resume code
#   suspendResume 1
# Max # of apps/node (default 4; must be 2 or 4)
#   loadLimit      4
# Number (e.g. 32) or size (e.g. 2G) of compute node fake numa nodes
# This must be identically configured on the compute nodes
# via kernel parameter 'numa=fake=<num>' or 'numa=fake=<siz>'
#   fakeNumaNodes 32
#   fakeNumaNodes 1G

# Values used for systems with Gemini or Aries network:
# - nttSize: if set to 0, the apps will NOT use the NTT
#   (and on Gemini, all apps will get a global PTag)
#   nttSize        0

# Values used for systems with Gemini network:
# - pTagGlobalNodes: if set to X, apps >= than X nodes
#   will use a global PTag (and NOT use the NTT)
#   pTagGlobalNodes 5000
# - pTagFreeDelay: delay freeing PTags for X seconds
#   pTagFreeDelay   30

# Value used for systems with Aries network:
# - pKeyFreeDelay: delay freeing PKeys for X seconds
#   pKeyFreeDelay   30

#
# Variables to enable user/system protection domains
#
# Max # of user protection domains allowed
#   pDomainMax      10
# List of system protection domains used by all applications
#   pDomainIDs      sys1,sys2

#
# Directory containing shared files
#
#   sharedDir       /ufs/alps_shared

```

```

/apsched

apsys

# Debug level - a bitmask of what debug info to log:
# 0x0001 - General debug
# 0x0002 - More detailed debug
# 0x0004 - Periodic messages issued in poll loops
# 0x0008 - Accounting debug messages
# 0x0010 - Reserved for future use
# 0x0020 - Timing of msync()/write() calls
    debug      1
# aruEnable - Y or Yes generates Application Resource Utilization
(ARU) files
#   aruEnable   Yes
# aruPath - file name template for ARU file
# specified name will have the apid appended when the file is created
# ignored unless aruEnable is set to Y or Yes
#   aruPath     /tmp/arufile
# prologPath - location of the executable file to be run before
application
#   prologPath  /usr/local/adm/sbin/prolog
# epilogPath - location of the executable file to be run after
application
#   epilogPath  /usr/local/adm/sbin/epilog
# prologTimeout - time in seconds before prolog is aborted as "hung"
#   prologTimeout  10
# epilogTimeout - time in seconds before epilog is aborted as "hung"
#   epilogTimeout  10
/apsys

aprun
# each of the aprun settings creates an association between an ALPS
value
# and a WLM environment variable that will be defined on the compute
nodes
#   defineNid    SLURM_NODEID
#   defineEachID SLURM_PROCID
#   defineNPPN   SLURM_TASKS_PER_NODE
#   defineLocalEnt SLURM_LOCALID
/aprun

apstat
#   nodeTable
NID,Arch,State,HW,Rv,Pl,PgSz,Avl,Conf,Placed,PEs,Apids
    nodeTable
NID,Arch,State,CU,Rv,Pl,PgSz,Avl,Conf,Placed,PEs,Apids
#   appsTable   Apid,ResId,User,PEs,Nodes,Age,State,Command
#   resvTable    ResId,ApId,From,Arch,PEs,N,d,Memory,State
#   pendingAppsTable  Pid,User,w:d:N,Nid,Age,Command,Why
#   gpuTable     NID,Module,State,Memory(MB),Family,ResId
# Gemini pDomainTable
#   pDomainTable  PDomainID,Type,Uid,Ptag,Cookie
# Aries pDomainTable
#   pDomainTable  PDomainID,Type,Uid,Cookie,Cookie2
/apstat

#
# Configure application and reservation cleanup
#
# configured - on or off (default = on)

```

```

# reports - save information about ongoing cleanups for apstat -C
(on or off)
# reportWait - time (ms) to wait for node health before writing a
report
# iterationSleep - time (ms) to pause between cleanup iterations
# iterationMax - maximum cleanup iterations to execute before giving
up
# connectTimeout - time (ms) to allow an inter-node connection to be
established
# connectAttempts - maximum inter-node connection attempts
# waitMin - minimum time (ms) to give any cleanup iteration to
complete. The
#           maximum wait time is calculated from this and other
cleanup
#           parameters.
#
cleanup
  application
    reports      on
    reportWait   2000
    iterationSleep 1000
    iterationMax  10
    connectTimeout 1000
    connectAttempts 5
    waitMin      5000
  /application
  reservation
    configured   on
    reports      on
    reportWait   2000
    iterationSleep 1000
    iterationMax  10
    connectTimeout 1000
    connectAttempts 5
    waitMin      5000
  /reservation
/cleanup

```

ALPS Global Configuration Parameters

These configuration parameters affect all ALPS daemons.

logMethod Select 1 to generate traditional log files, 2 to do llm logging, or 3 to do both.

apsched Configuration Parameters

These configuration parameters affect amsched only.

alloc If this field is set to 0 or is not specified, the distinction between batch and interactive nodes is enforced. If this field is set as nonzero, no distinction is made by ALPS; job schedulers will likely still limit their placement only to nodes marked as batch.

Default: 0

bridge	<p>Enables the <code>apbridge</code> daemon to provide dynamic rather than static information about the system node configuration to <code>apsched</code>. Cray strongly recommends setting the <code>bridge</code> parameter to use the <code>apbridge</code> daemon.</p> <p>Default: 1 (enabled)</p>
fanout	<p>This value controls the width of the ALPS TCP/IP network fan-out tree used by <code>apinit</code> on the compute nodes for ALPS application launch, transfer, and control messages.</p> <p>Default: 32</p>
debug	<p>This field is set to a default level of 0 for <code>apsched</code> and 1 for <code>apsys</code>. For information about valid values, see the <code>apsched(8)</code> and <code>apsys(8)</code> man pages.</p> <p>Default: 0 (<code>apsched</code>), 1 (<code>apsys</code>)</p>
cpuAffinity	<p>Supports switchable default CPU affinity in <code>apsched</code>. Valid values are <code>cpu</code>, <code>depth</code>, <code>numa</code>, and <code>none</code>. If the user has not explicitly set the CPU affinity using the <code>aprun -cc</code> option, the <code>aprun</code> command checks for and uses the CPU affinity from <code>apsched</code>; if there is not a value from <code>apsched</code>, it uses the default value. For more information, see the <code>aprun(1)</code> man page.</p> <p>Default: <code>cpu</code></p>
lustreFlush	<p>Supports switchable default Lustre cache flushing as part of application exit processing on the compute nodes. Enabling this Lustre cache flushing provides more consistent application run times. When Lustre cache flushing is enabled, all of the Lustre cache flushing completes as part of the application exit processing. The next application executing on the same set or subset of compute nodes no longer inherits a variable amount of run time due to Lustre cache flushing from a previous application.</p> <p>Valid values are 0 (disabled) and 1 (enabled). <code>Apsched</code> provides this <code>lustreFlush</code> value to the <code>apinit</code> daemon to enable or disable Lustre cache flushing as part of application exit processing.</p> <p>This value cannot be set on an individual application basis. It is a system-wide setting.</p> <p>Default: 1 (enabled)</p>
memoryCompact	<p>Supports switchable memory compaction at application exit. Valid values are 0 (disabled) and 1 (enabled).</p> <p>Default: 1 (enabled)</p>
nodeShare	<p>Controls which compute node cores and memory are put into an application <code>cpuset</code> on the compute node. The valid values are <code>exclusive</code> and <code>share</code>.</p> <p>The <code>exclusive</code> setting puts all of a compute node's cores and memory resources into an application-specific <code>cpuset</code> on the compute node. This allows the application access to any and all of the compute node cores and memory. This can be useful when specifying a particular CPU affinity binding string through the <code>aprun -cc</code> option.</p> <p>The <code>share</code> setting restricts the application specific <code>cpuset</code> contents to only the application reserved cores and memory on NUMA node boundaries. That is, if an application requests and is assigned cores and memory on NUMA node 0, only NUMA node 0 cores and memory will be contained within the application <code>cpuset</code>. The application</p>

will not have access to the cores and memory on other NUMA nodes on that compute node.

To override the default system-wide setting in `/etc/opt/cray/alps/alps.conf` on an individual basis, use the `aprun -F` option. For more information, see the `aprun(1)` man page.

Default: `exclusive`

maxResv	The maximum number of simultaneous reservations allowed. Default: 4096
batchCPCU	Specifies the default number of CPUs per compute unit for nodes whose allocation mode attribute is marked as <code>batch</code> . Defaults: 0 (Gemini systems); 1 (Aries systems)
interactiveCPCU	Specifies the default number of CPUs per compute unit for nodes whose allocation mode attribute is marked as <code>interactive</code> . Defaults: 0 (Gemini systems); 1 (Aries systems)
resFullNode	When set to a non-zero value, reservations will get all node resources (threads) regardless of other reservation parameters. The other reservation parameters will be taken into account to determine how many nodes are needed to accommodate the reservation. For example, a reservation with a <code>width</code> of 8 and an <code>nppn</code> of 4 will reserve two nodes. When <code>resFullNode</code> is not set, ALPS would limit the user to launching only four instances of the application on each node. With <code>resFullNode</code> set, ALPS will still reserve two nodes but it will reserve all the hyper-threads and associated resources on both nodes. Also, the user would be allowed to launch as many application instances as there are hyper-threads. Default: 1 (enabled)
claimsPerRes	Maximum number of concurrent claims (<code>aprun</code> instances) allowed per reservation. Default: 1000
suspendResume	Determines whether suspend/resume is enabled. Valid values are 0 (disabled), 1 (enabled in Phase 2 mode), and -1 (enabled in Phase 1 mode). Suspend/resume enables applications to oversubscribe compute node CPUs; BASIL provides a switching mechanism that enables one job to suspend another job on the node. In Phase 1 mode a maximum of two co-resident jobs is permitted; in Phase 2 mode, this limit is raised to four.
loadLimit	Determines the maximum number of applications per node permitted if suspend/resume is enabled. Valid values are 2 or 4. Default: 4
fakeNumaNodes	Determines the number or size of compute node fake NUMA nodes. If set as an integer, this is interpreted as the number; if set as a string (i.e., 2G), this is interpreted as a size. This must be configured identically on the compute nodes via the kernel parameter <code>numa=fake=number</code> or <code>numa=fake=size</code> .

nttSize	If set to 0, applications will not use the NTT (Network Translation Table). This parameter is used primarily for testing purposes.
pTagGlobalNodes	The minimum application size in nodes that forces ALPS to assign a global pTag value. If set, applications placed on a number of nodes equal to or greater than this value will use a global pTag, and not the NTT. This value is used on Gemini systems only.
pTagFreeDelay	Specifies the number of seconds to delay allocation of a pTag used previously. (The current PTag allocation implementation rotates through the pTags range.) This value is used on Gemini systems only. Default: 30 seconds
pKeyFreeDelay	Specifies the number of seconds to delay freeing a pKey used previously. This value is used on Aries systems only. Default: 30 seconds
pDomainMax	Specifies the maximum number of concurrent preallocated protection domains allowed. The absolute maximum is 256 and <code>apsched</code> will silently truncate higher values to 256. Default: 0 (disabled)
pDomainIDs	A comma-separated list of one or more system service dedicated protection domain identifiers.
sharedDir	(Required) Specifies the directory path to the file that contains ALPS control data. If <code>ALPS_MOUNT_SHARED_FS</code> is set to <code>yes</code> , this is assumed to be a mount point. Default: <code>/ufs/alps_shared</code>

apsys Configuration Parameters

These configuration parameters affect `apsys` only.

debug	This field is set to a default level of 1 for both <code>apsched</code> and <code>apsys</code> . For information about valid values, see the <code>apsched(8)</code> and <code>apsys(8)</code> man pages. Default: 1
aruEnable	Set to <code>Y</code> to generate Application Resource Utilization (ARU) files. Default: not set
aruPath	Specify the path name template for the ARU file. The <code>apid</code> will be appended to the specified file name when the file is created. This value is ignored if <code>aruEnable</code> is not set to <code>Y</code> .
prologPath	Specifies the location of a file to be executed before an application is launched. If this variable is not specified, nothing will be run before the <code>aprun</code> commands. Default: not set
epilogPath	Specifies the location of a file to be executed after an application exits. If this variable is not specified, nothing will be run after the <code>aprun</code> commands.

Default: not set

prologTimeout Specifies the maximum number of seconds to allow for the prolog to run. If it runs longer, the prolog will be terminated and an error will be returned. If this variable is not specified, the prolog will be given unlimited time to complete.

Default: not set

epilogTimeout Specifies the maximum number of seconds to allow for the epilog to run. If it runs longer, the epilog will be terminated. If this variable is not specified, the epilog will be given unlimited time to complete.

Default: not set

aprun Configuration Parameters

These configuration parameters affect aprun only. Each of these settings creates an association between an ALPS value and a SLURM workload manager environment variable that will be defined on the compute nodes.

defineNid	Corresponding environment variable: <i>SLURM_NODEID</i>
defineEachID	Corresponding environment variable: <i>SLURM_PROCID</i>
defineNPPN	Corresponding environment variable: <i>SLURM_TASKS_PER_NODE</i>
defineLocalEnt	Corresponding environment variable: <i>SLURM_LOCALID</i>

apstat Configuration Parameters

These values can be used to customize the apstat reports. Each configuration parameter is followed by a comma-separated list defining the data elements to be included in the report. The data elements are described in the apstat(1) man page.

nodeTable	Default configuration: <i>NID, Arch, State, HW, Rv, Pl, PgSz, Avl, Conf, Placed, PEs, Apids</i>
appsTable	Default configuration: <i>ApId, ResId, User, PEs, Nodes, Age, State, Command</i>
resvTable	Default configuration: <i>ResId, ApId, From, Arch, PEs, N, d, Memory, State</i>
pendingAppsTable	Default configuration: <i>PId, User, w:d:N, Nid, Age, Command, Why</i>
gpuTable	Default configuration: <i>NID, Module, State, Memory (MB) , Family, ResId</i>
pDomainTable	Default configuration (Gemini): <i>PDomainID, Type, Uid, PTag, Cookie</i> Default configuration (Aries): <i>PDomainID, Type, Uid, Cookie, Cookie2</i>

Application and Reservation Cleanup Configuration Parameters

The application and reservation cleanup functions have similar configuration value fields but can be configured separately.

configured (Reservation cleanup only)	On or off. (Application cleanup is always on.) Default: on
reports	If on, saves information about ongoing cleanups for use by the <code>apstat -C</code> command. Default: on
reportWait	The time in milliseconds to wait for node health before writing a report. Default: 2000
iterationSleep	The time in milliseconds to pause between cleanup iterations. Default: 1000
iterationMax	The maximum number of cleanup iterations to execute before giving up. Default: 10
connectTimeout	The time in milliseconds to allow an inter-node connection to be established. Default: 1000
connectAttempts	The maximum number of inter-node connection attempts to execute before giving up. Default: 5
waitMin	The minimum time in milliseconds to give any cleanup iteration to complete. The maximum wait time is calculated from this and the other cleanup parameters. Default: 5000

Suspend/Resume

Suspend/resume enables users to oversubscribe compute node CPUs, and provides a mechanism (via the batch system and BASIL) whereby applications can put other applications in a suspended state and take over use of the node. Note that suspend/resume enables users to oversubscribe compute node CPUs, not node memory. The amount of memory available to any given application is inversely proportional to the number of applications placed on the node: in other words, if two applications are placed on a node, each can use no more than half the node's memory.

When suspend/resume is enabled, the default `apsched` behavior changes from "exclusive" to "shared," unless a node reservation is explicitly exclusive; i.e., if the batch system reserves the node in interactive mode. When transitioning to or from suspend/resume, `apinit` clears any hugepage minimum values, and while suspend/resume is enabled, the `aprun -F exclusive, -r numcores, -ss,` and other NUMA node arguments are ignored.

Finally, note that NVIDIA GPU drivers do not support suspending and resuming jobs. Attempting to suspend an application that makes use of GPU resources may result in unpredictable results including abnormal program termination.

Resynchronize ALPS and the SDB Command After Manually Changing the SDB

Manual changes to node attributes and status can be reflected in ALPS by using the `apmgr resync` command. The `apmgr resync` command requests ALPS to reevaluate the configuration and attribute information and update its information. For example, after making manual changes to the SDB using the `xtprocadmin -e` or `xtprocadmin --noevent` command, use the `apmgr resync` command so that ALPS becomes aware of the changes.

Display Reserved Resources

The `apstat -r` command displays the batch job ID in the `From` field; for executables launched interactively, `apstat` displays `aprun` in the `From` field:

```
% apstat -r
  ResId   ApId  From           Arch   PEs N d Memory State
A  140   2497406 batch:741789   XT    512 - -  1333 conf,claim
   141   2497405 batch:741790   XT    768 24 1  1333 NID list,conf,claim
A  141   2497407 batch:741790   XT    768 - -  1333 conf,claim
```

The `apstat -A apid` command filters information by application IDs. Multiple application IDs can be include in a space-separated list. For example:

```
% apstat -avv -A 3848874
Total (filtered) placed applications: 1
Placed  Apid ResId   User   PEs Nodes   Age   State Command
      3848874 1620 crayuser  512   22   0h07m  run  dnp3+pat

Application detail
Ap[0]: apid 3848874, pagg 0x2907, resId 1620, user crayuser,
      gid 1037, account 0, time 0, normal
Reservation flags = 0x2001
Created at Tue Jul 12 14:20:08 2011
Originator: aprun on NID 8, pid 6369
Number of commands 1, control network fanout 32
Network: pTag 131, cookie 0xfb860000, NTTgran/entries 1/22, hugePageSz 2M
Cmd[0]: dnp3+pat -n 512, 1365MB, XT, nodes 22
Placement list entries: 512
Placement list: 6-7,11,20-21,24-25,36-39,56-59,69-71,88-91
```

The `apstat -R resid` command filters information about reservation IDs. Multiple reservation IDs can be included in a space-separated list. For example:

```
% apstat -rvv -R 1620
  ResId   ApId  From           Arch   PEs N d Memory State
  1620   3848874 aprun           XT    512 0 1  1365 atomic,conf,claim
```

```

Reservation detail for resid 1620
Res[1]: apid 3848874, pagg 0, resId 619, user crayuser,
      gid 1037, account 8944, time 0, normal
Batch System ID = 1971375
Created at Tue Jul 12 14:20:08 2011
Number of commands 1, control network fanout 32
Cmd[0]: dnsp3+pat -n 512, 1365MB, XT, nodes 22
Reservation list entries: 512
Reservation list: 6-7,11,20-21,24-25,36-39,56-59,69-71,88-91

```

Release a Reserved System Service Protection Domain

1. Ensure that no applications are running or reboot the system.
2. Remove the desired identifier from the `/etc/opt/cray/alps/alps.conf` `pDomainIDs` list.
3. Send a `SIGHUP` signal to `apsched` so that it restarts and reads the new `pDomainIDs` list.

Set a Compute Node to Batch or Interactive Mode

To set a node to be either batch or interactive mode, use the `xtprocaadmin` command to set the `alloc_mode` column of the `SDB processor` table. Then execute the `apmgr resync` command so that ALPS becomes aware of the changes.

Retrieve node allocation status

The `apstat -n` command displays the application placement status of nodes that have a state of `UP` and their allocation mode (`B` for batch or `I` for interactive) in the `State` column.

The `apstat` utility does not have dynamic run-time information about an application, so an `apstat` display does not imply anything about the running state of an application. An `apstat` display indicates statically that an application was placed and that the `aprun` claim against the reserved resources has not yet been released.

```

% apstat -n
  NID Arch State HW Rv Pl  PgSz  Avl  Conf  Placed  PEs  Apids
  20  XT UP  B 12 - -  4K 3072000  0  0  0
  21  XT UP  B 12 - -  4K 3072000  0  0  0
  22  XT UP  B 12 - -  4K 3072000  0  0  0
  23  XT UP  B 12 - -  4K 3072000  0  0  0
<snip>
  63  XT UP  B 12 12 12  4K 3072000 3072000 1572864 12 221180
  64  XT UP  B 12 12 12  4K 3072000 3072000 1572864 12 221180
  65  XT UP  B 12 12 12  4K 3072000 3072000 1572864 12 221180
  66  XT UP  B 12 12 12  4K 3072000 3072000 1572864 12 221182
<snip>
Compute node summary

```

arch	config	up	use	held	avail	down
XT	744	744	46	12	686	0

Manually Start, Stop, or Restart ALPS Daemons on a Specific Service Node

Prerequisites

- Must be logged on to the specific service node as `root`.

ALPS is automatically loaded and started when CNL is booted on compute nodes. The system administrator can manually start, stop, and restart daemons as necessary.

- Start ALPS daemons on a specific service node:

```
boot:~ # /etc/init.d/alps start
```

- Stop ALPS daemons on a specific service node:

```
boot:~ # /etc/init.d/alps stop
```

- Restart ALPS daemons on a specific service node:

```
boot:~ # /etc/init.d/alps restart
```

The `/etc/init.d/alps restart` command stops and then starts the ALPS daemons on the node.

Manually Clean Up ALPS, PBS, Moab/TORQUE After a Login Node Goes Down

If a login node goes down and will not be rebooted, job reservations associated with jobs deleted with `qdel` may not be released by ALPS. In this case, the `apstat -r` command lists the reservations as state `pendCancel` and leaves the jobs orphaned. Use the following procedure to manually clean up ALPS and the workload manager.

- Verify that the batch job still appears in the `qstat` output.

```
crayadm@smw:~> qstat -as 106728.sdb
```

```
sdb:
```

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	Elap S	Time
106728.sdb	root	workq	qsub.scrip	6231	1	1	--	--	R	00:00
Job run at Thu Dec 03 at 14:31 on (login1:ncpus=1)										

- Purge job from the workload manager and verify that it was purged. On the SDB node, type:

```
sdb:~ # qdel -W force 106728.sdb
```

- Verify that the job no longer exists.

```
sdb:~ # qstat -as 106728.sdb
qstat: Unknown Job Id 106728.sdb
```

- Restart apsched on the SDB node:

```
sdb:~ # /etc/init.d/alps restart
```

- Use apmgr to cancel the reservation that still exists in ALPS.

```
sdb:~ # apstat -r | grep 106728
ResId    ApId From           Arch    PEs N d Memory State
   5    2949806 batch:106728    XT      1 0 1    500 conf
sdb:~ # apmgr cancel 5
```

- Use apstat to verify that the reservation no longer exists.

```
sdb:~ # apstat -r | grep 106728
sdb:~ #
```

Verify that ALPS is Communicating with Cray System Compute Nodes

Executing the following aprun command on a login node will return a list of host names of the Cray system compute nodes used to execute the last program.

```
login:~> cd /tmp
login:/tmp> aprun -b -n 16 -N 1 /bin/cat /proc/sys/kernel/hostname
```

When an Application Terminates

The Node Health Checker (NHC) is invoked automatically upon the termination of an application or the end of a batch system reservation. Cleanup is therefore a two-stage process:

- Application cleanup is performed following an application exit.
- Reservation-level cleanup is performed following the termination of a batch system reservation.

System resources cannot be freed for reallocation until both the application cleanup and reservation cleanup have completed successfully.

Application Cleanup

During normal operations, an application exit is considered *orderly* when all of the application processes on a compute node have exited completely. ALPS gathers and consolidates all of this local exit information from each of the compute nodes within the application list; the exit information is sent to `aprun` over the ALPS application-specific TCP fan-out tree control network. When all of the exit information from all of the nodes is received by `aprun`, the `aprun` utility forwards the compiled information to `apsys` just before `aprun` itself exits.

Once all exit information has been received from the compute nodes, the application exit is considered orderly. An orderly exit does not necessarily mean that the application completed successfully, merely that exit information about the application was received by `aprun` and forwarded to `apsys`. The `apsys` utility then sends an exit message to `apsched`, which begins reservation-level cleanup.

An *unorderly* exit means that the node exit information has not been received by `apsys` before `aprun` exits. A typical example of an unorderly exit is when a `SIGKILL` signal is sent to `aprun` by the batch system after the application's wall time limit is exceeded. Since no exit information is available to `apsys` in an unorderly exit, `apsys` does not know the true state of the application processes on the compute nodes. Therefore, ALPS must perform application cleanup on each of the assigned compute nodes before it is safe to begin reservation cleanup.

Cleanup uses a tree-based overlay network formed using the `apinit` daemons on compute nodes associated with an unorderly exit to deliver a `SIGKILL` signal to application processes and to query for application presence. The overlay network is separate from the ALPS launch fan-out tree. All compute nodes that have a lingering application presence and all compute nodes with an unknown application presence status are gathered and used to inform the cleanup algorithm when to complete.

In the `apsys` log file, compute nodes that have a lingering application presence are reported in a `Match` list. Compute nodes with an unknown application presence status are reported in an `Unreached` list. The following example indicates that `apid 227061` remains resident on only one compute node (node 20), and that application presence status information has been received from all compute nodes:

```
14:00:21: Target Nodes: Match list portion for apid 227061 (1/1): 20
14:00:21: Target Nodes: Unreached list portion for apid 227061 (0/0):
```

Application cleanup begins with ALPS contacting each assigned compute node and sending a `SIGKILL` signal to any remaining application processes. Node health monitoring checks compute node conditions and marks a compute node `admindown` if it detects a problem. ALPS cannot free the node for reallocation until all of the application processes have exited or node health monitoring has marked the affected compute nodes either `admindown` or `suspect`. Until that time, the application will continue to appear in `apstat` displays.

After cleanup completes, or after every iteration of cleanup starting with the third iteration, the `xtcleanup_after` script is invoked.

Reservation-level Cleanup

Every time an application is launched on a set of compute nodes, ALPS creates a reservation tracker. When a reservation is terminated, either explicitly by the workload manager, implicitly upon orderly exit of the application, or via the `apmgr cancel` command, the information maintained by the reservation tracker is used to perform reservation-level cleanup.

Reservation-level cleanup is a two-step process, and largely transparent to the user. In the first step, the reservation is placed into a pending cancel state, regardless of the number of claims associated with the reservation or how the reservation was created. The `apsched` daemon then determines whether there are claims associated with the reservation, and if there are none, `apsched` instructs `apsys` to proceed with reservation

cleanup. If there are claims associated with the reservation, then `apsys` performs application cleanup and keeps the reservation in pending cancel state until all application processes exit or the node in question is marked `admindown`.

In the second step, every node associated with the reservation and with an `up` status is targeted for reservation removal. The `apsys` agent then uses the `alpsc_cleanup` library to remove the reservation. When this task is complete, `apsys` communicates to `apsched` that the reservation has been cleaned up, after which `apsched` removes the reservation from the ALPS shared state and sends confirmation back to `apsys`. The node is then returned to the pool of system resources available for new reservations.

aprun Actions

The `aprun` command is the ALPS application launch command on login nodes and the SDB node. The `aprun` command has a persistent TCP connection to a local `apsys`, as well as a persistent TCP connection to an `apinit` daemon child on the first compute node with in the assigned placement list, but not to an `apinit` on each assigned compute node.

After receiving a placement list from `apsched`, `aprun` writes information into the `syslog`, as in the example below.

```
May 18 10:38:16 nid00256 aprun[22477]: apid=1985825, Starting, user=10320,
batch_id=2325008, cmd_line="aprun -n 1 -b /tmp/hostname.xx ",
num_nodes=1, node_list=384

May 18 10:38:16 nid00256 aprun[22477]: apid=1985825, Error, user=10320,
batch_id=2325008, [NID 00384] 2010-05-18 10:38:15 Apid 1985825: cannot
execute: exit(107) exec failed

May 18 10:38:17 nid00256 apsys[22480]: apid=1985825, Finishing, user=10320,
batch_id=2325008
```

In a typical case of an orderly exit, `aprun` receives application exit information over the connection from that `apinit`. `aprun` then forwards the exit information over the connection to `apsys`. The ordering of application exit signals and exit codes is arbitrary. The `aprun` command displays any nonzero application exit information and uses the application exit information to determine its own exit code:

Application 284004 exit signals: Terminated

In the case of an unorderly exit, `aprun` exits without receiving application exit information. When `aprun` exits, its TCP connections are closed. The socket closes trigger application cleanup activity by both `apinit` and `apsys`.

An unorderly exit may occur for a variety of reasons, for example:

- The batch system sends a `SIGKILL` signal to `aprun` due to the application wall time expiring
- The `apkill` or `kill` commands are used to send a `SIGKILL` signal to `aprun`
- The `aprun` command receives a fatal message from `apinit` due to some fatal error during launch or at other points during the application lifetime, causing `aprun` to write the message to `stderr` and exit
- The `aprun` command receives a fatal read, write or unexpected close error on the TCP socket it uses to communicate with `apinit`

apinit Actions

The `apinit` daemon is the ALPS privileged daemon that launches and manages applications on compute nodes. For each application, the `apinit` daemon forks a child `apshepherd` process. Within `ps` displays, the child `apshepherd` processes retain the name "`apinit`".

The per-application TCP fan-out control tree has `aprun` as the root. Each compute node `apshepherd` within this control tree has a parent controller and may have a set of controlling nodes. Whenever a parent controller socket connection closes, the local `apshepherd` attempts to kill any application processes still executing and then will exit. This socket closing process results in a ripple effect through the fan-out control tree, resulting in automatic application tear down.

Whenever the `aprun` TCP connection to the `apshepherd` on the first compute node within the placement list closes, the tear-down process begins. During an application orderly exit, the exit information is sent to `aprun`, followed by the `aprun` closure of the socket connection, resulting in the exit of the `apshepherd`. The `apshepherd` exit causes its controlling socket connections to close as well. Each of those `apshepherds` will exit, and the application specific fan-out tree shuts down.

When the `aprun` TCP socket closure is not expected and the application processes are still executing, the `apshepherd` will send a `SIGKILL` signal to each local application process and then exit. There can be local delays in kernel delivery of the `SIGKILL` signal to the application processes due to application I/O activity. The application process will process the `SIGKILL` signal after the I/O completes. The `apinit` daemon is then responsible to monitor any remaining application processes.

This kill and exit process ripples throughout the control tree. However, if any compute node within the control tree is unresponsive, the ripple effect will stop for any compute nodes beyond that branch portion of the tree. In response to this situation, ALPS must take action independent of the shutdown of the control tree to ensure all of the application processes have exited or that compute nodes are marked either `admindown` or `suspect` by node health monitoring. The `apsys` daemon is involved in invoking the independent action.

apsys Actions

The `apsys` daemon is a local privileged ALPS daemon that runs on each login node and the SDB node. When contacted by `aprun`, the `apsys` daemon forks a child agent process to handle that specific local `aprun`. The `apsys` agent provides a privileged communication path between `aprun` and `apsched` for placement and exit information exchanges. The `apsys` agent name remains "`apsys`" within `ps` displays.

During an orderly application exit, the `apsys` agent receives exit information from `aprun` and forwards that information to `apsched`. However, during an unordered exit, when the `aprun` socket connection closes prior to receipt of exit information, the `apsys` agent is responsible to start application cleanup on the assigned compute nodes.

To begin application cleanup, the `apsys` agent invokes cleanup and the `apsys` agent blocks until cleanup completes.

Node Health Checker Actions

The Node Health Checker (NHC) is automatically invoked by ALPS upon the termination of an application. ALPS passes a list of nodes associated with the terminated application to NHC. NHC performs tests, which are specified in the NHC configuration file, to determine if compute nodes allocated to the application are healthy enough to support running subsequent applications. In particular, NHC attempts to reboot any nodes marked `admindown` or `suspect`. NHC removes any nodes incapable of running an application from the resource pool.

NHC verifies that the Application Level Placement Scheduler (ALPS) acknowledges a change that NHC has made to a node's state. If ALPS does not acknowledge a change, then NHC recognizes this disagreement between itself and ALPS. NHC then changes the node's state to `admindown` state and exits.

For an overview of NHC, see the `intro_NHC(8)` man page. For additional information about configuring node health checker, see [Configure the Node Health Checker \(NHC\)](#) on page 120.

Verify Cleanup

There are a number of circumstances that can delay completion of application cleanup after an unorderly exit. This delay is often detected through `apstat` displays that still show the application and the resource reservation for that application.

As described in previous sections, check the various log files to understand what activity has taken place for a specific application.

- Check the `/var/opt/cray/alps/log/apsys/YYYYMMDD` log files for that `apid`; verify cleanup has been invoked.
- Check the applicable node health monitoring log file (`/var/log/xtcheckhealth_log`) for that `apid`.
- Check the SMW `/var/opt/cray/log/sessionid/console-YYMMDDHHMM` log file for that `apid`.

Comprehensive System Accounting

NOTICE: With the release of CLE 5.1UP00 and CLE 4.2UP02, the Resource Utilization Reporting (RUR) software was added. RUR is an administrator tool that gathers statistics about how Cray systems are being used by applications. As a result of adding RUR, Comprehensive System Accounting is deprecated and will be removed in a future release. Users of CSA are encouraged to migrate to RUR. For further information on RUR, see [Resource Utilization Reporting](#) on page 226.

Comprehensive System Accounting (CSA) is open-source software that includes changes to the Linux kernel so that the CSA can collect more types of system resource usage data than under standard Fourth Berkeley Software Distribution (BSD) process accounting. CSA software also contains interfaces for the Linux process aggregates (`paggs`) and jobs software packages. The CSA software package includes accounting utilities that perform standard types of system accounting processing on the CSA generated accounting files. CSA, with Cray modifications, provides:

- Project accounting capabilities, which provide a way to charge computer system resources to specific projects
- An interface with various other job management systems in use at Cray sites
- A data management system for collecting and reporting accounting data
- An interface that you use to create the project account and user account databases, and to later modify them, as needed
- An interface that allows the project database to use customer-supplied user, account, and project information that resides on a separate Lightweight Directory Access Protocol (LDAP) server
- An interface with the ALPS application management systems so that application accounting records that include application start, termination, and placement information can be entered into the system accounting database

Complete features and capabilities of CSA are described in the `csa(8)` and `intro_csa(8)` man pages. The accounting utilities provided for administrative use are: `csanodeacct`, `csaperiod`, and `csarun`. The related man pages are accessible by using the `man` command.

NOTE: CSA runs only on login nodes and compute nodes. The SMW, boot node, SDB node, Lustre MDS nodes, and Lustre OSS nodes do not support CSA.

Interacting with Batch Entry Systems or the PAM job Module

Jobs are created on the system using either a batch job entry system (when such a system is used to launch jobs) or by the PAM `job` module for interactive sessions.

NOTE: You must be running TORQUE snapshot (release) `2.4.0-snap.20080925140` or later to take advantage of CSA support for the Cray platform.

You must run PBS Professional 9.2 or later to take advantage of CSA support for the Cray platform.

Compute node project accounting for applications submitted through workload managers (for example, PBS Professional) depends on the ability of the workload manager to obtain and propagate the project ID to ALPS at job submission time. If the workload manager does not support the ability to obtain and propagate the project ID to ALPS at job submission, the project ID must be set by using the `account` command prior to issuing an ALPS `aprun` command. Otherwise, project ID information will not be included in any CSA accounting records for the job.

CSA Configuration File Values

The CSA configuration file, `csa.conf`, is included with the Cray Linux Environment (CLE) software release package. This file contains default settings for several configuration parameters you must change to tailor CSA to your individual site configuration. On Cray systems `csa.conf` is located on the shared root in `/etc/opt/cray/csa/csa.conf` for login nodes and on the SMW in `/opt/xt-images/templates/default/etc/opt/cray/csa/csa.conf` for compute nodes.

NOTE: The two copies of this file must be identical with the exception of the `NODE_PROCESS_ACCOUNT` parameter.

Each Cray system that runs CNL has its own unique hardware configuration, including the number of nodes on the system, the physical location of the nodes, and a unique file system configuration. For this reason, the default `csa.conf` files can only be used as a template. A new version of the CNL compute node image must be created after editing `csa.conf` in order to implement the changes.

The parameters shown in the following table are used to define the accounting file system configuration and the node configuration for your system. You must change the settings of these parameters so that they conform to your system configuration.

Table 8. CSA Parameters That Must Be Specific to Your System

Parameter	Description
<code>CSA_START</code>	Defines whether CSA is enabled (<code>on</code>) or disabled (<code>off</code>). By default, CSA is disabled.
<code>ACCT_SIO_NODES</code>	Declares the number of account file system mount points. There must be at least one account file system mount point. The maximum number of mount points is 10. Multiple mount points are allowed so that the individual node accounting files can be distributed across more than one file system in order to provide better scaling for large system configurations. Use the <code>df</code> command to display the possible file system mount points. The actual maximum number of <code>ACCT_SIO_NODES</code> that may be specified is limited by the number of file systems available on your system.
<code>ACCT_FILE_SYSTEM_00</code>	Must be one entry for each declared file system mount point. Numbering must begin with <code>00</code> , and numbers must be consecutive. For example, if you have specified <code>ACCT_SIO_NODES 1</code> , you will only define <code>ACCT_FILE_SYSTEM_00</code> . If you have specified <code>ACCT_SIO_NODES 2</code> , you will also need to define <code>ACCT_FILE_SYSTEM_01</code> .
<code>...</code>	
<code>ACCT_FILE_SYSTEM_nn</code>	
<code>_lus_nid00023_csa_XT</code>	The default file system mount point. It must be changed to correspond to a file system that exists on your system. There is one of these entries for each <code>ACCT_FILE_SYSTEM</code> declared.

Parameter	Description
	NOTE: The program that parses the configuration file does not allow any special characters, other than the underscore character (<code>_</code>) in configuration names. Therefore, in the file system paths used in the mount point description, each forward slash character (<code>/</code>) character must be represented by an underscore (<code>_</code>) character. This also means that an account file system mount point cannot have a <code>_</code> character in the pathname.
<code>SYSTEM_CSA_PATH</code>	Defines the pathname on the common file system where CSA establishes its working directories for generating accounting reports. This parameter is only used on the service node image. It is not used on the compute nodes.
<code>NODE_PROCESS_ACCOUNT</code>	Defines whether all process account records written on a node will be written to the common file system, or whether the process account records for each application will be combined into a single application summary record that represents the total execution of the application on a node. This parameter may be set differently on the shared root and compute node images.

For other parameters in `csa.conf`, default settings should be acceptable.

Configure CSA

CSA is disabled by default. When CSA is enabled, all system accounting, including service node accounting, is performed by CSA. Therefore, there is no need to have BSD process accounting enabled on service nodes. To enable CSA, set `CSA_START` to `on` in `csa.conf`.

NOTE: You must include the CSA RPM in your CNL boot image. If you set values in the `shell_bootimage.sh` script, make sure to edit the same values in `CLEinstall.conf` so that any new features remain enabled after the next CLE update or upgrade.

Perform the procedures in this section, in order, to correctly set up CSA.

Obtaining File System and Node Information

- From a login node, enter the `df` command to determine which file systems are available for writing CSA accounting data.

```
login:~ > df
```

```
rootfs                173031424 158929920   5311488   97% /
initramdevs           8268844      76   8268768    1% /dev
10.131.255.254:/rr/current
                    173031424 158929920   5311488   97% /
10.131.255.254:/rr/current/.shared/node/8/etc
                    173031424 158929920   5311488   97% /etc
10.131.255.254:/snv   48070496 13872768  31755840  31% /var
```

```

10.131.255.254:/snv 48070496 13872768 31755840 31% /var
none 8268844 12 8268832 1% /var/lock
none 8268844 940 8267904 1% /var/run
none 8268844 0 8268844 0% /var/tmp
tmpfs 8268844 12 8268832 1% /tmp
ufs:/ufs 38457344 26436608 10067968 73% /ufs
ufs:/ostest 20169728 10874880 8269824 57% /ostest
23@gni:/lus_system 215354196400 60192583820 144222067004 30% /lus/nid00011
30@gni:/ib54ex 114821632416 5588944 108983420416 1% /lus/nid00064

```

- Determine and record the file system information you want to use for CSA.

The files systems of interest for saving accounting data are those two systems whose mount points are /lus/nid00011 and /lus/nid00064, respectively. Record this information for later use.

- Determine the hardware node configuration on your system.

Run the `xtprocdadmin` command to get a complete list of nodes.

```
login:~ > xtprocdadmin
```

NID	(HEX)	NODENAME	TYPE	STATUS	MODE	PSLOTS	FREE
0	0x0	c0-0c0s0n0	service	up	batch	4	0
3	0x3	c0-0c0s0n3	service	up	batch	4	0
4	0x4	c0-0c0s1n0	service	up	batch	4	4
7	0x7	c0-0c0s1n3	service	up	batch	4	4
...							
475	0x1db	c3-0c2s6n3	compute	up	batch	4	4
476	0x1dc	c3-0c2s7n0	compute	up	batch	4	4
477	0x1dd	c3-0c2s7n1	compute	up	batch	4	4
478	0x1de	c3-0c2s7n2	compute	up	batch	4	4
479	0x1df	c3-0c2s7n3	compute	up	batch	4	4

For this example system, you want to choose a set of nodes that will have their accounting files written to /lus/nid00011 and another set of nodes that will have their accounting files written to /lus/nid00064. You also need to make sure there is no overlap between the two sets of nodes.

Editing the `csa.conf` File

After you have the file system mount point and node configuration information for your system, you are ready to edit the `csa.conf` file. On Cray systems this file is located on the shared root at `/etc/opt/cray/csa/csa.conf` for login nodes and on the SMW at `/opt/xt-images/templates/default/etc/opt/cray/csa/csa.conf` for compute nodes.

NOTE: You must use `xtopview` to edit the shared root image of `csa.conf` file on the boot node. You can use any text editor to edit the compute node image of `csa.conf` file on the SMW. If the `/opt/xt-images/templates/default/etc/opt/cray/csa/csa.conf` files does not exist on the SMW, you may copy the file from the shared root.

Editing Other System Configuration Files

You also must make configuration changes to other system files. Use the `xtopview` command on the boot node to make the changes. For detailed information about using `xtopview`, see [Manage the System Configuration with the `xtopview` Tool](#) on page 103 or the `xtopview(8)` man page.

- Add the `csaacct` user name to `/etc/passwd` on the shared root.

```
csaacct:*:391:391:CSA:/var/lib/csa:/sbin/nologin
```

- Add the `csaacct` group name to `/etc/group` on the shared root.

```
csaacct!:391:
```

- Update the shadow password file, `/usr/sbin/pwconv`, to reflect the changes you have made.
 - Add the `csaacct` group name to `/etc/group` on the CNL image.
- NOTE:** The `csaacct` group and `gid` must be the same on the shared root and CNL image.
- Create additional PAM entries in `/etc/pam.d/common-session` to enable CSA. For more information about creating PAM entries, see [Configure CSA Job Accounting for non-CCM CNOS Jobs](#) on page 223.

Creating a CNL Image with CSA Enabled

After you have modified the compute node copy of `csa.conf` on the SMW, you must rebuild the compute node image. For more information about how to rebuild the compute node image, see [Service Node and Compute Node Boot Image Preparation](#).

You can edit the shared root version of `csa.conf` and install the new version using the `xtopview` command. For more information about editing the shared root image of `csa.conf` using the `xtopview` utility, see [Manage the System Configuration with the xtopview Tool](#) on page 103 or the `xtopview(8)` man page.

Configure CSA Project Accounting

The project database allows your site to define project names and assign an account number to each project. Users can have a list of account numbers that they can use for charging computing resources. Each user has a default account number that is assigned at login time.

The project database resides on the system SDB node as a MySQL database. To set up a CSA project accounting for your system, perform this procedure.

If you do not want to use CSA project accounting, complete [Disabling project accounting](#) instead of this procedure.

1. Establish the project database, `UserProject`, and define the project database tables on the System Data Base (SDB) server:

```
sdb:~ # mysql -u root -h sdb -p < /opt/cray/projdb/default/sql/
create_UserProject.sql
```

2. Grant administrative access privileges to the project database:

```
sdb:~ # mysql -u root -h sdb -p < /opt/cray/projdb/default/sql/
create_accounts.sql
```

3. Use the `xtopview` command from the boot node to create and edit the `/etc/opt/cray/projdb/projects` file on the shared root so that it contains a list of valid account numbers and the associated project names.

The `/etc/opt/cray/projdb/projects` file consists of a list of entries where each entry contains a project number followed by a project name. A colon character separates the project number from the project name. A project number and an account number are the same thing. The following example shows a simple project file:

```
0:root
100:sysadm
101:ProjectA
102:ProjectB
103:Big Name Project that is insignificant and unimportant
1234567890:Big Name Project with Blanks in the Name
```

4. Use the `xtopview` command from the boot node to create and edit the `/etc/opt/cray/projdb/useracct` file on the shared root so that it contains a list of authorized users and the valid account numbers for each user.

Each line of the user accounts file contains the login name of a user and list of accounts that are valid for that user. The first account number in the list is the user's default account. The default account number is assigned to the user at login time by the `pam_job` module. The user name is separated from the first account ID by a colon (:). Additional account numbers are separated by a comma (,).

The following shows a simple user account file:

```
root:0
u1000:100
u1001:101,103
u1002:100,101
u1003:100,103,1234567890
```

5. On the login node, edit the `~crayadm/.my.cnf` file in the home directory of the project database administrator so that it contains the following lines:

```
[client]
user=sys_mgmt
password=sys_mgmt
host=sdb
```

6. On the login node, change the permissions and owner on the `~crayadm/.my.cnf` file, as follows:

```
login:/home/crayadm:~> chmod 600 ~crayadm/.my.cnf
login:/home/crayadm:~> chown crayadm:crayadm ~crayadm/.my.cnf
```

7. If you are using customer-supplied user, account, and project information that resides on a separate LDAP server, use the `xtopview` command from the boot node to edit the `/etc/opt/cray/projdb/projdb.conf` project accounting configuration file so that it contains site-specific values for the parameters listed in [Project Accounting Parameters That Must Be Specific to Your System](#).

Table 9. Project Accounting Parameters That Must Be Specific to Your System

Parameter	Description
PROJDBTYPE	If you are using customer-supplied user, account, and project information that resides on a separate LDAP server, change from <code>MYSQL</code> (default) to <code>CUSTOM</code> .

Parameter	Description
CUSTOM_VALIDATE	<p>If you are using customer-supplied user, account, and project information that resides on a separate LDAP server, specify the full path name to the customer-supplied function that performs the necessary validation, for example <code>/usr/local/sbin/validate_account</code>.</p> <p>Input parameters to the validation function are position order dependent, as follows:</p> <pre>user_name account_number</pre>

- On a login node, run the `projdb` command with the `-c` option to create the project database. After the project database has been established, any users gaining access to the system through the job PAM module are assigned a default account ID at the time of system access. The project database package commands are installed in `/opt/cray/projdb/default/bin`, which must be in your `PATH` variable to access the commands.

```
login:/home/crayadm:~> projdb -c -p /etc/opt/cray/projdb/projects -u /etc/opt/cray/projdb/useracct -v
```

Configure CSA Job Accounting for non-CCM CNOS Jobs

- Cluster Compatibility Mode (CCM) does not support CSA accounting. However, the CNOS class must support CSA accounting for non-CCM jobs. To accomplish this, use the following configuration.

```
# xtopview
default/:/# vi /etc/opt/cray/ccm/ccm_mounts.local

/etc/pam.d/common-session-ccm /etc/pam.d/common-session bind 0

default/:/# exit
```

- In CNOS Class view:

```
common-session includes:
session optional pam_mkhomedir.so skel=/software/skel
session required pam_limits.so
session required pam_unix2.so
session optional pam_ldap.so
session optional pam_umask.s
session optional /opt/cray/job/default/lib64/security/pam_job.so
common-session.ccm does not include the pam_job entry:
session optional pam_mkhomedir.so skel=/software/skel
session required pam_limits.so
session required pam_unix2.so
session optional pam_ldap.so
```

For the procedure to disable CSA for the CNOS class view, see *CLE User Application Placement Guide (S-2496)*.

For additional information about setting up job accounting on your system, read the `INSTALL` file that is included in the `job` RPM.

For more information about editing the shared root image of the `pam` configuration files using the `xtopview` utility, see [Manage the System Configuration with the xtopview Tool](#) on page 103 or the `xtopview(8)` man page.

Creating Accounting cron Jobs

CSA depends on your system having a persistent `/var` file system for the shared root. For CSA to run successfully, you must establish the following cron jobs.

The normal order for the cron jobs is: `csanodeacct`, `csarun`, and then `csaperiod` (if necessary).

csanodeacct cron Job for Login Nodes

On Cray system compute nodes, when an application terminates, the Application Launch and Placement Scheduler (ALPS) initiates the CSA software that moves the local node accounting file records to a node-specific directory on the common file system (Lustre). On login nodes, this does not happen, and accounting records continue to accumulate indefinitely until the `csanodeacct` script is invoked to move the data to the common file system. Therefore, you need to periodically run a cron job on each login node to make sure that the local accounting files are moved as needed. This cron job must be owned by `root`.

Running a csanodeacct cron job on each login node to move local accounting files

The following example shows moving accounting files from the local file system to the common file system on an hourly basis at 10 minutes before the hour. This crontab must be executed for each login node:

```
50 * * * * /opt/cray/csa/default/sbin/csanodeacct
```

csarun cron Job

You normally execute the `csarun` script at defined intervals to generate a set of system accounting reports.

Executing the csarun script

To run `csarun` once per day at one minute before midnight, use a crontab entry of the following form:

```
59 23 * * * /opt/cray/csa/default/sbin/csarun
```

NOTE: This crontab must be executed from only **one** login node since it executes the `csanodemerg` script that merges all of the local node accounting files into a single system wide accounting file.

csaperiod cron Job

You can invoke the `csaperiod` script to run periodic accounting at different intervals than the regular system accounting interval.

Running periodic accounting at different intervals than the regular system accounting interval

To run `csaperiod` once a week on Sunday at 5 minutes after midnight, use a crontab entry of the following form:

```
5 0 * * 0 /opt/cray/csa/default/sbin/csaperiod
```

NOTE: This crontab must be executed from only **one** login node since it executes the `csanodemerg` script that merges all of the local node accounting files into a single system-wide accounting file.

Enabling CSA

Using the `xtopview` command on the boot node is the only method to configure, enable, or disable services on the shared-root file system. You cannot configure, enable, or disable services on the login node itself. If your site has configured a `login` class for your system, invoke the following command sequence from the boot node as `root`:

```
boot# xtopview -x /etc/opt/cray/sdb/node_classes -c login
class/login:/# chkconfig job on
class/login:/# chkconfig csa on
class/login:/# xtspec -c login /etc/init.d/job
class/login:/# xtspec -c login /etc/init.d/csa
class/login:/# exit
```

On the subsequent system boot, this starts up the specified services on all nodes of the `login` class.

NOTE: If your site has not configured a `login` class, you must enable CSA for the individual login nodes using the `xtopview -n [nid#]` syntax rather than the `xtopview -c login` syntax shown. You must repeat the process for each login node. See the `xtopview(8)` man page for complete command option information.

Using LDAP with CSA

The `projdb` command and the `-l` option on the `account` command are not supported with customer-provided account validation.

The following Cray supplied library functions do not support this feature: `db_add_project`, `db_add_user`, `db_get_proj_acct`, `db_get_proj_name`, `db_get_user_accts`, `db_has_table`, `db_print_table`, `db_truncate_table`, and `db_validate_acct`.

For a description of the `/etc/opt/cray/projdb/projdb.conf` file and additional information on using a customer-supplied database, see the `projdb(8)` and `intro_csa(8)` man pages.

Resource Utilization Reporting

Resource Utilization Reporting (RUR) is an administrator tool for gathering statistics on how system resources are being used by applications or jobs. RUR is a low-noise, scalable infrastructure that collects compute node statistics before an application or job runs and again after it completes. The extensible RUR infrastructure allows plugins to be easily written to collect data uniquely interesting to each site. Cray supplied plugins collect a variety of data, including process accounting, energy usage, memory usage, and GPU accounting.

When RUR is enabled on a Cray system running CLE, resource utilization statistics are gathered from compute nodes running all applications or jobs. RUR is configured to run per application, per job, or both. RUR runs primarily before an application/job has started and after it ends, ensuring minimal impact on performance.

Prior to application/job runtime, the ALPS or WLM prologue script calls an RUR prologue script that, based on enabled plugins, initiates pre-application/pre-job data staging on all compute nodes used by the application/job. This staging may involve resetting counters to zero or collecting initial values of counters. Following application/job completion, the ALPS or WLM epilogue script calls an RUR epilogue script that gathers these counters, compares them to the initial values, where applicable, stages the data on the compute nodes, and then transfers data from the compute nodes to the login/MOM node. RUR post-processes the data to create a summary report that is written out to a log file or other backing store.

Plugin Architecture

RUR supports a plugin architecture, allowing many types of usage data to be collected while using the same software infrastructure. Two basic types of RUR plugins are supported: *data plugins*, which collect particular statistics about system resources, and *output plugins*, which send the output of the RUR software stack to a backing store.

Cray supplies plugins as part of the RUR distribution, including six data collection plugins, three output plugins, and one example plugin. Sites choose which plugins to enable or disable by modifying the RUR configuration file. See [Enable/Disable Plugins](#) on page 229 for more information. Sites can also create custom plugins, specific to their needs, as described in [Create Custom RUR Data Plugins](#) on page 241 and [Create Custom RUR Output Plugins](#) on page 243.

RUR Configuration File

The RUR configuration file `/etc/opt/cray/rur/rur.conf` is located on the shared root. The file consists of distinct sections, identified by a header, `[section]`, that contain settings for specific RUR components. Changing the behavior of RUR components is possible through modification of the config file. Configurable values fall into five categories:

- Script/binary location
- Temporary data storage location
- Component timeout specs
- Enabling/disabling plugins
- Optional plugin behavior

Changes to the configuration file are automatically propagated to the nodes via the shared-root mount, requiring no administrator intervention or system reboot. An example configuration file `rur.config.example` is distributed with the RPM.

Enable Per-Application RUR

By default, RUR is installed but not enabled during the CLE installation process. Although RUR is not a part of ALPS, it is initiated through the ALPS prologue and epilogue scripts.

1. Edit the ALPS configuration file, `/etc/opt/cray/alps/alps.conf`, on the shared root and search for the `prologPath` and `epilogPath` parameters.
ALPS supports only one prologue script and one epilogue script; therefore, enabling RUR is dependent on whether or not these parameters are already defined for ALPS.

- a. If `prologPath` and `epilogPath` are not defined, modify the parameters as follow, where `config_set` is the name of an optional tag that indicates which version of a plugin to call. See [Define Multiple Plugin Versions Using config_sets](#) on page 230.

```

apysys
  prologPath    /opt/cray/rur/default/bin/rur_prologue.py [-s config_set,config_set,...]
  epilogPath    /opt/cray/rur/default/bin/rur_epilogue.py [-s config_set,config_set,...]
  prologTimeout 60
  epilogTimeout 60
/apysys

```

- b. For either parameter that is defined, a wrapper script must be written that will run both the ALPS script and the RUR script. Cray recommends adjusting the `prologTimeout` and `epilogTimeout` parameters to be the sum of the timeouts expected for the constituent scripts. Because RUR supports its own timeout, it is recommended to run RUR first, with a timeout, allowing the second plugin to run even if RUR times out.

2. Restart ALPS on the login node(s).

```
login:~ # /etc/init.d/alps restart
```

3. Verify that the `rur` argument is set to `True`, `yes`, `1`, or `enable` within the `[global]` section of the RUR configuration file.

```

[global]
rur: True

```

4. Verify that specific plugins called with `config_sets` are enabled.

Enable Per-Job RUR

By default, RUR is installed but not enabled during the CLE installation process. Although per-job RUR is not part of a WLM, it is initiated through the WLM prologue and epilogue scripts. Job level RUR data is identified by records with `apid: 0` and `jobid: ID_of_WLM_job`.

1. Edit the WLM prologue and epilogue scripts, according to the specific WLM system guidelines, to call the `zur_prologue` and `zur_epilogue` scripts, respectively; where `jobid` is the WLM jobid and `config_set` is the name of an optional tag that indicates which version of a plugin to call. See [Define Multiple Plugin Versions Using config_sets](#) on page 230 for further information.

```
/opt/cray/rur/default/bin/rur_prologue.py -j jobid [-s config_set,config_set,...]
```

```
/opt/cray/rur/default/bin/rur_epilogue.py -j jobid [-s config_set,config_set,...]
```

2. Consult the specific WLM documentation with regards to restarting the WLM.
3. Verify that the `zur` argument is set to `True`, `yes`, `1`, or `enable` within the `[global]` section of the RUR configuration file.

```
[global]
zur: True
```

4. Verify that specific plugins called with `config_sets` are enabled.

Enable Performance-Enhanced RUR

By default, RUR runs Python scripts at application/job termination on all compute nodes in the application/job. This requires loading Python, the Python standard library, RUR scripts and the configuration file on each node. As the number of nodes in the application scale, the load on the `/ds1` file system grows and can significantly affect performance. A performance-enhanced version of RUR is available. It uses PyInstaller to create binaries and shared libraries that are installed in the compute node boot image, thereby reducing the `/ds1` file system load and improving performance.

Limitations: The `KNCstats` plugin is not supported by performance-enhanced RUR.

1. Edit the `/etc/opt/cray/rur/rur.conf` file on the shared root.
2. Set `compute_local_python` to `True`, `Yes`, `1` or `enable` within the `[global]` section.

```
[global]
zur: True
compute_local_python: True
```

Systems configured with site-written RUR plugins (i.e., not provided by Cray) must complete additional steps to include these plugins in performance-enhanced RUR.

3. For plugins written in Python:
 - a. Generate binaries and shared libraries using PyInstaller.
 - b. Install them into the compute node boot image.
 - c. Modify the RUR configuration file, adjusting the path of the plugin staging component to reflect the new path name and/or file type.

4. For binary or shell script plugins:
 - a. Install them into the compute node boot image.
 - b. Modify the RUR configuration file, adjusting the path of the plugin staging component to reflect the new path name and/or file type.

Disable RUR

1. Edit `/etc/opt/cray/rur/rur.conf` on the shared root.
2. Disable RUR by setting the `rur` argument to `False`, `no`, `0`, or `disable` within the `[global]` section.

```
[global]
rur: false
```

Enable/Disable Plugins

1. Edit the `/etc/opt/cray/rur/rur.conf` file on the shared root.
2. To enable a plugin, list it in the `[plugins]` section (for data plugins) or `[outputplugins]` section (for output plugins) and set to `true`.

```
[plugins]
taskstats: true
```

3. To disable a plugin, either remove it from the list or set it to `false`.

```
[plugins]
taskstats: false
```

For example, the following `[plugins]` section from the configuration file explicitly enables data plugins `taskstats` and `timestamp`, but disables `gpustat`. Additionally, plugins `energy`, `kncstats`, and `memory` are implicitly disabled by their absence from the list.

```
[plugins]
gpustat: false
taskstats: true
timestamp: true
```

Enable Plugin Options

Many Cray supplied plugins include options that change the plugin's behavior, such as amount of data reported or location of an output file. Follow the procedure to enable a plugin's optional behavior.

1. Edit the `/etc/opt/cray/rur/rur.conf` file on the shared root.
2. Add the following `arg` line in the appropriate `[plugin]` section, where `option` is the plugin-specific value for the desired behavior.

```
arg: option[,option,...]
```

For example, to select `memory` plugin's `extended_buddy` option, add `arg: extended_buddy` in the `[memory]` section as shown.

```
[memory]
stage: /opt/cray/rur/default/bin/memory_stage.py
post:  /opt/cray/rur/default/bin/memory_post.py
arg:   extended_buddy
```

3. Verify that the desired plugin has been enabled in either the `[plugins]` or `[outputplugins]` section of the RUR configuration file, as described in [Enable/Disable Plugins](#) on page 229.

Define Multiple Plugin Versions Using `config_sets`

A plugin can be defined multiple times with different options by tagging it with a `config_sets` argument line. A site might choose to do this, for example, if the output format desired for per-job reporting is different than that desired for per-application reporting.

1. Edit the `/etc/opt/cray/rur/rur.conf` file on the shared root.
2. Create a copy of the specific plugin's definition section. The name of the duplicate plugin can be modified but it is not required.
3. Edit both plugin sections to enable the desired options, as necessary. See [Enable Plugin Options](#) on page 229 for details.
4. Add the following `config_sets` line to at least one of the plugins, where `config_set` is the tag name that will be used when invoking RUR such that the plugins defined with this tag will be invoked.

```
config_sets: config_set[,config_set,...]
```

For example:

```
# energy plugin for per-application reporting (json-list format)
[energy]
stage: /opt/cray/rur/default/bin/energy_stage.py
post:  /opt/cray/rur/default/bin/energy_post.py
arg:   json-list
config_sets: alpset
```

```
# energy plugin for per-job reporting (json-dict format)
[energy]
stage: /opt/cray/rur/default/bin/energy_stage.py
post: /opt/cray/rur/default/bin/energy_post.py
arg: json-dict
config_sets: wlmset
```

5. Verify that the desired plugin(s) has been enabled in either the `plugins` or `outputplugins` section of the configuration file as described in [Enable/Disable Plugins](#) on page 229.

Limit the Report Generated by an Output Plugin

By default, an enabled output plugin reports data for all data plugins. The `data_sets` argument enables the administrator to limit the set of data plugins for which an output plugin reports. A site might choose to do this, for example, so that different output plugins report on different data plugins.

1. Edit the `/etc/opt/cray/rur/rur.conf` file on the shared root.
2. Add the following `data_sets` argument line to an output plugin section, where `data_plugin` is the name of a valid data plugin.

```
data_sets: data_plug[, data_plugin, ...]
```

For example, to limit the report generated by the `file` output plugin to `taskstats` and `memory`:

```
[file]
output: /opt/cray/rur/default/bin/file_output.py
#arg: path-to-flat-textfile
arg: /lus/scratch/RUR/output/rur.output
data_sets: taskstats, memory
```

3. Verify that the desired plugin is enabled in the `outputplugins` section of the configuration file as described in [Enable/Disable Plugins](#) on page 229.

RUR Debugging Level

The value of `debug_level`, if set, within the `[global]` section of the configuration file determines the level of debug verbosity written to the messages file on the SMW. Valid values are `debug`, `info`, `warning`, `error`, and `critical`.

The energy Data Plugin (Cray XC Series only)

The `energy` plugin collects compute node energy usage data. The amount of data reported and the format in which it is written is determined by the value of the argument `arg` set in the `[energy]` section of the RUR configuration file.

If `arg` is not set (default) or set to `json-list`, the plugin reports the following, written in JavaScript Object Notation (JSON) list format:

energy_used The total energy (joules) used across all nodes. On nodes with accelerators, this value includes `accel_energy_used`, the total energy used by the accelerators.

RUR default energy output

This example shows default `energy` data as written to `/var/opt/cray/log/partition-current/messages-date` on the SMW:

```
2013-08-30T11:19:06.545114-05:00 c0-0c0s2n2 RUR 18657
p2-20130829t090349 [RUR@34] uid: 12345, apid: 10963, jobid: 0,
cmdname: /opt/intel/vtune_xe_2013/bin64/amplxe-cl plugin: energy
['energy_used', 318]
```

If `arg` is set to `json-dict`, the plugin also reports the following extended energy data, written in JSON dictionary format:

error	If a Python exception occurs during the post or staging scripts, the following data is reported:
traceback	Stack frame list
type	Python exception type
value	Python exception parameter
nid	NID on which exception occurred
cname	cname on which exception occurred
nodes	Number of nodes in job
nodes_cpu_throttled	Number of nodes experiencing CPU power/thermal throttling
nodes_memory_throttled	Number of nodes experiencing memory power/thermal throttling
nodes_power_capped	Number of nodes with nonzero power cap
nodes_throttled	Number of nodes experiencing any of the following types of throttling: <ul style="list-style-type: none"> ▪ CPU power/thermal throttling ▪ Memory power/thermal throttling
nodes_with_changed_power_cap	Number of nodes with power caps that changed during execution. On nodes with accelerators, this value includes the number of accelerators with power caps that changed.
max_power_cap	Maximum nonzero power cap

<code>max_power_cap_count</code>	Number of nodes with the maximum nonzero power cap
<code>min_power_cap</code>	Minimum nonzero power cap
<code>min_power_cap_count</code>	Number of nodes with the minimum nonzero power cap

On nodes with accelerators, the extended data also include the following data:

<code>accel_energy_used</code>	Total accelerator energy (joules) used
<code>nodes_accel_power_capped</code>	Number of accelerators with nonzero power cap
<code>max_accel_power_cap</code>	Maximum nonzero accelerator power cap
<code>max_accel_power_cap_count</code>	Number of accelerators with the maximum nonzero power cap
<code>min_accel_power_cap</code>	Minimum nonzero accelerator power cap
<code>min_accel_power_cap_count</code>	Number of accelerators with the minimum nonzero power cap

RUR extended energy output

This example shows extended `energy` data as written to `/var/opt/cray/log/partition-current/messages-date` on the SMW:

```
2014-01-17T10:05:54.026557-06:00 c0-0c0s1n1 RUR 11674
p0-20140116t214834 [rur@34] uid: 12345, apid: 286342, jobid: 0,
cmdname: /bin/cat, plugin: energy {"energy_used": 5641,
"accel_energy_used": 1340, "nodes": 32, "nodes_power_capped": 3,
"min_power_cap": 155, "min_power_cap_count": 2, "max_power_cap": 355,
"max_power_cap_count": 1, "nodes_accel_power_capped": 3,
"min_accel_power_cap": 200, "min_accel_power_cap_count": 3,
"max_accel_power_cap": 200, "max_accel_power_cap_count": 3,
"nodes_throttled": 0, "nodes_with_changed_power_cap": 0}
```

The gpustat Data Plugin

The `gpustat` plugin collects the following utilization statistics for NVIDIA GPUs, if present. The data is written in JSON list format.

<code>maxmem</code>	Maximum memory used across all nodes
<code>summem</code>	Total memory used across all nodes
<code>gpusecs</code>	Time spent processing on GPUs

RUR gpustat output

This example shows `gpustat` data as written in `/var/opt/cray/log/partition-current/messages-date` on the SMW.

```
2013-07-09T15:50:42.761257-05:00 c0-0c0s2n2 RUR 11329
p2-20130709t145714 [RUR@34] uid: 12345, apid: 8410, jobid: 0,
```

```
cmdname: /tmp/dostuff plugin: gpustats ['maxmem', 108000, 'summem',
108000, 'gpusecs', 44]
```

The kncstats Data Plugin

The `kncstats` plugin collects the following process accounting data from Intel Xeon Phi (KNC) coprocessors, if present. The data is written in JSON list format.

<code>core</code>	Set to 1 if core dump occurred
<code>exitcode</code>	Lists all unique exit codes
<code>max_rss</code>	Maximum memory used
<code>rchar</code>	Characters read by process
<code>stime</code>	System time
<code>utime</code>	User time
<code>wchar</code>	Characters written by process

RUR kncstats output

This example shows `kncstats` data as written to `/var/opt/cray/log/partition-current/messages-date` on the SMW:

```
2014-02-25T18:49:12.101383-06:00 c0-0c0s1n1 RUR 11274
p0-20140225t135439 [RUR@34] uid: 12345, apid: 539224, jobid: 0,
cmdname: /bin/date, plugin: kncstats ['utime', 8000, 'stime', 20000,
'max_rss', 620, 'rchar', 2730, 'wchar', 119, 'exitcode:signal',
['0:0'], 'core', 0]
```

The memory Data Plugin

The `memory` plugin collects information from `/proc` and `/sys` that is useful when assessing the memory performance of an application or job. The data is written in JSON dictionary format. The type of data reported is determined by the value of the argument `arg` set in the `[memory]` section of the RUR configuration file.

IMPORTANT: The `memory` plugin does not provide consolidated information for all nodes within an application; instead it reports memory statistics for each node within the application. This can result in a large amount of RUR output data for systems of even modest size. When the `memory` plugin is enabled, it produces a significant amount of output.

If `arg` is not set (default), the plugin reports the following data:

<code>error</code>	If a Python exception occurs during the post or staging scripts, the following data is reported:
--------------------	--

traceback	Stack frame list
type	Python exception type
value	Python exception parameter
nid	NID on which exception occurred
cname	cname on which exception occurred
%_of_boot_mem	The % of boot memory for each order chunk in /proc/buddyinfo summed across all memory zones
Active (anon)	Total amount of memory in active use by the application
Active (file)	Total amount of memory in active use by cache and buffers
boot_freemem	Contents of /proc/boot_freemem
current_freemem	Contents of /proc/current_freemem
free	Number of hugepages that are not yet allocated
hugepages-sizekB	The hugepage size for the select entries from /sys/kernel/mm/hugepages/hugepages-*kB/*
Inactive (anon)	Total amount of memory that is candidate to be swapped out
Inactive (file)	Total amount of memory that is candidate to be dropped from cache
nr	Number of hugepages that exist at this point
resv	Number of hugepages committed for allocation, but no allocation has occurred
Slab	Total amount of memory used by the kernel
surplus	Number of hugepages above nr

RUR default memory output

This example shows the default `memory` data as written to `/var/opt/cray/log/partition-current/messages-date` on the SMW.

```
2014-03-21T11:37:24.480982-05:00 c0-0c0s0n2 RUR 23710
p0-20140321t091957 [RUR@34] uid: 12345, apid: 33079, jobid: 0,
cmdname: /bin/hostname, plugin: memory {"current_freemem": 21858372,
"meminfo": {"Active (anon)": 35952, "Slab": 105824, "Inactive (anon)":
1104}, "hugepages-2048kB": {"nr": 5120, "surplus": 5120},
"%_of_boot_mem": ["67.23", "67.23", "67.23", "67.22", "67.21",
"67.18", "67.11", "67.04", "66.94", "66.83", "66.77", "66.66",
"66.53", "66.38", "65.87", "65.07", "63.05", "61.43"], "nid": "8",
"cname": "c0-0c0s2n0", "boot_freemem": 32432628}
```

If `arg` is set to `extended_buddy`, the output relating to `/proc/buddyinfo` includes NUMA node granularity information in addition to the existing node granularity information. This information is useful when troubleshooting certain fragmentation related issues.

RUR extended memory output

This example shows extended `memory` data as written to `/var/opt/cray/log/partition-current/messages-date` on the SMW:

```

2014-03-21T11:37:24.480982-05:00 c0-0c0s0n2 RUR 23710
p0-20140321t091957 [RUR@34] uid: 12345, apid: 33079, jobid: 0,
cmdname: /bin/hostname, plugin: memory {"current_freemem": 21858372,
"meminfo": {"Active(anon)": 35952, "Slab": 105824, "Inactive(anon)":
1104}, "hugepages-2048kB": {"nr": 5120, "surplus": 5120},
"Node_0_zone_DMA": ["0.05", "0.05", "0.05", "0.05", "0.05", "0.05",
"0.05", "0.05", "0.05", "0.04", "0.04", "0.03", "0.00", "0.00",
"0.00", "0.00", "0.00", "0.00"], "%_of_boot_mem": ["67.23", "67.23",
"67.23", "67.22", "67.21", "67.18", "67.11", "67.04", "66.94",
"66.83", "66.77", "66.66", "66.53", "66.38", "65.87", "65.07",
"63.05", "61.43"], "nid": "8", "cname": "c0-0c0s2n0", "boot_freemem":
32432628, "Node_0_zone_DMA32": ["6.07", "6.07", "6.07", "6.07",
"6.07", "6.07", "6.07", "6.06", "6.05", "6.04", "6.01", "5.94",
"5.86", "5.76", "5.46", "4.85", "3.23", "3.23"], "Node_0_zone_Normal":
["61.11", "61.11", "61.11", "61.11", "61.09", "61.07", "60.99",
"60.93", "60.84", "60.75", "60.72", "60.70", "60.67", "60.62",
"60.42", "60.22", "59.81", "58.20"]}

```

The taskstats Data Plugin

ATTENTION: The default setting of the configuration argument `arg` will change from `json-list` to `json-dict` in release CLE6.0. This will result in changes to the content and format of the default output. `json-list` is deprecated and will be removed in a future release but will remain functional until that time.

The `taskstats` plugin collects process accounting data. The amount of data reported and the format in which it is written is determined by the value of the argument `arg` set in the `[taskstats]` section of the RUR configuration file.

If `arg` is not set (default), the plugin reports the following basic process accounting data similar to that provided by UNIX process accounting or `getrusage`. These values are sums across all nodes, except for the memory used, which is the maximum value across all nodes. The data is written in JSON list format.

<code>core</code>	Set to 1 if core dump occurred
<code>exitcode</code>	Lists all unique exit codes
<code>max_rss</code>	Maximum memory used
<code>rchar</code>	Characters read by process
<code>stime</code>	System time
<code>utime</code>	User time
<code>wchar</code>	Characters written by process

RUR default taskstats output

This example shows default `taskstats` output as written to `/var/opt/cray/log/partition-current/messages-date` on the SMW.

For a job that exits normally:

```

2013-11-02T11:09:49.457770-05:00 c0-0c1s1n2 RUR 2417
p0-20131101t153028 [RUR@34] uid: 12345, apid: 86989, jobid: 0,

```

```
cmdname: /lus/tmp/rur01.2338/./CPU01-2338 plugin: taskstats ['utime',
10000000, 'stime', 0, 'max_rss', 940, 'rchar', 107480, 'wchar', 90,
'exitcode:signal', ['0:0'], 'core', 0]
```

For a job that core dumps:

```
2013-11-02T11:12:45.020716-05:00 c0-0c1s1n2 RUR 3731
p0-20131101t153028 [RUR@34] uid: 12345, apid: 86996, jobid: 0,
cmdname: /lus/tmp/rur01.3657/./exit04-3657 plugin: taskstats ['utime',
4000, 'stime', 144000, 'max_rss', 7336, 'rchar', 252289, 'wchar', 741,
'exitcode:signal', ['0:9', '139:0', '0:11', '0:0'], 'core', 1]
```

If `arg` is set to `xpacct`, the plugin also provides the following extended process accounting data similar to that which was collected by the deprecated Cray System Accounting (CSA).

abortinfo If abnormal termination occurs, a list of `abort_info` fields is reported

apid Application ID as defined by application launcher

bkiowait Total delay time (ns) waiting for synchronous block I/O to complete

btime UNIX time when process started

comm String containing process name. May be different than the header, which is the process run by the launcher.

coremem Integral of RSS used by process in MB-usec

ecode Process exit code

etime Total elapsed time in microseconds

gid Group ID

jid Job ID - the PAGG job container used on the compute node

majfault Number of major page faults

minfault Number of minor page faults

nice POSIX `nice` value of process

nid String containing node ID

pgswpcnt Number of pages swapped; should be 0 on Cray compute nodes

pid Process ID

pjid Parent job ID - the PAGG job container on the MOM node

ppid Parent process ID

prid Job project ID

rcalls Number of read system calls

rchar Characters read by process

¹ The current memory usage is added to these counters (i.e., `coremem`, `vm`) every time. A tick is charged to a task's system time. Therefore, at the end we will have memory usage multiplied by system time and an average usage per system time unit can be calculated.

rss	RSS highwater mark
sched	Scheduling discipline used on node
uid	User ID
vm	Integral of virtual memory used by process in MB-usecs ²
wcalls	Number of write system calls
wchar	Characters written by process

RUR extended `taskstats` output

This example shows RUR extended `taskstats` output:

```
2013-10-18T10:29:38.285378-05:00 c0-0c0s1n1 RUR 24393
p1-20131018t081133 [RUR@34] uid: 12345, apid: 370583, jobid: 0,
cmdname: /bin/cat, plugin: taskstats ['btime', 1386061749, 'etime',
8000, 'utime', 0, 'stime', 4000, 'coremem', 442, 'max_rss', 564,
'max_vm', 564, 'pgswapcnt', 63, 'minfault', 15, 'majfault', 48,
'rchar', 2608, 'wchar', 686, 'rcalls', 19, 'wcalls', 7, 'bkiowait',
1000, 'exitcode:signal', [0], 'core', 0]
```

If `arg` is set to `xpacct`, `per-process`, the plugin reports extended accounting data for every compute node process rather than a summary of all processes for an application. `per-process` must be set in combination with `xpacct`.



CAUTION: If `per-process` is set and many processes are run on each node, the volume of data generated and stored on disk can become an issue.

RUR per-process `taskstats` output

This example shows RUR per-process `taskstats` output. This output was generated with the `json-dict` option set.

```
2013-12-03T13:25:34.446167-06:00 c0-0c2s0n2 RUR 7623
p3-20131202t090205 [RUR@34] uid: 12345, apid: 1560, jobid: 0,
cmdname: ./it.sh, plugin: taskstats {"uid": 12795, "wcalls": 37,
"pid": 2997, "vm": 16348, "jid": 395136991233, "bkiowait": 1201616,
"majfault": 1, "etime": 0, "btime": 1386098731, "gid": 0, "ppid":
2992, "utime": 0, "nice": 0, "sched": 0, "nid": "92", "prid": 0,
"comm": "mount", "stime": 4000, "wchar": 3465, "rss": 1028,
"minfault": 352, "coremem": 1109, "ecode": 0, "rcalls": 22, "pjid":
7045, "pgswapcnt": 0, "rchar": 12208}

2013-12-03T13:25:34.949138-06:00 c0-0c2s0n2 RUR 7623
p3-20131202t090205 [RUR@34] uid: 12345, apid: 1560, jobid: 0,
cmdname: ./it.sh, plugin: taskstats {"uid": 12795, "wcalls": 0, "pid":
2998, "vm": 20268, "jid": 395136991233, "bkiowait": 0, "majfault": 0,
"etime": 0, "btime": 1386098731, "gid": 0, "ppid": 2992, "utime": 0,
"nice": 0, "sched": 0, "nid": "92", "prid": 0, "apid": 1560, "comm":
```

² The current memory usage is added to these counters (i.e., `coremem`, `vm`) every time. A tick is charged to a task's system time. Therefore, at the end we will have memory usage multiplied by system time and an average usage per system time unit can be calculated.

```
"ls", "stime": 4000, "wchar": 0, "rss": 1040, "minfault": 360,
"coremem": 3140, "ecode": 0, "rcalls": 19, "pjid": 7045, "pgswapcnt":
0, "rchar": 10629}
```

If `arg` is set to `json-dict`, the data is written in JSON dictionary format.

If `arg` is set to `json-list`, the data is written in JSON list format (default).

The timestamp Data Plugin

The `timestamp` plugin collects the start and end times of an application or job.

RUR timestamp output

This example shows `timestamp` data, as written

in `/var/opt/cray/log/partition-current/messages-date` on the SMW, for an application that slept 20 seconds:

```
2013-08-30T14:32:07.593469-05:00 c0-0c0s5n2 RUR 12882
p3-20130830t074847 [RUR@34] uid: 12345, apid: 6640, jobid: 0,
cmdname: /bin/sleep plugin: timestamp APP_START 2013-08-30T14:31:46CDT
APP_STOP 2013-08-30T14:32:06CDT
```

The file Output Plugin

The `file` plugin allows RUR data to be stored to a flat text file on any file system to which the login node can write. This plugin is also intended as a very simple guide for anyone interested in writing an output plugin.

This example shows sample output from `file` to a location defined in the RUR configuration file:

```
uid: 1000, apid: 8410, jobid: 0, cmdname: /tmp/dostuff plugin:
taskstats ['utime', 32000, 'stime', 132000, 'max_rss', 1736, 'rchar',
44524, 'wchar', 289] uid: 1000, apid: 8410, jobid: 0, cmdname: /tmp/
dostuff plugin: energy ['energy_used', 24551] uid: 1000, apid: 8410,
jobid: 0, cmdname: /tmp/dostuff plugin: gpustats ['maxmem', 108000,
'summem', 108000]
```

The llm Output Plugin

The `llm` plugin aggregates log messages from various Cray nodes and places them on the SMW. `llm` has its own configuration options, but typically it will place RUR messages into the messages log file `/var/opt/cray/log/partition-current/messages-date` on the SMW. The messages shown in the previous sections are in LLM log format.

The user Output Plugin

The `user` plugin writes RUR output for a user's application to the user's home directory (default) or a user-defined location, only if the user has indicated that this behavior is desired (as described below).

The naming of the default output file(s), `rur.suffix`, is dependent on the value of the argument `arg`, which defines a report type and is set in the `user` section of the RUR configuration file. If `arg` is set to:

- apid** An output file is created for each application executed and *suffix* is the `apid`.
- jobid** An output file is created for each job submitted and *suffix* is the `jobid`
- single** All output is placed in a single file and no suffix is appended to the output file name.

User Options

Users have the option to opt-in or out for the `user` plugin, redirect plugin output to a specific file or directory, or override the default report type.

- By default, RUR data is not written to a user's directory. A user must either create the file `~/.rur/user_output_optin` to indicate that data should be written, or create a file that initiates one of the following two options.
 1. Users may redirect the output of RUR by specifying a redirect location in `~/.rur/user_output_redirect`. The contents of this file must be a single line that specifies the absolute or relative (from the user's home directory) path of the directory or file to which the RUR output data is to be written. If the redirect file either does not exist, points to a path that does not exist, or points to a path to which the user does not have write permission, then the output is written to the user's home directory.
 2. A user with an existing `~/.rur/user_output_redirect` file can temporarily stop RUR data from being written by setting the redirect path to `/dev/null`.
- Additionally, the user may override the default report type by specifying a valid report type in `~/.rur/user_output_report_type`. Valid report types are `apid`, `jobid`, or `single`, resulting in the user's RUR data being written to one file per application, one file per job, or a single file, respectively. If the file `~/.rur/user_output_report_type` is empty or contains an invalid type, then the default report type, as defined in the configuration file, is created.

The database Example Output Plugin

The `database` plugin is provided as a guide for sites wanting to output RUR data to a site-supplied database. Sites will need to configure their own systems, provide an external database, create their own tables, and modify `database_output.py` to collect the desired data.

MySQL is the database supported by the example plugin. The following arguments are defined for connecting to a database:

- `DB_NAME='rur'`
- `DB_USER='rur_user'`

- `DB_PASS='rur_pass'`
- `DB_HOST='rur_host'`

The database plugin collects the values: `energy_used`, `apid`, `jobid`, and `uid`, and saves this data to a table, `energy`. It does this by performing the following:

- Digests RUR data into a dictionary and saves it to class `DbData`
- Creates rules for saving data collected in `DbData` to particular tables
- Uses the rules to scan the `DbData` dictionary and INSERT that data into a database

Cray recommends that the database is not hosted on SDB or login nodes. It should also be noted that, depending on job load, interacting with an external database may cause system latency.

Create Custom RUR Data Plugins

A data plugin is comprised of a *staging component* and a *post processing component*. The data plugin staging component is called by `rur-stage.py` on the compute node prior to the application/job running and again after the application/job has completed. The staging component may reset counters before application/job execution and collect them after application/job completion, or it may collect initial and final values prior to and after application/job execution, respectively, and then calculate the delta values. Python functions have been defined to simplify writing plugins, although it is not necessary for the plugin to be written in Python. The interface for the data plugin staging component is through command line arguments.

Data Plugin Staging Component

All data plugin staging components must support the following arguments:

<code>--apid=<i>apid</i></code>	Defines the application ID of the running application.
<code>--timeout=<i>time</i></code>	Defines a timeout period in seconds during which the plugin must finish running. Set to 0 for unlimited; default is unlimited.
<code>--pre</code>	Indicates the plugin is being called prior to the application/job.
<code>--post</code>	Indicates the plugin is being called after the application/job.
<code>--outputfile=<i>output_file</i></code>	Defines where the output data is written. Each plugin should define a default output file in <code>/var/spool/RUR/</code> if this argument is not provided.
<code>--arg=<i>arg</i></code>	A plugin-specific argument, set in the RUR config file. RUR treats this as an opaque string.

The output of an RUR data plugin staging component is a temporary file located in `/var/spool/RUR` on the compute node. The file name must include both the name of the plugin, as defined in the RUR config file, and `.apid`. The RUR gather phase will automatically gather the staged files from all compute nodes after the application/job has completed and place it in `gather_dir` as defined in the configuration file.

Data plugin staging component

```
#
# Copyright (c) 2013 Cray Inc. All rights reserved.
#
```

```

# Sample data plugin staging component
#
#!/usr/bin/env python
import sys, os, getopt
from rur_plugins import rur_plugin_args
def main():
    apid, inputfile, outputfile, timeout, pre, post, \
        parg = rur_plugin_args(sys.argv[1:])
    if outputfile is "":
        outputfile = "/var/spool/RUR/pluginname."+str(apid)
    if (pre==1):
        zero_counters()
    else:
        write_postapp_stateto(outputfile)

if __name__ == "__main__":
    main()

```

Data Plugin Post Processing Component

A data plugin also requires a post processing component that processes the data staged by the staging component and collected during the RUR gather phase. The post processing component is called by `rur-post.py`. The input file contains records, one node per line, of all of the statistics created by the staging component. The output of the post processing component is a file containing the summary of data from all compute nodes.

All data plugin post processing components must support the following arguments:

- `--apid=apid` Defines the application ID of the running application.
- `--timeout=time` Defines a timeout period in seconds during which the plugin must finish running. Set to 0 for unlimited; default is unlimited.
- `--inputfile=input_file` Specifies the file from which the plugin gets its input data.
- `--outputfile=output_file` Specifies the file to which the plugin writes its output data.

Data plugin post processing component

```

#
# Copyright (c) 2013 Cray Inc. All rights reserved.
#
# Sample data plugin post processing component
#
#!/usr/bin/env python
import sys, os
from rur_plugins import rur_args

def main():
    apid, inputfile, outputfile, timeout = rur_args(sys.argv[1:])
    if outputfile is "":
        outputfile = inputfile + ".out"

    pc = PostCompute()
    pc.process_file(inputfile)
    formatted = pc.present_entries([('plugin_foo_data', 'sum')])
    fout=open(outputfile, 'w+')
    fout.write("energy %s" % formatted)

```

```
if __name__ == "__main__":
    main()
```

Create Custom RUR Output Plugins

Output plugins allow RUR data to be outputted to an arbitrary backing store. This can be a storage device or another piece of software that then consumes the RUR data. The output plugin is passed a number of command line arguments that describe the application/job run and provide a list of input working files (the output of data plugin post processing components). The plugin takes the data in the working files and exports it to the destination specified in the RUR configuration file for the specific output plugin.

Data passed to custom output plugins can be optionally configured to be JSON-formatted by adding the `use_json` argument to the `[global]` section of the configuration file and setting it to `True`, `yes`, `1`, or `enable`.

TIP: If there is an error from an output plugin, the error message appears in the ALPS log `/var/opt/cray/alps/log/apsys` on the service node rather than the LLM logs on the SMW.

Output Plugin

```
#
# Copyright (c) 2013 Cray Inc. All rights reserved.
#
# Sample output plugin
#
#!/usr/bin/env python
import sys, os
from rur_plugins import rur_output_args

def main():
    apid, jobid, uid, cmdname, inputfilelist, timeout, \
    parg = rur_output_args(sys.argv[1:])

    outfile = open(parg, "a")
    for inputfile in inputfilelist:
        infile = open(inputfile, "r")
        lines = infile.readlines()
        for line in lines:
            outfile.write(line)
        infile.close()
    outfile.close()
```

Implement a Site-Written RUR Plugin

For a site written plugin to run, it must be added to the RUR configuration file and enabled. Follow the procedure to define and configure a new plugin.

1. Ensure that the site written plugin is located on a file system that is readable by compute nodes, owned by `root`, and not writeable by non-`root` users.
2. Add a new plugin definition section to the RUR configuration file:
 - a. If adding a data plugin, the definition section must include: the plugin name, a `stage` definition (the complete path to the plugin's data staging script), and a `post` definition (the complete path to the plugin's post processing script).
For example, to define the site written data plugin `siteplug`, the entry within the RUR configuration would be similar to the following:

```
# The siteplug Data Plugin collects data that is
#   of particular interest to this site.
# Stage - The staging component run by rur_stage on the
#   compute node
# Post - The post-processing component run by rur_post on
#   the login/mom node
[siteplug]
stage: /opt/cray/rur/default/bin/siteplug_stage.py
post: /opt/cray/rur/default/bin/siteplug_post.py
```

- b. If adding an output plugin, the definition section must include: the plugin name, an `output` definition (the complete path to the output plugin script or binary), and an optional argument.
For example, to define the site written output plugin `siteout`, the entry within the RUR configuration would be similar to the following:

```
# The siteout output plugin.
# Write RUR output to a text file on the site's huge
# archive file system.
[siteout]
output:/opt/cray/rur/site/bin/site_output.py
arg:hsmuser@hsmbackup.site.com:/hsmuser/rurbackup
```

3. Add the new plugin to either the data plugin or output plugin configuration section, labeled `[plugins]` or `[outputplugins]`, respectively. Indicate `true` to enable or `false` to disable plugin execution. This example shows the new `siteplug` data plugin enabled and the `siteout` output plugin disabled:

```
# Data Plugins section Configuration
# Define the supported Data plugins and enable/disable
# them. Plugins defined as "Plugin: False" will not run,
# but will be parsed for correct config file syntax.
[plugins]
gpustat: true
taskstats: true
siteplug: true

# Output Plugins section Configuration
# Define which output plugins are supported, and enable/
# disable them. Plugins defined as "Plugin: False" will
# not run, but will be parsed for correct config file
# syntax.
[outputplugins]
llm: true
file: false
siteout: false
```

Additional Plugin Examples

This is a set of RUR plugins that report information about the number of available huge pages on each node. The huge page counts are reported in `/proc/buddyinfo`. There are two versions of the staging component: one that reports what is available and the second that reports changes during the application run.

Huge pages data plugin staging component (version A)

```
#
# Copyright (c) 2013 Cray Inc. All rights reserved.
#
# This is an RUR plugin that reports information about the number of
# available
# huge pages on each node. This is reported in /proc/buddyinfo.
#
# Each node reports its nid and the number of available pages of
# each size.
#
#!/usr/bin/env python
import sys, os, getopt
from rur_plugins import rur_plugin_args
def main():
    apid, inputfile, outputfile, timeout, pre, post, parg
=rur_plugin_args(sys.argv[1:])
    if outputfile == 0:
        outputfile = "/var/spool/RUR/buddyinfo."+str(apid)
    if (pre==1):
        zero_counters()
    else:
        nidf = open("/proc/cray_xt/nid", "r")
        n = nidf.readlines()
        nid = int(n[0])
        inf = open("/proc/buddyinfo", "r")
        b = inf.readlines()
        sizes = dict( [( '2M' , 0 ), ( '4M' , 0 ), ( '8M' , 0 ), ( '16M' ,
0 ), ( '32M' , 0 ), ( '64M' , 0 )])

        for line in b:
            l = line.split()
            sizes['2M'] += int(l[13])
            sizes['4M'] += int(l[14])
            sizes['8M'] += int(l[15])
            sizes['16M'] += int(l[16])
            sizes['32M'] += int(l[17])
            sizes['64M'] += int(l[18])

            o = open(outputfile, "w")
            o.write("{6} {0} {1} {2} {3} {4}
{5}".format(sizes['2M'], sizes['4M'], \
            sizes['8M'], sizes['16M'], sizes['32M'], sizes['64M'], nid))
            o.close()

if __name__ == "__main__":
    main()
```

Huge pages data plugin staging component (version B)

```

#
# Copyright (c) 2013 Cray Inc. All rights reserved.
#
# This is an RUR plugin that reports information about the number of
available
# huge pages on each node. This is reported in /proc/buddyinfo.
#
# This plugin records the number of available pages before the job
is launched.
# At job completion time it reports the change
#
#!/usr/bin/env python
import sys, os, getopt
from rur_plugins import rur_plugin_args
def main():
    apid, inputfile, outputfile, timeout, pre, post, parg
=rur_plugin_args(sys.argv[1:])
    if outputfile == 0:
        outputfile = "/var/spool/RUR/buddyinfo."+str(apid)
    if (pre==1):
        inf = open("/proc/buddyinfo", "r")
        b = inf.readlines()
        sizes = dict( [( '2M' , 0 ), ('4M', 0), ('8M', 0),
('16M', 0), ('32M', 0), ('64M', 0)])
        for line in b:
            l = line.split()
            sizes['2M'] += int(l[13])
            sizes['4M'] += int(l[14])
            sizes['8M'] += int(l[15])
            sizes['16M'] += int(l[16])
            sizes['32M'] += int(l[17])
            sizes['64M'] += int(l[18])

            o = open("/tmp/buddyinfo_save", "w")
            o.write("{0} {1} {2} {3} {4}
{5}".format(sizes['2M'], sizes['4M'], \
            sizes['8M'], sizes['16M'], sizes['32M'],
sizes['64M']))
            o.close()
        else:
            nidf = open("/proc/cray_xt/nid", "r")
            n = nidf.readlines()
            nid = int(n[0])
            inf = open("/proc/buddyinfo", "r")
            b = inf.readlines()
            sizes = dict( [( '2M' , 0 ), ('4M', 0), ('8M', 0),
('16M', 0), ('32M', 0), ('64M', 0)])

            for line in b:
                l = line.split()
                sizes['2M'] += int(l[13])
                sizes['4M'] += int(l[14])
                sizes['8M'] += int(l[15])
                sizes['16M'] += int(l[16])
                sizes['32M'] += int(l[17])
                sizes['64M'] += int(l[18])

            obf = open("/tmp/buddyinfo_save", "r")

```

```

ob = obf.readlines()
n=0

obd0 = ob[0]
obd = obd0.split()

diff = [
    (int(obd[0]) - sizes['2M']),
    (int(obd[1]) - sizes['4M']),
    (int(obd[2]) - sizes['8M']),
    (int(obd[3]) - sizes['16M']),
    (int(obd[4]) - sizes['32M']),
    (int(obd[5]) - sizes['64M'])
]

o = open(outputfile, "w")
# uncomment the following line to get the actual sizes
#o.write("sizes {6} {0} {1} {2} {3} {4}
{5}\n".format(sizes['2M'], sizes['4M'], \
    sizes['8M'], sizes['16M'], sizes['32M'],
sizes['64M'], nid))
o.write("diff {6} {0} {1} {2} {3} {4} {5}".format(diff[0],
diff[1], diff[2], \
    diff[3], diff[4], diff[5], nid))
o.close()
os.unlink("/tmp/buddyinfo_save")

if __name__ == "__main__":
    main()

```

Huge pages data plugin post processing component

```

#
# Copyright (c) 2013 Cray Inc. All rights reserved.
#
# This is a RUR postprocessing plugging for the buddyinfo data
# collection. It copies the input files to output, adding a
# "buddyinfo" label.
#
#!/usr/bin/env python
import sys, os
from rur_plugins import rur_args

def main():
    apid, inputfile, outputfile, timeout = rur_args(sys.argv[1:])
    if outputfile == 0:
        outputfile = inputfile + ".out"

    fin=open(inputfile, "r")
    l = fin.readlines()

    fout=open(outputfile, 'w+')
    for line in l:
        fout.write("buddyinfo {0}".format(line))

if __name__ == "__main__":
    main()

```

Application Completion Reporting (ACR) to RUR Migration Tips

Cray supplied RUR data plugins collect the same data found in Mazama's Application Completion Reporting (ACR) feature (deprecated), but RUR does not include a reporting utility like `mzreport`. When using RUR's `llm` output plugin, the type of data reported by `mzreport` can be extracted from the output files as demonstrated in the following sections.

ACR Job Reporting

The information provided by `mzreport -j` and `mzreport --job` can easily be obtained in the RUR environment from the log files `/var/opt/cray/log/partition-current/messages-date` by invoking the following command:

```
smw:~ # grep -e "RUR" messages-* |grep -e "jobid: jobid"
```

ACR Timespan Reporting

In ACR, `mzreport -t` and `mzreport -T` control the span of time over which job completions are reported. The following example is a simple Python script, `timesearch.py`, that provides this functionality.

```
#cat timesearch.py
#!/usr/bin/env python
for rurline in [line for line in open(sys.argv[1], 'r') if 'RUR' in line]:
    if (rurline.split(' ')[1] > sys.argv[2]) and (rurline.split(' ')[1] <
        sys.argv[3]):
        print rurline
```

The script is called with the log file of interest and the desired start/stop time stamps, where `start_time` and `end_time` are formatted as "`yyyy-mm-ddThh:mm:ss`", as follows:

```
smw:~ # python ./timesearch.py messages-date "start_time" "end_time"
```

ACR Exit Code Reporting

The `get_exit.py` Python script listed here provides a list of the user IDs with the most non-zero exit codes.

```
# cat get_exit.py
#!/usr/bin/env python
import os,sys,re

statre = re.compile("(\\w*):(\\w*)',\\s*\\[(\\w*):(\\w*)'(, )?+\\]")
statsre = re.compile("(\\w*):(\\w*)")
uidre = re.compile("uid:\\s*(\\w*)")
cnt = {}

for rurline in [line for line in open(sys.argv[1], 'r') if 'RUR' in line]:
    if 'taskstats' in rurline:
        sus = statre.search(rurline)
        status = sus.group()
        stats = statsre.findall(status)
        for stat in stats[1:]:
            if stat[0] != '0':
                uid = int(uidre.findall(rurline)[0])
                if cnt.get(str(uid)):
```

```

        cnt[str(uid)] += 1
    else:
        cnt[str(uid)] = 1

x = sorted(cnt, key = cnt.get, reverse=True)
print "uids with the most non-zero exit codes %s" % x[:sys.argv[2]]

```

The script is called with the log file of interest and the number of user IDs on which to report, as follows:

```
smw:~ # python ./get_exit.py messages-date num
```

Application Resource Utilization (ARU) to RUR Migration Tips

Sites that use ARU (deprecated) will have an easy transition to RUR as all of the data provided in ARU is available in RUR, but in a slightly different format.

This example shows that the following ARU output is available by enabling the `taskstats` plugin's default behavior:

ARU output:

```

2012-11-26T08:52:37.802113-06:00 c0-0c0s0n2 aphys 19864
p0-20121126t060549 -
apid=6240364, Finishing, user=8855, batch_id=114.sdb, exit_code=0,
exitcode_array=0,
exitsignal_array=0, utime=0 stime=0 maxrss=3168 inblocks=0 outblocks=0
cpus=8
start=Mon Nov 26 08:52:37 2012 stop=Wed Dec 31 18:00:00 1969
cmd=growfiles

```

RUR `taskstats` default output:

```

2013-11-02T11:09:49.457770-05:00 c0-0c1s1n2 RUR 2417
p0-20131101t153028 [RUR@34]
uid: 10973, apid: 86989, jobid: 0, cmdname: /lus/esfs/overby/
rur01.2338/./CPU01-2338
plugin: taskstats ['utime', 10000000, 'stime', 0, 'max_rss', 940,
'rchar', 107480,
'wchar', 90, 'exitcode:signal', ['0:0'], 'core', 0]

```

This example shows that the following ARU output is available by enabling the RUR `timestamp` plugin.

ARU output:

```

2012-11-26T08:53:15.618239-06:00 c0-0c0s0n2 aphys 20604
p0-20121126t060549 -
apid=6240378, Finishing, user=8855, batch_id=121.sdb, exit_code=0,
exitcode_array=0,
exitsignal_array=0, utime=0 stime=0 maxrss=3152 inblocks=0 outblocks=0
cpus=1
start=Mon Nov 26 08:52:51 2012 stop=Wed Dec 31 18:00:00 1969
cmd=close2_01

```

RUR timestamp plugin output:

```
2013-08-30T14:32:07.593469-05:00 c0-0c0s5n2 RUR 12882
p3-20130830t074847 [RUR@34] uid: 0,
apid: 6640, jobid: 0, cmdname: /bin/sleep plugin: timestamp APP_START
2013-08-30T14:31:46CDT APP_STOP 2013-08-30T14:32:06CDT
```

CSA to RUR Migration Tips

The Cray supplied RUR data plugin `taskstats`, when enabled and configured for extended accounting data, collects all of the data in the CSA process accounting record with the exception of `ac_sbu`, the system billing units.

RUR extended `taskstats` output

This example shows RUR extended `taskstats` output:

```
2013-10-18T10:29:38.285378-05:00 c0-0c0s1n1 RUR 24393
p1-20131018t081133 [RUR@34] uid: 12345, apid: 370583, jobid: 0,
cmdname: /bin/cat, plugin: taskstats ['btime', 1386061749, 'etime',
8000, 'utime', 0, 'stime', 4000, 'coremem', 442, 'max_rss', 564,
'max_vm', 564, 'pgswpcnt', 63, 'minfault', 15, 'majfault', 48,
'rchar', 2608, 'wchar', 686, 'rcalls', 19, 'wcalls', 7, 'bkiowait',
1000, 'exitcode:signal', [0], 'core', 0]
```

RUR does not include the report generation capabilities provided by CSA, however, the type of data reported by CSA can be extracted from the messages files on the SMW. The following is a short Python script for searching through these files. It allows filtering for group ID (`-g`), job ID (`-j`), user ID (`-u`), and system time exceeding a certain value (`-s`); similar to the `csacom` filters `-g`, `-j`, `-u`, `-O`, respectively.

```
#!/usr/bin/env python
# Usage: filter-messages [-g gid] [-j jid] [-u uid] [-s stime] -f messages-date
import os,sys,re,getopt,collections

def getcmdlineargs(args):
    arglist = collections.defaultdict(lambda: 0, {})
    options, remainder = getopt.getopt(args,
        'g:j:u:s:f:',
        ['gid=', 'jid=', 'uid=', 'Stimeexceeds=', 'filename='])

    for opt,arg in options:
        if opt in ('-g', '--gid'):
            arglist['gid'] = arg
        if opt in ('-j', '--jid'):
            arglist['jid'] = arg
        if opt in ('-u', '--uid'):
            arglist['uid'] = arg
        if opt in ('-s', '--Stimeexceeds'):
            arglist['stimeexceeds'] = arg
        if opt in ('-f', '--filename'):
            arglist['filename'] = arg
    return arglist
```

```
def reeqgt(tag, restr, rurline, eq):
    retre = re.compile("'" + str(restr) + "'," + "\s*(\w*)")
    field = retre.findall(rurline)
    if field == []:
        return False
    if eq and tag == field[0]:
        return True
    elif (not eq) and tag <= field[0]:
        return True
    return False

arglist = getcmdlineargs(sys.argv[1:])
if not arglist['filename']:
    exit(1)
for rurline in [line for line in open(arglist['filename'], 'r') if 'RUR' in
line]:
    if 'taskstats' in rurline:
        if arglist['jid'] and not (reeqgt(arglist['jid'], 'jid', rurline, 1)):
            continue
        if arglist['uid'] and not (reeqgt(arglist['uid'], 'uid', rurline, 1)):
            continue
        if arglist['gid'] and not (reeqgt(arglist['gid'], 'gid', rurline, 1)):
            continue
        if arglist['stimeexceeds'] and not (reeqgt(arglist['stimeexceeds'],
'stime',
rurline, 0)):
            continue

    print "%s" % rurline,
```

Dynamic Shared Objects and Cluster Compatibility Mode

Users can link and load dynamic shared objects in their applications by using the compute node root runtime environment (CNRTE) in the Cray Linux Environment (CLE). CLE includes software that enables compiling with dynamic libraries, using an alternate to the `initramfs` file system on the CNL compute nodes, called the compute node root. The compute node root is essentially the read-only DVS-projected shared root file system. This supports the ability to run a limited set of dynamically linked binaries on compute nodes.

The main benefit of this feature is expanded use of programs and libraries that require shared libraries on Linux cluster systems. If an independent software vendor (ISV) program ships with compiled binaries and dynamic libraries, this feature will be advantageous. Users are able to effectively reduce memory and executable footprint when shared objects, called multiple times, use the same segment of memory address space. Users can create applications that no longer need recompiling when libraries change.

Administrators enable this option at install time by modifying parameters in the `CLEinstall.conf` file.

For additional information, see [Install and Configuring Cray Linux Environment \(CLE\) Software \(S-2444\)](#) and [CLE User Application Placement Guide \(S-2496\)](#).

CNRTE is the framework used to allow compute node access to dynamic shared objects and libraries. Configuring and installing the compute node root runtime environment involves setting up the shared root as a DVS-projected file system. This process entails configuring DVS server nodes and updating the compute node boot images to enable them as clients.

Configure the Compute Node Root Runtime Environment (CNRTE)

1. Determine which service or compute nodes will be the compute node root servers.

There are essentially two classes of nodes in a Cray system: service or compute. Service nodes have connectivity to external file systems and networks, access to the shared root of the boot node, and a full set of Linux services. Compute nodes have reduced services and a lightweight kernel to allow a maximized utilization of computational resources. Some services do not require external connectivity but are still desirable. There is also a practical limit to the number of available service nodes for each site. CLE allows the service node image to run on a node otherwise considered a compute node to act as an internal DVS server of the Cray system shared root.

NOTE: Any compute nodes chosen here will no longer be a part of the available compute node pool. An allocation mode of `other` will be assigned to these compute nodes in the service database (SDB). These nodes will no longer belong to the group of batch and interactive nodes in the SDB and they will be unavailable to ALPS.



CAUTION: Do not place DVS servers on the same node as a Lustre (Object Storage, Metadata or Management) server. Doing so can cause load oversubscription on the node and reduce performance.

If the `/etc` files are specialized with a `cnos` class, the `cnos` class `/etc` files will be mounted on top of the projected shared root content on the compute nodes. This class specialization allows the compute nodes to have access to a different set of `/etc` files that exist on the DVS servers. Otherwise, the compute nodes will use the set of `/etc` files that are specific to their DVS server and that are contained in the shared root of the DVS server projects.

- When editing the `CLEinstall.conf` file and running the `CLEinstall` program, modify the parameters specific to shared object support according to the site-specific configuration. The `CLEinstall` program will automatically configure the system for the compute node root runtime environment.

DSL=yes	This variable enables dynamic shared objects and libraries for CLE. The default is <code>no</code> . Setting this option to <code>yes</code> will automatically enable DVS.
DSL_nodes=17 20	The decimal NIDs of the nodes that will act as compute node root servers. These nodes can be a combination of service or compute nodes. Each NID is separated by a space.
DSL_mountpoint=/dsl	This path is the DVS mount point on the compute nodes; it is the projection of the shared root file system.
DSL_attrcache_timeout=14400	This value is the attribute cache time out for compute node root servers. The value represents the number of seconds before DVS attributes are considered invalid and they are retrieved from the server again.

- Follow the appropriate procedures in *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)* to complete the installation.

The `/etc/opt/cray/cnrte/roots.conf` file contains site-specific values for custom root file systems. To specify a different pathname for `roots.conf` edit the configuration file `/etc/sysconfig/xt` and change the value for the variable, `CRAY_ROOTFS_CONF`. In the `roots.conf` file, the system default compute node root used is specified by the symbolic name `DEFAULT`. If no default value is specified, `/` will be assumed. In the following example segment of `roots.conf`, the default case uses `/dsl` as the reference root file system:

```
DEFAULT=/dsl
INITRAMFS=/
DSL=/dsl
```

Users can override the system default compute node root value by setting the `CRAY_ROOTFS` environment variable to a value from the `roots.conf` file. This changes the compute node root used for launching jobs. For example, to override the use of `/dsl` set `CRAY_ROOTFS` to `INITRAMFS`.

An administrator can modify the contents of this file to restrict user access. For example, if the administrator only wants to allow applications to launch using the compute node root, the `roots.conf` file would read like the following:

```
% cat /etc/opt/cray/cnrte/roots.conf
DEFAULT=/dsl
```

Cluster Compatibility Mode

A Cray system is not a cluster but a massively parallel processing (MPP) computer. An MPP is one computer with many networked processors used for distributed computation, and, in the case of Cray system architectures, a high-speed communications interface that facilitates optimal bandwidth and memory operations between those processors. When operating as an MPP machine, the Cray compute node kernel (Cray CNL) typically does not have a full set of the Linux services available that are used in cluster ISV applications.

Cluster Compatibility Mode (CCM) is a software solution that provides the services needed to run most cluster-based independent software vendor (ISV) applications out-of-the-box with some configuration adjustments. It is built on top of the compute node root runtime environment (CNRTE), the infrastructure that provides dynamic library support in Cray systems.

CCM may be coupled to the workload management system. It enables users to execute cluster applications together with workload-managed jobs that are running in a traditional MPP batch or interactive queue. Essentially, CCM uses the batch system to logically designate part of the Cray system as an emulated cluster for the duration of the job.

Figure 5. Cray System Job Distribution Cross-section

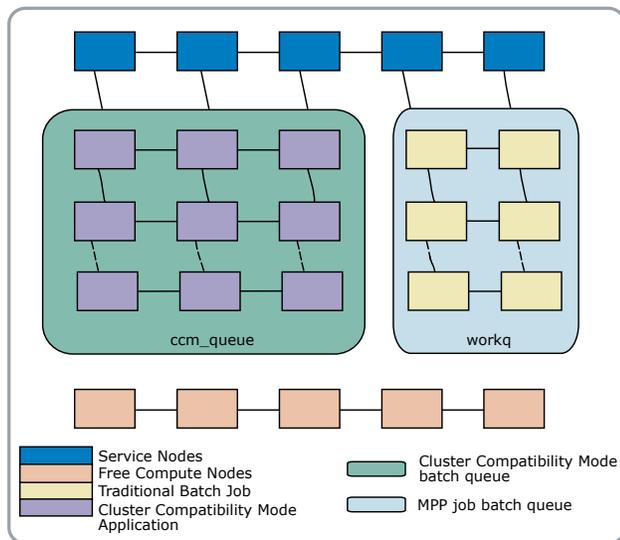
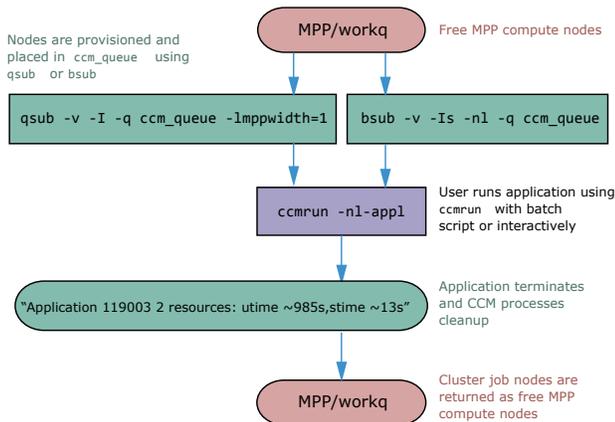


Figure 6. CCM Job Flow Diagram



CCM Preconditions and Configuration Options

Preconditions

- Dynamic library support is installed.
- (Optional) RSIP must be installed if applications need access to a license server; see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*.
- PBS 10.2RC2 (Emerald), LSF 8.0, or Torque-2.4.1b1-snap.200908271407 or later versions are installed.

Configuration Options

The following are exclusively post-install options included in `/etc/opt/cray/ccm/ccm.conf`:

CCM_DEBUG=no Setting this option to `yes` enables debug logging for CCM. These logs will be stored on the PBS MOM node in `/var/log/crayccm`. Cray recommends the site setting this option to `yes`.

CCM_RESOURCES="ccm ccm2" This option indicates that the administrator has configured a custom application resource that can be allocated and used for a job. Users requesting a job will consume one of the pool of available resources listed here. The job submission is checked against the list provided when making a determination whether the job is a CCM targeted job.

NOTE: Only one of `CCM_RESOURCES` or `CCM_QUEUES` is required.

To configure `yp`, `/etc/defaultdomain` and `/etc/yp.conf` must be properly configured on the compute node specialized view. Cray recommends using the `cnos` class within `xtopview` to set up this specialized view.

Use DVS to Mount Home Directories on the Compute Nodes for CCM

For each DVS server node configured, mount the path to the user home directories. Typically, these will be provided from a location external to the Cray system.

1. Specialize and add a line to the `/etc/fstab` file on the DVS server by using `xtopview` in the node view. For example, if the DVS server is `c0-0c0s2n3` (node 27 on a Cray XE system), type the following:

```
boot:~ # xtopview -m "mounting home dirs" -n 27
node/27:/ # xtspec -n 27 /etc/fstab
node/27:/ # vi /etc/fstab
nfs_home_server:/home          /home          nfs          tcp,rw  0 0
node/27:/ # exit
```

2. Log into each DVS server and mount the file system:

```
boot:~ # ssh nid00027
nid00027:~ # mount /home
nid00027:~ # exit
```

3. To allow the compute nodes to mount their DVS partitions, add an entry in the `/etc/fstab` file in the compute image for each DVS file system. For example:

```
smw:~ # vi /opt/xt-images/templates/default/etc/fstab
/home /home dvs path=/home,nodename=c0-0c0s2n3
```

4. For each DVS mount in the `/etc/fstab` file, create a mount point in the compute image.

```
smw:~ # mkdir -p /opt/xt-images/templates/default/home
```

5. Update the boot image to include these changes; follow the steps in [Prepare Compute and Service Node Boot Images](#) on page 37.

NOTE: This step can be deferred and the boot image can be updated once before booting the system.

Modify CCM and Platform-MPI System Configurations

Follow this procedure to enable additional features such as debugging and Linux NIS (Network Information Service) support.

1. Edit the CCM configuration file by using `xtopview` in the default view.

```
boot:~ # xtopview -m "configuring ccm.conf"
default:// # vi /etc/opt/cray/ccm/ccm.conf
```

To configure additional CCM debugging, set `CCM_DEBUG=yes`.

To enable NIS support, set `CCM_ENABLENIS=yes`.

2. If the site configuration is such that the paths for the `qstat` command are not at a standard location, change the values in the configuration file for `CRAY_QSTAT_PATH` and `CRAY_BATCH_VAR` accordingly for the site configuration.
3. Save and close `ccm.conf`.
4. Exit `xtopview`.

```
default:// # exit
boot:~ #
```

IMPORTANT: If applications will use Platform-MPI (previously known as *HP-MPI*), Cray recommends that users populate their `~/ .hmpi.conf` (or `~/ .pmi.conf`) file with these values.

```
MPI_REMSH=ssh
MPIRUN_OPTIONS="-cpu_bind=MAP CPU:
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,\
20,21,22,23,24,25,26,27,28,29,30,31"
```

Set up Files for the cnos Class

The `cnos` compute nodes that have access to the shared root through CNRTE will have a specialized class of its own `/etc` files. Login files and all `/etc` files should be migrated to the `cnos` class in order for CCM to work.

1. Use `xtopview` to access the `cnos` class specialized files.
If the SDB has not been started, use the `-x /etc/opt/cray/sdb/node_classes` option to specify node/class relationships.

```
boot:~# xtopview -m "CCM cnos setup" -c cnos
```

2. To add a file or modify a file, edit the file and then specialize it for the `cnos` class.

```
class/cnos:~# vi /etc/file
class/cnos:~# xtspec -c cnos /etc/file
```

Repeat the above steps for each new file to add or modify for the compute nodes.

3. Exit `xtopview`.

```
class/cnos:~# exit
```

`xtopview` prompts the administrator to type `c` and enter a brief comment describing the changes made. Type **Ctrl-d** or a period on a line by itself to complete a comment. Do this each time `xtopview` is exited to log a record of revisions into a version control system.

Link the CCM Prologue/Epilogue Scripts for use with PBS and Moab TORQUE on Login Nodes

Prerequisites

This procedure requires that a workload management system such as PBS or Moab TORQUE is already installed.

Add a line to reference to append the CCM `prologue` and `epilogue` scripts to the end of the existing batch `prologue` and `epilogue`. The PBS batch `prologue` is configured on all PBS MOM nodes in `/var/spool/PBS/mom_priv/prologue`. The Moab TORQUE batch `prologue` is configured on all TORQUE MOM nodes in `/var/spool/torque/mom_priv/prologue`.

This procedure assumes that `/bin/bash` as the current shell, but this can be modified appropriately for others.

1. Add the following lines to prologue:

```
#!/bin/bash
ccm_dir=/opt/cray/ccm/default/etc

if [ -x $ccm_dir/cray-ccm-prologue ] ; then
    $ccm_dir/cray-ccm-prologue $1 $2 $3
fi
```

2. Add the following lines to epilogue:

```
#!/bin/bash
ccm_dir=/opt/cray/ccm/default/etc

if [ -f $ccm_dir/cray-ccm-epilogue ] ; then
    $ccm_dir/cray-ccm-epilogue $1 $2 $3 $4 $5 $6 $7 $8 $9
fi
```

3. Set the executable bit for prologue and epilogue if not set:

```
system :/var/spool/PBS/mom_priv # chmod a+x prologue epilogue
```

4. Change the default batch time-out value. Cray recommends changing this to 120 seconds. This allows the system enough time to startup and shutdown all infrastructure on the nodes associated with the CCM job. To change the batch time out, append the following line to `/var/spool/PBS/mom_priv/config` or `/var/spool/torque/mom_priv/config`:

```
$prologalarm 120
```

Create a General CCM Queue and Queues for Separate ISV Applications

1. Set up a general CCM queue by issuing the following `qmgr` commands on the PBS server node:

```
# module load pbs
# qmgr
Qmgr: create queue ccm_queue
Qmgr: set queue ccm_queue queue_type = Execution
Qmgr: set queue ccm_queue resources_max.mpparch = XT
Qmgr: set queue ccm_queue resources_min.mpparch = XT
Qmgr: set queue ccm_queue resources_min.mppwidth = 1
Qmgr: set queue ccm_queue resources_default.mpparch = XT
Qmgr: set queue ccm_queue resources_default.mppwidth = 1
Qmgr: set queue ccm_queue enabled = True
Qmgr: set queue ccm_queue started = True
Qmgr: exit
```

For Moab TORQUE, add this command while creating the queue:

```
set server query_other_jobs = True
```

2. Repeat for additional application-specific queues, if desired.

Configure Platform LSF for use with CCM

Prerequisites

This procedure requires that Platform LSF workload management system is installed.

1. Determine the path to the directory that contains the files `lsb.queues` and `lsb.params`. This path is `${LSF_TOP}/conf/lsbatch/${LSF_CLUSTER_NAME}/configdir`, where `LSF_TOP` and `LSF_CLUSTER_NAME` are themselves paths that were defined at install time.

For example, if

```
LSF_TOP=/opt/xt-lsfhpcand
```

and

```
LSF_CLUSTER_NAME=nid00196
```

then the full path to the directory containing the queue configuration files is:

```
/opt/xt-lsfhpc/conf/lsbatch/nid00196/configdir
```

2. Create a `ccm_queue` for Platform LSF. Refer to Platform documentation for information on managing LSF queues.
3. Enable `PRE_EXEC` and `POST_EXEC` scripts for the queue set up in [Create a General CCM Queue and Queues for Separate ISV Applications](#) on page 258 by setting the following parameters in `lsb.queues`:

```
QUEUE_NAME = ccm_queue
PRE_EXEC = /opt/cray/ccm/default/etc/lsf_ccm_pre
POST_EXEC = /opt/cray/ccm/default/etc/lsf_ccm_post
LOCAL_MAX_PREEEXEC_RETRY=1
DESCRIPTION=ccm_queue
```

To enable LSF using an application profile rather than a queue, set the following in `lsb.applications`:

```
Begin Application
NAME=ccm
DESCRIPTION=Sets up an application profile for CCM
PRE_EXEC=/opt/cray/ccm/default/etc/lsf_ccm_pre
POST_EXEC=/opt/cray/ccm/default/etc/lsf_ccm_post
LOCAL_MAX_PREEEXEC_RETRY=1
```

For more information on the `lsb.applications` file, see the *Platform LSF Configuration Reference* manual.

4. In the file `lsb.params` set the `JOB_INCLUDE_POSTPROC` to ensure that the job reservation remains in a running state until execution of the `POST_EXEC` script completes and all necessary clean up has finished:

```
JOB_INCLUDE_POSTPROC=Y
```

5. On the boot node shared root file system, update `/etc/lsf.sudoers` using `xtopview`:

```
boot:~ # xtopview
default:/:/ # vi /etc/lsf.sudoers
```

Make the `root` user the `LSB_PRE_POST_EXEC_USER`:

```
LSB_PRE_POST_EXEC_USER=root
```

- Exit the editor and change the default permissions for `/etc/lsf.sudoers` so that the batch system infrastructure can properly communicate with compute nodes:

```
default:/:/ # chmod 600 /etc/lsf.sudoer
```

- Exit `xtopview`.
- When the configuration of the system is completed and the compute nodes are started, verify that write permissions are correct by using `touch` to create a dummy file within CCM:

```
% ccmrun touch foo
```

If `foo` is created in the user directory then the write permissions are set correctly.

Create Custom Resources with PBS

- Edit the `/var/spool/PBS/server_priv/resourcedef` file on the SDB node.
- Add the following line:

```
ccm type=boolean
```

- Invoke one of the following commands to create a custom resource.

- For batch:

```
qsub -lmpwidth=width -lccm=1 job_script.pbs
```

- For interactive:

```
qsub -I -lmpwidth=width -lccm=1 ./job_script.pbs
```

Create Custom Resources with Moab

Moab custom resources are managed as generic global node resources. These can be configured in the `moab.cfg` file in the installed Moab spool directory `/var/spool/moab/moab.cfg`.

- Add the following entry to `moab.cfg` to allow up to 1024 `ccm` instances on the system at one time:

```
NODECFG[GLOBAL] GRES=ccm:1024
```

- To consume this resource at runtime, invoke the following command:

```
% qsub -I -lmpwidth=1 -lgres=ccm
```

The following information is set for the job: Resource_List.gres = ccm

InfiniBand and OpenFabrics Interconnect Drivers

InfiniBand (IB) and OpenFabrics remote direct memory access (RDMA) is supported on service nodes for Cray systems running the Cray Linux Environment (CLE) operating system.

No separate installation is required. The kernel-space libraries and drivers are built against Cray's kernel. OpenFabrics Enterprise Distribution (OFED™) and InfiniBand RPMs are included in the CLE release and installed by default. However, OFED will not run on the Cray system until the I/O nodes are configured to use IB.

To configure IB and OFED, see the procedures provided in this chapter; to configure IB and OFED during installation or upgrade of the CLE software, see *Install and Configuring Cray Linux Environment (CLE) Software (S-2444)*.

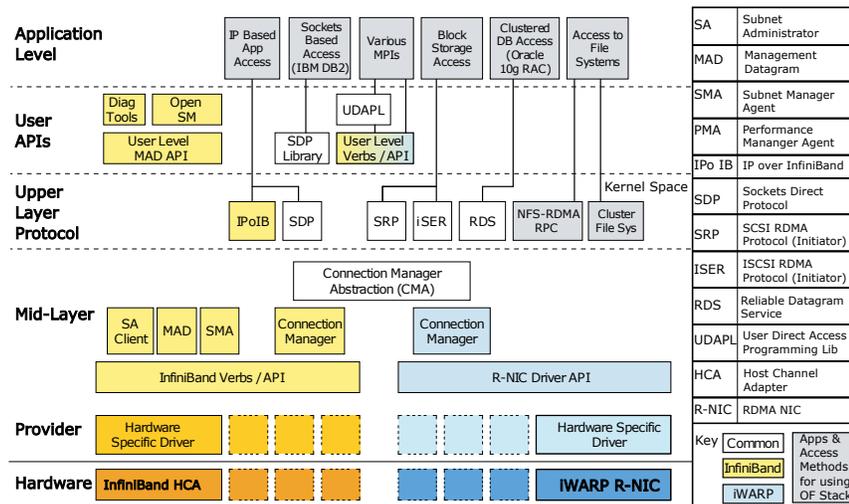
Cray supports InfiniBand as an I/O interconnect. IB enables efficient zero-copy, low-latency RDMA transfers between network peers. As a result, IB gives Cray systems the most efficient transfer mechanism from the high speed network (HSN) to external I/O devices.

CLE includes a subset of the OpenFabrics Enterprise Distribution (OFED) to support the use of InfiniBand on Cray I/O nodes. OFED is the software stack on the host that coordinates user-space and kernel-space access to the IB hardware. IB support is restricted to I/O service nodes that are equipped with PCI Express (PCIe) cards for network connectivity.

IB can be used on Lustre router nodes as a network interconnect between the Cray system and external Lustre servers.

The OFED software stack consists of many different components. These components can be categorized as kernel modules (drivers) and user/system libraries and utilities, commands and daemons for InfiniBand administration, configuration, and diagnostics; Cray maintains the kernel modules so that they are compatible with CLE.

Figure 7. The OFED Stack (source: OpenFabrics Alliance)



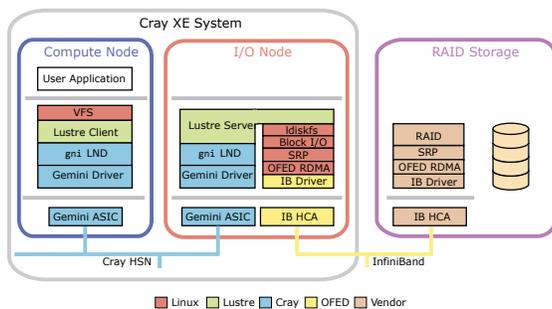
InfiniBand Uses

InfiniBand is a payload-agnostic transport. It can move small messages or large blocks efficiently between network endpoints. The following examples demonstrate how Cray uses InfiniBand and the OFED stack to support block I/O, file I/O, and standard network inter-process communication.

Storage Area Networking

InfiniBand can transport block I/O requests to external storage targets. ANSI T10's SCSI RDMA Protocol (SRP) is currently the only SCSI-transporting protocol supported on Cray systems with InfiniBand. [Cray System Connected to Storage Using SRP](#) on page 263 shows SRP on InfiniBand connecting the Cray to an external RAID array. The OFED stack is shown in the storage array for clarity; it is provided by the site-specific third party storage vendor.

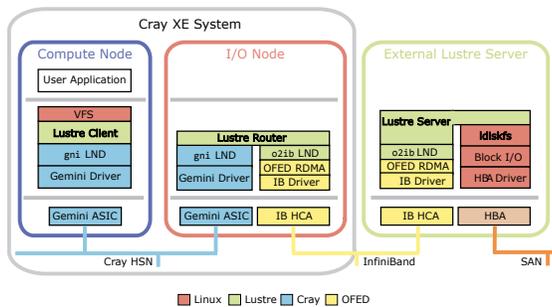
Figure 8. Cray System Connected to Storage Using SRP



Lustre Routing

Cray uses InfiniBand on the service nodes to connect Cray compute nodes to Lustre File System by Cray (CLFS) servers, as shown in [Cray Service Node Acting as an InfiniBand Lustre Router](#) on page 263. In this configuration, the Cray service node is no longer a Lustre server. Instead, it runs a Lustre router provided by the LNET layer. The router moves LNET messages between the Cray HSN and the external IB network, which transports file-level I/O requests between the clients on the Cray HSN and the servers over the IB fabric. Please speak with a Cray service representative regarding an CLFS solution.

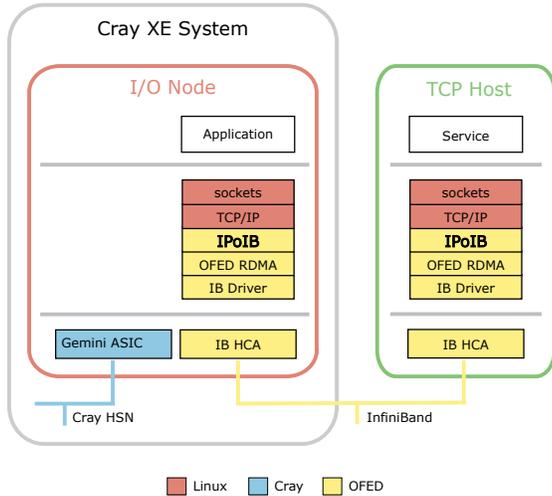
Figure 9. Cray Service Node Acting as an InfiniBand Lustre Router



IP Connectivity

InfiniBand can also carry socket-based inter-process traffic typical of commodity clusters and TCP/IP networking. InfiniBand supports the IP over IB (IPoIB) protocol. Since IB plugs-in below the socket interface, neither the application nor the service needs to be recompiled to communicate over an InfiniBand network. Both protocols are diagrammed on a service node in *Cray Service Node in IP over IB Configuration* on page 264.

Figure 10. Cray Service Node in IP over IB Configuration



Upper Layer InfiniBand I/O Protocols

In addition to the OFED RDMA stack, Cray supports three upper layer protocols (ULPs) on its service nodes as shown in the table. Because all ULPs use the OFED stack, the InfiniBand configuration must be followed for all IB service nodes. It is only necessary to configure the specific ULPs intended for use on the service node.

For example, a Lustre server with an IB direct-attached storage array uses the SCSI RDMA Protocol (SRP), not the LNET Router. On the other hand, if the Lustre servers are external to the Cray system, the service node uses the LNET router instead of SRP. IP over InfiniBand (IPoIB) is used to connect non-RDMA socket applications across the IB network.

Table 10. Upper Layer InfiniBand I/O Protocols for Cray Systems

Upper Layer Protocol	Purpose
IP over IB (IPoIB)	Provides IP connectivity between hosts over IB.
Lustre (OFED LND)	Base driver for Lustre over IB. On service nodes, this protocol enables efficient routing of LNET messages from Lustre clients on the Cray HSN to external IB-connected Lustre servers. The name of the LND is <code>o2iblnd</code> .
SCSI RDMA Protocol (SRP)	T10 standard for mapping SCSI over IB and other RDMA fabrics. Supported by DDN and LSI for their IB-based storage controllers.

Configure InfiniBand on Service Nodes

InfiniBand includes the core OpenFabrics stack and a number of upper layer protocols (ULPs) that use this stack. Configure InfiniBand by modifying `/etc/sysconfig/infiniband` for each IB service node.

1. Use the `xtopview` command to access service nodes with IB HCAs.

For example, if the service nodes with IB HCAs are part of a node class called `lnet`, type the following command:

```
boot:~ # xtopview -x /etc/opt/cray/sdb/node_classes -c lnet
```

Or

Access each IB service node by specifying either a node ID or physical ID. For example, access node 27 by typing the following:

```
boot:~ # xtopview -x /etc/opt/cray/sdb/node_classes -n 27
```

2. Specialize the `/etc/sysconfig/infiniband` file:

```
node/27:/ # xtspec -n 27 /etc/sysconfig/infiniband
```

3. Add IB services to the service nodes by using standard Linux mechanisms, such as executing the `chkconfig` command while in the `xtopview` utility or executing `/etc/init.d/openibd start | stop | restart` (which starts or stops the InfiniBand services immediately). Use the `chkconfig` command to ensure that IB services are started at system boot.

```
node/27:/ # chkconfig --force openibd on
```

4. While in the `xtopview` session, edit `/etc/sysconfig/infiniband` and make these changes.

```
node/27:/ # vi /etc/sysconfig/infiniband
```

- a. By default, IB services do not start at system boot. Change the `ONBOOT` parameter to `yes` to enable IB services at boot.

```
ONBOOT=yes
```

- b. By default at boot time, the Internet Protocol over InfiniBand (IPoIB) driver loads on all nodes where IB services are configured. Verify that the value for `IPOIB_LOAD` is set to `yes` to enable IPoIB services.

```
IPOIB_LOAD=yes
```

IMPORTANT: LNET routers use IPoIB to select the paths that data will travel via RDMA.

- c. The SCSI RDMA Protocol (SRP) driver loads by default on all nodes where IB services are configured to load at boot time. If the Cray system needs SRP services, verify that the value for `SRP_LOAD` is set to `yes` to enable SRP.

```
SRP_LOAD=yes
```

IMPORTANT: Direct-attached InfiniBand file systems require SRP; Lustre file systems external to the Cray system do not require SRP.

5. Exit `xtopview`.

```
node/27:/ # exit
boot:~ #
```

NOTE: The system administrator is prompted to type `c` and enter a brief comment describing the changes made. To complete the comment, type `Ctrl-d` or a period on a line by itself. Do this each time `xtopview` is exited to log a record of revisions into an RCS system.

6. Proper IPoIB operation requires additional configuration. See [Configure IP Over InfiniBand \(IPoIB\) on Cray Systems](#) on page 267.

Subnet Manager (OpenSM) Configuration

InfiniBand fabrics require at least one Subnet Manager (SM) operating on each IB subnet in order to activate its respective IB port connected to the fabric. This is one critical difference between IB fabrics and Ethernet, where simply connecting a cable to an Ethernet port is sufficient to get an active link. Managed IB switches typically include an SM and, therefore, do not require any additional configuration of any of the hosts. Unmanaged IB switches, which are considerably less expensive, do not include a SM and, thus, at least one host connected to the switch must act as a subnet manager. InfiniBand standards also support switchless (point-to-point) connections as long as an SM is installed. An example of this case is when a service node is connected to direct-attached storage through InfiniBand.

The OpenFabrics distribution includes OpenSM, an open-source IB subnet management and subnet administration utility. Either one of the following configuration steps is necessary if no other subnet manager is available on the IB fabric. The subnet manager RPMs are installed in the shared root by running `CLEinstall`. OpenSM can be started from the service node on a single port at boot time or manually from the command line to load multiple instances per host.

Start OpenSM at Boot Time

Follow this procedure to start a single instance of OpenSM on a service node at boot time.

This procedure assumes that the IB HCA is in node 8.

1. Access the service node that will host the instance of OpenSM.

```
boot:~ # xtopview -x /etc/opt/cray/sdb/node_classes -n 8
```

2. Specialize `/etc/sysconfig/opensm` for the IB node.

```
node/8:/ # xtspec -n 8 /etc/sysconfig/opensm
```

3. Edit `/etc/sysconfig/opensm` to have OpenSM start at boot time

```
# To start OpenSM automatically set ONBOOT=yes
ONBOOT=yes
```

4. Add IB services to the service nodes by using standard Linux mechanisms, such as executing the `chkconfig` command while in the `xtopview` utility or executing `/etc/init.d/opensmd start|stop|restart|status` (which starts or stops the OpenSM service immediately). The `chkconfig` command can be used to ensure that the OpenSM service is started at system boot.

```
node/8:/ # /sbin/chkconfig --force opensmd on
```

Configure IP Over InfiniBand (IPoIB) on Cray Systems

1. Use `xtopview` to access each service node with an IB HCA by specifying either a node ID or physical ID. For example, to access node 27, type the following:

```
boot:~ # xtopview -x /etc/opt/cray/sdb/node_classes -n 27
```

2. Specialize the `/etc/sysconfig/network/ifcfg-ib0` file.

```
node/27:/ # xtspec -n 27 /etc/sysconfig/network/ifcfg-ib0
```

3. Modify the site-specific `/etc/sysconfig/network/ifcfg-ib0` file on each service node with an IB HCA.

```
node/27:/ # vi /etc/sysconfig/network/ifcfg-ib0
```

For example, to use static IP address, `172.16.0.1`, change the `BOOTPROTO` line in the file.

```
BOOTPROTO='static'
```

Add the following lines to the file.

```
IPADDR='172.16.0.1'
NETMASK='255.128.0.0'
```

To configure the interface at system boot, change the `STARTMODE` line in the file.

```
STARTMODE='onboot'
```

4. Optional: To configure IPoIB for another IB interface connected to this node, repeat step 2 on page 267 and step 3 on page 267 for `/etc/sysconfig/network/ifcfg-ibn`. For LNET traffic, each IB interface should be assigned a unique IP address from the subnet that it will operate on. For TCP/IP traffic, multiple IB interfaces on a node must be assigned unique IP addresses from different subnets.

Configure SCSI RDMA Protocol (SRP) on Cray Systems

1. Use the `xtopview` command to access service nodes with IB HCAs.

For example, if the service nodes with IB HCAs are part of a node class called `ib`, type the following command:

```
boot:~ # xtopview -x /etc/opt/cray/sdb/node_classes -c ib
```

2. Edit `/etc/sysconfig/infiniband` and change the value of `SRP_DAEMON_ENABLE` to `yes`:

```
ib:// # vi /etc/sysconfig/infiniband
```

```
SRP_DAEMON_ENABLE=yes
```

3. Edit `srp_daemon.conf` to increase the maximum sector size for SRP.

```
ib:// # vi /etc/srp_daemon.conf
```

```
a      max_sect=8192
```

4. Edit `/etc/modprobe.conf.local` to increase the maximum number of gather-scatter entries per SRP I/O transaction.

```
ib:// # vi /etc/modprobe.conf.local
```

```
options ib_srp srp_sg_tablesize=255
```

5. Exit from `xtopview`.

```
ib:// # exit
```

Lustre Networking (LNET) Router

LNET is the Lustre networking layer. LNET isolates the file system code from the Lustre Networking Drivers (LNDs), which provide an interface to the underlying network transport. For more information on Lustre networking please see the Lustre documentation at <http://wiki.whamcloud.com/display/PUB/Documentation>.

Although LNET is automatically loaded with the Lustre servers and clients, it can be launched by itself to create a standalone router between networks instantiated by LNDs. LNET routing is most efficient when the underlying transports are capable of remote direct memory access (RDMA). Cray Lustre currently supports LNDs for a number of RDMA transports, including `gnilnd`, which is used for Cray systems using the Gemini or Aries interconnects, and `o2iblnd`, which is used for the OpenFabrics InfiniBand stack. Cray builds and distributes the OFED LND (`o2iblnd`) and `gnilnd` as part of its Lustre distribution.

Note that LNET routing is also available over GigE and 10GigE networks with `socklnd`, although this configuration does not support RDMA.

Routing Lustre involves configuring router nodes to route traffic between Lustre servers and Lustre clients which are on different networks. Routing Lustre requires that three types of nodes be configured: the router, the client, and the InfiniBand server. LNET assigns node IDs to each node on the network. While `gnilnd` uses node IDs (for example, `nnn@gni`) to enumerate ports on the Cray side, `o2iblnd` uses InfiniBand IP addresses (for example, `nn.nn.nnn.nnn@o2ib`) for hosts. As a result, IPoIB must be configured on each IB port. See [Configure IP Over InfiniBand \(IPoIB\) on Cray Systems](#) on page 267 for more information. For the rest of this discussion, assume that LNET routers are being created on two Cray service nodes, both of which have a single IB port connected to a switched InfiniBand fabric.

Table 11. LNET Network Address Configuration for Cray Systems

gni Address	Network Component	InfiniBand Address
27	Router 1	10.10.10.28
31	Router 2	10.10.10.32
10.128.0.255	IP Subnet	10.10.10.255
255.255.255.0	Subnet Mask	255.255.255.0

Configure the LNET Routers

The following description covers the configuration of the LNET router nodes.

1. Use `xtopview` to access the default view of the shared root.

```
boot:~ # xtopview -x /etc/opt/cray/sdb/node_classes
```

2. Create the `/etc/init.d/lnet` router controller (RC) script. An RC script is necessary to start LNET in the absence of any Lustre file services.

IMPORTANT: Cray does not provide an RC script with its release packages, but a sample script is provided here. Verify that this script will work for the system configuration or contact a Cray service representative for more information.

```
#lnet.rc
#!/bin/bash
#
# $Id: lnet.rc bogl Exp $
#
### BEGIN INIT INFO
# Provides:          lnet
# Required-Start:    $network openibd
# Required-Stop:     $network openibd
# X-UnitedLinux-Should-Start:
# Default-Start:     3
# Default-Stop:      0 1 2 5 6
# Description:       Enable lnet routers
### END INIT INFO
#set -x
PATH=/bin:/usr/bin:/usr/sbin:/sbin:/opt/cray/lustre-cray_gem_s/default/sbin
. /etc/rc.status
rc_reset
case "$1" in
    start)
        echo -n "Starting lnet "
        modprobe lnet
        lctl net up > /dev/null
        rc_status -v
        ;;
    stop)
        echo -n "Stopping lnet "
        lctl net down > /dev/null
        lustre_rmmod || true
        rc_status -v
        ;;
esac
```

```

restart)
    $0 stop
    $0 start
    rc_status
    ;;
*)
    echo "Usage: $0 {start|stop|restart}"
    exit 1
    ;;
esac
rc_exit

```

- a. Create the `/etc/init.d/lnet` file in the default view.

```
default/:// # vi /etc/init.d/lnet
```

- b. Copy the sample script into the new file and write it.

3. Use `chkconfig` to enable LNET because there are no mounts or Lustre server activity to load the LNET module implicitly.

```
default/:// # /sbin/chkconfig lnet on
```

4. Add the following LNET directives to the Cray shared root in `/etc/modprobe.conf.local`.

```
options lnet ip2nets="gni0 10.128.0.*; o2ib 10.10.10.*"
options lnet routes="gni0 10.10.10.[28,32]@o2ib; o2ib [27,31]@gni0"
```

NOTE: Larger system configuration LNET directives may exceed the 1024 character limit of `modprobe.conf` entries. [Specify Service Node LNET routes and ip2nets Directives with Files](#) allows administrators to source `ip2nets` and `routes` information from files to work around this limitation.

`o2ib` is the LNET name for the OFED LND and `gni` is the LNET name for the Cray LND. Here `ip2nets` is used instead of `networks` because it provides for an identical `modprobe.conf` across all Lustre clients in the Cray system.

The `ip2nets` directive tells LNET to load both LNDs and associates each LND with an IP subnet. It overrides any previous `networks` directive (for example, `lnet networks=gni`). On service nodes without an IB adapter, the `o2ib` LND does not load because there are no ports with the IP subnet used defined in `ip2nets`.

Each Cray system sharing an external Lustre file system must have a unique `gni` identifier for the LNET options listed in this step. In this case, the Cray system is using `gni0`. Other systems would use other numbers to identify their LNET networks (such as `gni1`, `gni2`, and so on).

5. Cray recommends enabling these options to improve network resiliency. Edit `/etc/modprobe.conf.local` on the Cray shared root to include:

```
options ko2iblnd timeout=100 peer_timeout=130
options ko2iblnd credits=2048 ntx=2048
options ko2iblnd peer_credits=126 concurrent_sends=63 peer_buffer_credits=128
options kgnilnd credits=2048 peer_health=1

options lnet check_routers_before_use=1
options lnet dead_router_check_interval=60
options lnet live_router_check_interval=60
```

```
options lnet router_ping_timeout=50
options lnet large_router_buffers=1024 small_router_buffers=16384
```

NOTE: These values must be configured for the specific system size.

6. Exit from xtopview.

xtopview prompts to add a comment about the operations performed. Enter **c**, and then enter a brief comment about the changes made to the file.

Specify Service Node LNET routes and ip2nets Directives with Files

1. Enter xtopview and create the following files (if they do not already exist):

```
boot:~ # xtopview
default:/: # touch /etc/lnet_ip2nets.conf
default:/: # touch /etc/lnet_routes.conf
```

2. Exit xtopview.

```
default:/: # exit
```

3. Enter the correct view of xtopview for the service node or node class for which the `modprobe` directives are to be modified and specialize the two files created in step 1 on page 271.

```
boot:~ # xtopview -c login
class/login:/ # xtspec /etc/lnet_ip2nets.conf
class/login:/ # xtspec /etc/lnet_routes.conf
```

4. Add the following lines to `/etc/lnet_ip2nets.conf` (substitute the site-specific values here).

```
gni0 10.128.0.*
o2ib 10.10.10.*
```

5. Add the following lines to `/etc/lnet_routes.conf` (substitute the site-specific values here).

```
gni0 10.10.10.[28,32]@o2ib
o2ib [27,31]@gni0
```

6. Add the following LNET directives to the Cray shared root in `/etc/modprobe.conf.local`.

```
options lnet ip2nets="/etc/lnet_ip2nets.conf"
options lnet routes="/etc/lnet_routes.conf"
```

7. Exit xtopview.

```
class/login:/ # exit
```

Manually Control LNET Routers

If `/etc/init.d/lnet` is not provided, send the following commands to each LNET router node to control them manually:

- For startup:

```
modprobe lnet
lctl net up
```

- For shutdown:

```
lctl net down
lustre_rmmmod
```

Configure the InfiniBand Lustre Server

The Lustre servers must be configured with proper routes to allow them to reach the Cray client nodes on the `gni` network. If there are multiple Cray systems involved, each Cray must use a unique `gni` network identifier (for example, `gni0`, `gni1`). The server `routes` configuration maps each `gni` network to the corresponding Cray router nodes. Add an `lnet routes` directive for each Cray system. Perform these steps on the remote host:

- Edit `/etc/modprobe.conf` on the remote host to include the route to the LNET network.

```
options lnet ip2nets="o2ib"
options lnet routes="gni0 10.10.10.[28,32]@o2ib"
```

If there are two Cray systems accessing the file system exported by these hosts, then both Cray systems must be included in the `lnet routes` directive.

```
options lnet routes="gni0 10.10.10.[28,32]@o2ib; \
gni1 10.10.10.[71,72,73,74]@o2ib"
```

In this example, there are two Cray systems: `gni0` with two router nodes and `gni1` with four.

- Add the following options to `/etc/modprobe.conf` to improve network resiliency :

```
options ko2iblnd timeout=100 peer_timeout=0 keepalive=30
options ko2iblnd credits=2048 ntx=2048
options ko2iblnd peer_credits=126 concurrent_sends=63

options lnet avoid_asym_router_failure=1
options lnet dead_router_check_interval=60
options lnet live_router_check_interval=60
options lnet router_ping_timeout=50
options lnet check_routers_before_use=1
```

Because Lustre is running on the external host, there is no need to start LNET explicitly.

Configure the LNET Compute Node Clients

Because compute nodes are running the Lustre client, they do not need explicit commands to start LNET. There is, however, additional configuration required for compute nodes to be able to access the remote Lustre servers via the LNET routers. These changes are made to `/etc/modprobe.conf` for the compute node image used in booting the system.

1. Edit `/etc/modprobe.conf` for the compute node boot image. The `lnet ip2nets` directive identifies the LND. If there is more than one Cray system sharing the file system, then this identifier (`gni`) must be unique for each Cray system.

```
options lnet ip2nets="gni0"
options lnet routes="o2ib [27,31]@gni0"
```

2. Add the following options to `/etc/modprobe.conf` to improve network resiliency :

```
options lnet avoid_asym_router_failure=1
options lnet dead_router_check_interval=60
options lnet live_router_check_interval=60
options lnet router_ping_timeout=50
options lnet check_routers_before_use=1
```

3. For InfiniBand-connected LNET clients (such as Cray Development and Login (CDL) nodes), add the following options to `/etc/modprobe.conf`:

```
options ko2iblnd timeout=100 peer_timeout=0 keepalive=30
options ko2iblnd credits=2048 ntx=2048
options ko2iblnd peer_credits=126 concurrent_sends=63
```

4. Modify `/etc/fstab` in the compute node boot image to identify the external server. The file system is described by the LNET node ID of the MGS server (and its failover partner if Lustre failover is configured)

```
10.10.10.1@o2ib:10.10.10.2@o2ib: /boss1 /mnt/boss1 lustre rw,flock,lazystatfs
```

Here, the `fstab` mount option `rw` gives read/write access to the client node. The `flock` option is to allow Lustre's client node to have exclusive access to the file lock, and the `lazystatfs` option prevents command hangs (such as `df`) if one or more OSTs are unavailable.

In this example, the Lustre file system with the `fsname` "boss1" is provided by the Lustre management server (MGS) on the InfiniBand fabric at IP address `10.10.10.1` (with `10.10.10.2` being the failover partner). Because both routers have access to this subnet, the Lustre client performs a round-robin with its requests to the routers.

5. Update the boot image to include these changes; follow the steps in [Prepare Compute and Service Node Boot Images](#) on page 37.

IMPORTANT: Accessing any externally supplied Lustre file system requires that both the file server hosts and the LNET routers be up and available before the clients attempt to mount the file system. Boot time scripts in the compute node image take care of reading `fstab` and running the necessary `mount` commands. In production, this is the only opportunity to run Lustre `mount` commands because kernel modules get deleted at the end of the boot process.

Remove an LNET Router from a Live System

The basic steps for removing an LNET router from a live system are:

1. Modify the LNET route table to remove the target router.
2. Wait for any queued messages on the router to drain.
3. Remove the router.

See the examples that follow for details.

Modify routes on a server

Issue the `lctl` command as `root` on each server utilizing the route being modified. For a flat LNET, all servers require route table modification. For fine-grained routing (FGR) schemes, only those servers belonging to the FGR group require route table modification. In this example, the administrator is logged on to an MDS as `root`, but the same commands can be executed in parallel on multiple servers using the `pdsh` command or something similar.

```
mds1# lctl show_route
net          gni hops 1 gw          10.149.11.56@o2ib down
pri 0
mds1# lctl del_route 10.149.11.56@o2ib
mds1# lctl show_route
mds1# lctl --net gni add_route 10.149.11.56@o2ib
mds1# lctl show_route
net          gni hops 1 gw          10.149.11.56@o2ib
down pri 0
```

Modify routes on a service node client

```
login1# lctl show_route
net          o2ib hops 1 gw          57@gni up
pri 0
net          o2ib hops 1 gw          56@gni up
pri 0
login1# lctl del_route 57@gni
login1# lctl show_route
net          o2ib hops 1 gw          56@gni up
pri 0
login1# lctl --net o2ib add_route 57@gni
login1# lctl show_route
net          o2ib hops 1 gw          56@gni up
pri 0
net          o2ib hops 1 gw          57@gni up
pri 0
```

Modify routes on a single compute node

Log on to the compute node and prepend `/ds1` to the `lctl` path.

```
login1# which lctl
/opt/cray/lustre-cray_gem_s/ \
2.5_3.0.82_0.7.9_1.0000.7715.35.1-1.0000.14681.27.2/sbin/lctl
login1# ssh nid00063
root@nid00063's password:
```

```

Welcome to the initramfs
# /dsl/opt/cray/lustre-cray_gem_s/ \
2.5_3.0.82_0.7.9_1.0000.7715.35.1-1.0000.14681.27.2/sbin/lctl
show_route
net                o2ib hops 1 gw                57@gni up
pri 0
net                o2ib hops 1 gw                56@gni up
pri 0
# /dsl/opt/cray/lustre-cray_gem_s/ \
2.5_3.0.82_0.7.9_1.0000.7715.35.1-1.0000.14681.27.2/sbin/lctl
del_route 57@gni
# /dsl/opt/cray/lustre-cray_gem_s/ \
2.5_3.0.82_0.7.9_1.0000.7715.35.1-1.0000.14681.27.2/sbin/lctl
show_route
net                o2ib hops 1 gw                56@gni up
pri 0
# /dsl/opt/cray/lustre-cray_gem_s/ \
2.5_3.0.82_0.7.9_1.0000.7715.35.1-1.0000.14681.27.2/sbin/lctl --net
o2ib add_route 57@gni
# /dsl/opt/cray/lustre-cray_gem_s/ \
2.5_3.0.82_0.7.9_1.0000.7715.35.1-1.0000.14681.27.2/sbin/lctl
show_route
net                o2ib hops 1 gw                56@gni up
pri 0
net                o2ib hops 1 gw                57@gni up
pri 0

```

Modify routes on multiple compute nodes

Use the `pcmd` command to change multiple compute nodes at once. In this example, a single target node is specified to reduce the command output. Specify `-n ALL_COMPUTE` to target all compute nodes in the system. The commands below are executed as `root` on a login node.

```

login1# module load nodehealth
login1# which lctl
/opt/cray/lustre-cray_gem_s/ \
2.5_3.0.82_0.7.9_1.0000.7715.35.1-1.0000.14681.27.2/sbin/lctl
login1# pcmd -n nid00063 "/opt/cray/lustre-cray_gem_s/ \
2.5_3.0.82_0.7.9_1.0000.7715.35.1-1.0000.14681.27.2/sbin/lctl
show_route"
Output from node nid00063:
net                o2ib hops 1 gw                56@gni up
pri 0
net                o2ib hops 1 gw                57@gni up
pri 0
Reply (complete) from nid00063 exit code: 0
login1# pcmd -n nid00063 "/opt/cray/lustre-cray_gem_s/ \
2.5_3.0.82_0.7.9_1.0000.7715.35.1-1.0000.14681.27.2/sbin/lctl
del_route 56@gni"
Reply (complete) from nid00063 exit code: 0
login1# pcmd -n nid00063 "/opt/cray/lustre-cray_gem_s/ \
2.5_3.0.82_0.7.9_1.0000.7715.35.1-1.0000.14681.27.2/sbin/lctl
show_route"
Output from node nid00063:
net                o2ib hops 1 gw                57@gni up
pri 0
Reply (complete) from nid00063 exit code: 0
login1# pcmd -n nid00063 "/opt/cray/lustre-cray_gem_s/ \
2.5_3.0.82_0.7.9_1.0000.7715.35.1-1.0000.14681.27.2/sbin/lctl --net

```

```

o2ib add_route 56@gni"
Reply (complete) from nid00063 exit code: 0
login1# pcmd -n nid00063 "/opt/cray/lustre-cray_gem_s/ \
2.5_3.0.82_0.7.9_1.0000.7715.35.1-1.0000.14681.27.2/sbin/lctl
show_route"
Output from node nid00063:
net                o2ib hops 1 gw                57@gni up
pri 0
net                o2ib hops 1 gw                56@gni up
pri 0
Reply (complete) from nid00063 exit code: 0

```

Tune Lustre Networking

This section documents Lustre networking (LNET) and Lustre network driver (LND) module parameters to use for configuring and tuning LNET and its features on Cray systems. This information is applicable for Lustre 2.4.x or later.

LNET Default Module Parameters

The default values, in parenthesis, for the Lustre network (LNET) module parameters follow.

forwarding	(" ") Explicitly enable/disable forwarding between networks. Set to "enabled" to explicitly enable, or "disabled" to explicitly disable.
tiny_router_buffers	(512 per CPT minimum) The number of 0 payload messages to buffer in the router.
small_router_buffers	(4096 per CPT minimum) The number of small (1 page) messages to buffer in the router.
large_router_buffers	(256 per CPT minimum) Number of large (> 1 page) messages to buffer in the router.
peer_buffer_credits	(Gets value from LND peer_buffer_credits) The number of router buffer credits per peer.
auto_down	(1) Automatically mark peers down on communications error (0 to disable; 1 to enable).
check_routers_before_use	(0) Assume routers are down and ping them before use (0 to disable; 1 to enable).
avoid_asym_router_failure	(1) Avoid asymmetrical router failures (0 to disable; 1 to enable).
dead_router_check_interval	(60) The number of seconds between dead router health checks (<= 0 to disable).
live_router_check_interval	(60) The number of seconds between live router health checks (<= 0 to disable).
router_ping_timeout	(50) The number of seconds to wait for the reply to a router health query.
config_on_load	(0) Configure network at module load (0 to disable; 1 to enable).
portal_rotor	Redirect PUTs to different cpu-partitions.
ip2nets	(N/A) LNET network -> IP Table.
networks	(N/A) Local networks.

routes	(N/A) Routes to non-local networks.
rnet_htable_size	The size of remote network hash table. The hash table size is the number of bits it takes to express the set <code>ln_num_routes</code> , minus 1.

LND Default Module Parameters for `gnilnd`

The default values, in parenthesis, for the `gnilnd` Lustre network driver (LND) module parameters follow.

credits	(256) The number of concurrent sends.
eager_credits	(256 * 1024) The Number of eager buffers.
peer_credits	(16) The number of per-peer send credits. Sending a single message to a peer requires one peer credit.
concurrent_sends	(0) The number of concurrent HW sends to a single peer.
fma_cq_size	(32768) The number of entries in the completion queue to allocate.
timeout	(60) The communications timeout in seconds.
min_reconnect_interval	(60/4) The minimum time to wait between reconnection attempts.
max_reconnect_interval	(60) The maximum time to wait between reconnection attempts.
max_immediate	((2<<10)) The immediate/RDMA breakpoint.
checksum	(0 for Aries; 3 for Gemini) Enables/disables checksumming.
checksum_dump	(0) Enable/disable additional debugging.
bte_dlvr_mod	(GNILND_RDMA_DLVR_OPTION) Enable hashing for BTE (RDMA) transfers.
bte_relaxed_ordering	(1) Enable relaxed ordering (PASSPW) for BTE (RDMA) transfers.
ptag	() The ptag for Gemini CDM.
max_retransmits	(1024) Maximum number of retransmits for FMA.
nwildcard	(4) The number of wildcard datagrams to post per net (interface).
nice	(-20) The nice value for <code>kgnilnd</code> threads.
rdmaq_intervals	(4) The number of intervals per second for <code>rdmaq</code> throttling.
loops	(100) The number of loops before scheduler is friendly.
hash_size	(503) The prime number for peer/conn hash sizing.
peer_health	(0) Disable peer timeout for LNET peer health.
peer_timeout	(Based on the <code>gnilnd</code> timeout) Peer timeout used for <code>peer_health</code> .
vmap_cksum	(0) Use <code>vmap</code> for all <code>kiiov</code> checksumming.
mbox_per_block	(GNILND_FMABLK) The number of mailboxes per block.
nphys_mbox	(0) The number of mailboxes to preallocate from physical memory.
mbox_credits	(GNILND_MBOX_CREDITS) The number of credits per mailbox.
sched_threads	(GNILND_SCHED_THREADS) The number of threads for moving data.
net_hash_size	(11) The prime number for net hash sizing.
hardware_timeout	(GNILND_HARDWARE_TIMEOUT) The maximum time for traffic to get from one node to another.

mdd_timeout	(GNILND_MDD_TIMEOUT) The maximum time in minutes for mdd to be held.
sched_timeout	(GNILND_SCHED_TIMEOUT) Scheduler aliveness in seconds max time.
sched_nice	(0 for compute nodes; -20 for service nodes) The scheduler's nice setting.
reverse_rdma	(0) Normal: 0; Reverse GET: 1; Reverse PUT: 2; Reverse both: 3.
dgram_timeout	(GNILND_DGRAM_TIMEOUT) <code>dgram</code> thread aliveness max time in seconds.
efault_lbug	(0) Enable/Disable LBUG when a compute node receives an EFAULT in a message.

LND Default Module Parameters for `o2iblnd`

The default values, in parenthesis, for the `o2iblnd` Lustre network driver (LND) module parameters follow.

service	(987) The service number within <code>RDMA_PS_TCP</code> .
cksum	(0) Set to non-zero to enable message (not RDMA) checksums.
timeout	(50) The timeout in seconds.
nscheds	(0) The number of threads in each scheduler pool. If set to zero, an estimated reasonable value based on the number of CPUs is used.
ntx	(512) The number of message descriptors allocated for each pool.
credits	(256) The number of concurrent sends. Shared by all CPTs.
peer_credits	(8) The number of concurrent sends to a single peer.
peer_credits_hiw	(0) When to eagerly return credits—sets high water mark.
peer_buffer_credits	(0) The number of per-peer router buffer credits.
peer_timeout	(180) The number of seconds without aliveness news to declare a peer dead (≤ 0 to disable).
ipif_name	(ib0) The IPoIB interface name.
retry_count	(5) The number of retransmissions when no ACK received.
rnr_retry_count	(5) The number of RNR retransmissions.
keepalive	(100) The idle time in seconds before sending a keepalive.
ib_mtu	(0) IB MTU 256/512/1024/2048/4096.
concurrent_sends	(0) Send work queue sizing. When the default is used, <code>concurrent_sends</code> is calculated based on <code>peer_credits</code> and <code>map_on_demand</code> . See below for more information.
map_on_demand	(0) Map on demand.
fmr_pool_size	(512) The size of the FMR pool on each CPT ($\geq ntx / 4$).
fmr_flush_trigger	(384) The number of dirty FMRs that triggers a pool flush.
fmr_cache	(1) Set to non-zero to enable FMR caching.
pmr_pool_size	(512) The size of MR cache PMR pool on each CPT.
dev_failover	(0) HCA failover for bonding (0 off, 1 on, other values reserved).
require_privileged_port	(0) Require privileged port when accepting a connection (0 off, 1 on).
use_privileged_port	(1) Use privileged port when initiating a connection.

Router Node Recommended LNET Parameters

```
## ko2iblnd parameters
options ko2iblnd timeout=10
options ko2iblnd peer_timeout=40
options ko2iblnd credits=2048
options ko2iblnd ntx=2048
### Note peer_credits must be consistent across all peers on the IB network
options ko2iblnd peer_credits=126
options ko2iblnd concurrent_sends=63
options ko2iblnd peer_buffer_credits=128

## kgnilnd parameters
options kgnilnd credits=2048
options kgnilnd peer_health=1

## LNet parameters
options lnet large_router_buffers=1024
options lnet small_router_buffers=16384
```

Additional Information About LNET Router Buffer Parameters

```
large_router_buffers=1024
```

- Routers only.
- Sets the number of large buffers (greater than one page) on a router node.
- Divided equally among CPU partitions (CPTs).

```
small_router_buffers=16384
```

- Routers only.
- Sets the number of small buffers (one page) on a router node.
- Divided equally among CPTs.

LNET Feature: Router Pinger

The router pinger determines the status of configured routers so that bad (dead) routers are not used for LNET traffic.

The router pinger is enabled on clients and servers when the LNET module parameters `live_router_check_interval` and `dead_router_check_interval` have values greater than 0. The router pinger is always enabled on routers, though it is typically only used to update the status of local network interfaces, meaning it does not do any ping. In multi-hop configurations (server->router1->router2->client), the router pinger on a router behaves similarly to its behavior on other nodes, meaning it **does** do ping. The `router_checker` thread (router pinger) periodically sends traffic (an LNET ping) to each known router. Live routers are pinged every `live_router_check_interval` (in seconds). Dead routers are pinged every `dead_router_check_interval` (in seconds). If a response is not received from an alive route after a timeout

period, then the route is marked down and is not used for further LNET traffic. Dead routes are marked alive once a response is received.

The `router_checker` is also integral in the use of Asymmetric Routing Failure (ARF). The payload of the ping reply contains the status (up or down) of each of the router's network interfaces. This information is used to determine whether a particular router should be used to communicate with particular remote network.

The ping timeout is determined from the `router_pinger_timeout`, `dead_router_check_interval`, and `live_router_check_interval` module parameters. The effective maximum timeout is `router_pinger_timeout + MAX(dead_router_check_interval, live_router_check_interval)`. In the recommended tunings, `50 + MAX(60, 60) = 110` seconds.

LNET Feature: Asymmetric Router Failure Detection

Asymmetric router failure detection enables LNET peers to determine whether a route to a remote network is alive.

Peers use only known good routes. Attempting to send a message via a bad route generally results in a communication failure, requiring a resend. Sending via bad route also consumes resources on the router that could be utilized for other communication.

Asymmetric router failure detection is enabled by setting `avoid_asym_router_failure=1` in the LNET module settings. This feature piggy-backs off of the router pinger feature. The ping reply sent from router to clients and servers contains information about the status of that router's network interfaces for remote networks. Clients and servers then use this information to determine whether a particular router should be used when attempting to send a message to a remote network.

For example, assume a router at `454@gni` with an IB interface on `o2ib@1000` and another IB interface on `o2ib@1002`. Suppose this router responds to a router checker ping with the following information:

```
o2ib@1000 -> down
o2ib@1002 -> up
```

When a client wants to send a message to a remote network, it considers each configured router in turn. When considering `454@gni`, the client knows that this router can be used to send a message to `o2ib@1002`, but not to `o2ib@1000`.

External Server Node Recommended LNET Parameters

```
## ko2iblnd parameters
options ko2iblnd timeout=10
options ko2iblnd peer_timeout=0
options ko2iblnd keepalive=30
options ko2iblnd credits=2048
options ko2iblnd ntx=2048
### Note peer_credits must be consistent across all peers on the IB network
options ko2iblnd peer_credits=126
options ko2iblnd concurrent_sends=63

## LNet parameters
options lnet router_ping_timeout=10
options lnet live_router_check_interval=35
options lnet dead_router_check_interval=35
## Sonexion only (if off by default)
options lnet avoid_asym_router_failure=1
```

```
## ptlrpc parameters
options ptlrpc at_max=400
options ptlrpc at_min=40
options ptlrpc ldlm_enqueue_min=260
```

Cray recommends an object-based disk (OBD) timeout of 100 seconds, which is the default value. Set this parameter using the `lctl conf_param` on the management server (MGS):

```
$ lctl conf_param fs_name.sys.timeout=100
```

For example:

```
$ lctl conf_param husk1.sys.timeout=100
$ cat /proc/sys/lustre/timeout
100
$
```

Internal Server (DAL) Recommended LNET Parameters

```
## ptlrpc parameters
options ptlrpc at_max=400
options ptlrpc at_min=40
options ptlrpc ldlm_enqueue_min=260
```

DVS Server Node Recommended LNET Parameters

Use the default settings.

External Client (CDL) Recommended LNET Parameters

```
## o2iblnd parameters
options ko2iblnd timeout=10
options ko2iblnd peer_timeout=0
options ko2iblnd keepalive=30
options ko2iblnd credits=2048
options ko2iblnd ntx=2048
### Note peer_credits must be consistent across all peers on the IB network
options ko2iblnd peer_credits=126
options ko2iblnd concurrent_sends=63
```

Additional Information About PTLRPC Parameters

```
ldlm_enqueue_min=260
```

The `ldlm_enqueue_min` parameter sets the minimum amount of time a server is willing to wait to see traffic on a lock before assuming a client is misbehaving and taking action to revoke the lock by evicting the client. It is desirable that this value be large enough such that clients are able to resend an RPC from scratch without being

evicted in the event that the first RPC was lost. The time it takes for an RPC to be sent is the sum of the network latency and the time it takes for the server to process the request. In Lustre, both of these variables have a lower bound of `at_min`. Additionally, it should be large enough such that clients are not evicted as a result of the HSN quiesce period. Thus we calculate the minimum value as:

```
ldlm_enqueue_min = max(2*net latency, net latency + quiesce duration) +
2*service time =
max(2*40, 40 + 140) + 2*40 = 180 + 80 = 260
```

NOTE: The quiesce duration of 140 in the above equation was determined experimentally. It could be smaller or larger depending on the nature of the HSN failure or the size of the system. The quiesce duration of 140 strikes a balance between resiliency of Lustre against extended network flaps (larger `ldlm_enqueue_min`) and the ability for Lustre to detect misbehaving clients (smaller `ldlm_enqueue_min`).

LNET Feature: Peer Health

Peer health queries the interface a peer is on to determine whether it is alive or dead before allowing traffic to be sent to that peer. If a peer is dead, then LNET aborts the send. This functionality is needed to avoid communication attempts with known dead peers, which wastes network interface credits, router buffer credits, and other resources that could otherwise be used to communicate with alive peers.

Enable peer health by setting these LND module parameters:

- `gnilnd` — Set the `peer_health` and `peer_timeout` parameters.
- `o2iblnd` — Set the `peer_timeout` parameter. Setting this parameter to 0 disables peer health.

When a Lustre network driver (LND) completes a transmit, receive, or connection setup operation for a peer, it records the current time in a `last_alive` field associated with the peer. When a client of LNET (for example, `ptlrpc`) attempts to send anything to a particular peer, the `last_alive` value for that peer is inspected and, if necessary, updated by querying the LND. If the `last_alive` is more than `peer_timeout`seconds (plus a fudge factor for `gnilnd`), then the peer is considered dead and the message is dropped.

NOTE: The LND query serves a dual purpose. In addition to dropping a message, it causes the LND to attempt a new connection to a dead peer.

For routed configurations, disable peer health on clients and servers because clients and servers always have a peer in the middle (the router) and router aliveness is determined by the router checker feature. Peer health interferes with the normal operation of the router checker thread by preventing the router checker pings from being sent to dead routers. Thus it would be impossible to determine when dead routers become alive again.

Configure Fine-grained Routing with `clcvt`

The `clcvt` command, available on the boot node and the SMW, aids in the configuration of LNET fine-grained routing (FGR). FGR is a routing scheme that aims to group sets of Lustre servers on the file system storage array with LNET routers on the Cray system. This grouping maximizes file system performance on larger systems by using a router-to-server ratio where the relative bandwidth is roughly equal on both sides. FGR also minimizes the number of LNET network hops (hop count) and file system network congestion by sending traffic to particular Lustre servers over dedicated network lanes instead of the default round-robin configuration.

The `clcv` command takes as input several file-system-specific files and generates LNET kernel module configuration information that can be used to configure the servers, routers, and clients for that file system. The utility can also create cable maps in HTML, CSV, and human-readable formats and validate cable connection on installed systems. For more information, such as available options and actions for `clcv`, see the `clcv(8)` man page.

clcv Prerequisite Files

The `clcv` command requires several prerequisite files in order to compute the `ip2nets` and `routes` information for the specific configuration. Before `clcv` can be executed for the first time, these files must be placed in an empty directory on the boot node or SMW, depending on where `clcv` is run.

Deciding how to assign which routers to which OSSs, what FGR ratios to use, which interface on which router to use for a LNET group, and router placement are all things that can vary greatly from site to site. LNET configuration is determined as the system is ordered and configured; see a Cray representative for the site-specific values.

<code>info.file-system-identifier</code>	A file with global file system information for the <code>cluster-name</code> server machine and each client system that will access it.
<code>client-system.hosts</code>	A file that maps the client system (such as the Cray mainframe) IP addresses to unique host names, such as the boot node <code>/etc/hosts</code> file. The <code>client-system</code> name must match one of the <code>clients</code> in the <code>info.file-system-identifier</code> file.
<code>client-system.ib</code>	A file that maps the client system LNET router InfiniBand IP addresses to system hardware <code>cnames</code> . The <code>client-system</code> name must match one of the <code>clients</code> in the <code>info.file-system-identifier</code> file. This file must be created by an administrator.
<code>clustername.ib</code>	A file that maps the Lustre server InfiniBand IP addresses to cluster (for example, Sonexion) host names. The <code>clustername</code> name must match the <code>clustername</code> in the <code>info.file-system-identifier</code> file. This file must be created by an administrator.
<code>client-system.rtrIm</code>	A file that contains <code>rtr -Im</code> command output (executed on the SMW) for the <code>client-system</code> .

The info.file-system-identifier File

`info.file-system-identifier` is a manually-created file that contains global file system information for the Lustre server machine and each client system that will access it. Based on the ratio of server to LNET routers in the configuration, the `[clustername]` section and each `[client-system]` section will define which servers and routers will belong to each InfiniBand subnet.

This file is of the form of a `ini` style file, and the possible keywords in the `[info]` section include `clustername`, `ssu_count`, and `clients`.

`clustername` Defines the base name used for all file system servers. For a Sonexion file system, as in the example below, it might be something like `snxs11029n` and all server hostnames will be `snxs11029nWWW`, where `WWW` is a three digit number starting at `000` and `001` for the primary and secondary Cray Sonexion Management Servers (CSMS), `002` for the MGS, `003` for the MDS, `004` for the first OSS, and counting up from there for all remaining OSSs.

- ssu_count** Defines how many SSUs make up a Sonexion file system. If this is missing, then this is not a Sonexion file system but an CLFS installation.
- clients** Defines a comma-separated list of mainframe names that front-end this file system.

The `info.file-system-identifier` file also needs a `[client-system]` section for each client system listed in the `clients` line of the `[info]` section to describe the client systems and a `[clustername]` section to describe the Lustre server system. Each of these sections contain a literal `lnet_network_wildcard` in the format of `LNET-name:IP-wildcard` which instructs the LNET module to match a host's IP address to `IP-wildcard` and, if it matches, instantiate LNET `LNET-name` on them. [Sample `info.file-system-identifier` file: `info.snx11029`](#) on page 284 shows a sample `info.file-system-identifier` configuration file.

The hostname fields in the `[client-system]` section of this file are fully-qualified interface specifications of the form `hostname(ibn)`, where `(ib0)` is the assumed default if not specified.

Sample `info.file-system-identifier` file: `info.snx11029`

```
# This section describes the size of this filesystem.
[info]
clustername = snx11029n
SSU_count = 6
clients = hera

[hera]
lnet_network_wildcard = gni1:10.128.*.*

# Because of our cabling assumptions and naming conventions, we only
# need to know which XIO nodes are assigned to which LNETs. From
that
# our tool can actually generate a "cable map" for the installation
folks.
o2ib6000: c0-0c2s2n0, c0-0c2s2n2 ; MGS and MDS
o2ib6002: c1-0c0s7n0, c1-0c0s7n1, c1-0c0s7n2, c1-0c0s7n3 ; OSSs 2/4/6
o2ib6003: c3-0c1s5n0, c3-0c1s5n1, c3-0c1s5n2, c3-0c1s5n3 ; OSSs 3/5/7
o2ib6004: c3-0c1s0n0, c3-0c1s0n1, c3-0c1s0n2, c3-0c1s0n3 ; OSSs
8/10/12
o2ib6005: c3-0c2s4n0, c3-0c2s4n1, c3-0c2s4n2, c3-0c2s4n3 ; OSSs
9/11/13

[snx11029n]
lnet_network_wildcard = o2ib6:10.10.100.*

o2ib6000: snx11029n002, snx11029n003 ; MGS and MDS
o2ib6002: snx11029n004, snx11029n006, snx11029n008 ; OSSs 2/4/6
o2ib6003: snx11029n005, snx11029n007, snx11029n009 ; OSSs 3/5/7
o2ib6004: snx11029n010, snx11029n012, snx11029n014 ; OSSs 8/10/12
o2ib6005: snx11029n011, snx11029n013, snx11029n015 ; OSSs 9/11/13
```

The `client-system.hosts` File

For a typical Cray system, this file can be the `/etc/hosts` file taken from the boot node. Simply make a copy of the `/etc/hosts` file from the boot node and save it in a working directory to later run the `clcv` command.

Sample *client-system*.hosts file: *hera*.hosts

```

#
# hosts          This file describes a number of hostname-to-address
#                mappings for the TCP/IP subsystem.  It is mostly
#                used at boot time, when no name servers are running.
#                On small systems, this file can be used instead of a
#                "named" name server.
# Syntax:
#
# IP-Address    Full-Qualified-Hostname  Short-Hostname
#
127.0.0.1      localhost

# special IPv6 addresses
::1           ipv6-localhost  localhost      ipv6-loopback

fe00::0       ipv6-localnet

ff00::0       ipv6-mcastprefix
ff02::1       ipv6-allnodes
ff02::2       ipv6-allrouters
ff02::3       ipv6-allhosts
# Licenses
172.30.74.55   tic      tic.us.cray.com
172.30.74.56   tac      tac.us.cray.com
172.30.74.57   toe      toe.us.cray.com
172.30.74.206  cflls01  cflls01.us.cray.com
172.30.74.207  cflls02  cflls02.us.cray.com
172.30.74.208  cflls03  cflls03.us.cray.com
##LDAP Server Info
172.30.12.46   kingpin  kingpin.us.cray.com      kingpin.cray.com
172.30.12.48   kingfish kingfish.us.cray.com
kingfish.cray.com
##esLogin Info
172.30.48.62   kiyi     kiyi.us.cray.com        el-login0.us.cray.com
10.2.0.1       kiyi-eth1
##Networker server
#172.30.74.90  booboo   booboo.us.cray.com

10.3.1.1       smw
10.128.0.1     nid00000  c0-0c0s0n0             dvs-0
10.128.0.2     nid00001  c0-0c0s0n1             boot001 boot002
10.128.0.31    nid00030  c0-0c0s0n2             #old ddn6620_mds
10.128.0.32    nid00031  c0-0c0s0n3             hera-rsip2
10.128.0.3     nid00002  c0-0c0s1n0
10.128.0.4     nid00003  c0-0c0s1n1
10.128.0.29    nid00028  c0-0c0s1n2
10.128.0.30    nid00029  c0-0c0s1n3
10.128.0.5     nid00004  c0-0c0s2n0
10.128.0.6     nid00005  c0-0c0s2n1
10.128.0.27    nid00026  c0-0c0s2n2
10.128.0.28    nid00027  c0-0c0s2n3
10.128.0.7     nid00006  c0-0c0s3n0
10.128.0.8     nid00007  c0-0c0s3n1
10.128.0.25    nid00024  c0-0c0s3n2
10.128.0.26    nid00025  c0-0c0s3n3
10.128.0.9     nid00008  c0-0c0s4n0             login  login1  hera

```

10.128.0.10	nid00009	c0-0c0s4n1	sdb001 sdb002
10.128.0.23	nid00022	c0-0c0s4n2	hera-rsip hera-rsip1
10.128.0.24	nid00023	c0-0c0s4n3	mds nid00023_mds
...			

The client-system.ib File

The *client-system.ib* file contains client-system LNET router InfiniBand IP address to *cname* mapping information in a */etc/hosts* style format. The hostname field in this file is a fully-qualified interface specification of the form *hostname(ibn)*, where (*ib0*) is the assumed default if not specified. This file must be created by an administrator.

Sample *client-system.ib* file: *hera.ib*

```
#
# This is the /etc/hosts-like file for Infiniband IP addresses
# on "hera".
#
10.10.100.101    c0-0c2s2n0
10.10.100.102    c0-0c2s2n2
10.10.100.103    c1-0c0s7n0
10.10.100.104    c1-0c0s7n1
10.10.100.105    c1-0c0s7n2
10.10.100.106    c1-0c0s7n3
10.10.100.107    c3-0c1s0n0
10.10.100.108    c3-0c1s0n1
10.10.100.109    c3-0c1s0n2
10.10.100.110    c3-0c1s0n3
10.10.100.111    c3-0c1s5n0
10.10.100.112    c3-0c1s5n1
10.10.100.113    c3-0c1s5n2
10.10.100.114    c3-0c1s5n3
10.10.100.115    c3-0c2s4n0
10.10.100.116    c3-0c2s4n1
10.10.100.117    c3-0c2s4n2
10.10.100.118    c3-0c2s4n3
```

The cluster-name.ib File

The *cluster-name.ib* file contains Lustre server InfiniBand IP addresses to cluster (for example, Sonexion) host name mapping information in a */etc/hosts* style format. This file must be created by an administrator.

Sample *cluster-name.ib* file: *snx11029n.ib*

```
#
# This is the /etc/hosts-like file for Infiniband IP addresses
# on the Sonexion known as "snx11029n".
#
10.10.100.1     snx11029n000    #mgmnt
10.10.100.2     snx11029n001    #mgmnt
10.10.100.3     snx11029n002    #mgs
```

```

10.10.100.4    snx11029n003    #mds
10.10.100.5    snx11029n004    #first oss, oss0
10.10.100.6    snx11029n005
10.10.100.7    snx11029n006
10.10.100.8    snx11029n007
10.10.100.9    snx11029n008
10.10.100.10   snx11029n009
10.10.100.11   snx11029n010
10.10.100.12   snx11029n011
10.10.100.13   snx11029n012
10.10.100.14   snx11029n013
10.10.100.15   snx11029n014
10.10.100.16   snx11029n015    #last oss, oss11

```

The client-system.rtrIm File

The *client-system.rtrIm* file contains output from the `rtr -Im` command as executed from the SMW. When capturing the command output to a file, use the `-H` option to remove the header information from `rtr -Im` or open the file after capturing and delete the first two lines.

Follow this procedure to create the *client-system.rtrIm* file on the SMW:

1. Log on to the SMW.

```

crayadm@boot:~> ssh smw
Password:
Last login: Sun Feb 24 23:05:29 2013 from boot

```

2. Run the following command to capture the `rtr -Im` output (without header information) to a file:

```

crayadm@boot:~> rtr -Im -H > hera.rtrIm

```

3. Move the *hera.rtrIm* file to the working directory from which the `clcv` command will be run.

```

crayadm@boot:~> mv hera.rtrIm /path/to/working/dir/

```

Sample *client-system.rtrIm* file: *hera.rtrIm*

0	0	c0-0c0s0n0	c0-0c0s0g0	0	0	0
1	1	c0-0c0s0n1	c0-0c0s0g0	0	0	0
2	4	c0-0c0s1n0	c0-0c0s1g0	0	0	1
3	5	c0-0c0s1n1	c0-0c0s1g0	0	0	1
4	8	c0-0c0s2n0	c0-0c0s2g0	0	0	2
5	9	c0-0c0s2n1	c0-0c0s2g0	0	0	2
6	12	c0-0c0s3n0	c0-0c0s3g0	0	0	3
7	13	c0-0c0s3n1	c0-0c0s3g0	0	0	3
8	16	c0-0c0s4n0	c0-0c0s4g0	0	0	4
9	17	c0-0c0s4n1	c0-0c0s4g0	0	0	4
10	20	c0-0c0s5n0	c0-0c0s5g0	0	0	5
11	21	c0-0c0s5n1	c0-0c0s5g0	0	0	5
12	24	c0-0c0s6n0	c0-0c0s6g0	0	0	6
13	25	c0-0c0s6n1	c0-0c0s6g0	0	0	6
14	28	c0-0c0s7n0	c0-0c0s7g0	0	0	7
15	29	c0-0c0s7n1	c0-0c0s7g0	0	0	7

30	32	c0-0c0s0n2	c0-0c0s0g1	0	1	0
31	33	c0-0c0s0n3	c0-0c0s0g1	0	1	0
28	36	c0-0c0s1n2	c0-0c0s1g1	0	1	1
29	37	c0-0c0s1n3	c0-0c0s1g1	0	1	1
...						

Generate ip2nets and routes Information

When the prerequisite files have been created and gathered, the administrator can generate the persistent-storage file with the `clcv` generate action; this portable file will then be used to create `ip2nets` and `routes` directives for the servers, routers, and clients.

The following procedures frequently use the `--split-routes=4` flag, which will print information that can be loaded into `ip2nets` and `routes` files. This method of adding `modprobe.conf` directives is particularly valuable for large systems where the directives might otherwise exceed the `modprobe` buffer limit.

Create the persistent-storage File

1. Move all of the prerequisite files to an empty directory on the boot node or SMW (the `clcv` command is only available on the boot node or the SMW). The working directory should look similar to this when done:

```
crayadm@hera-smw:~/working_dir> ll
total 240
-rw-rw-r-- 1 crayadm crayadm 23707 Feb  8 14:27 hera.hosts
-rw-rw-r-- 1 crayadm crayadm  548 Feb  8 14:27 hera.ib
-rw-rw-r-- 1 crayadm crayadm 36960 Feb  8 14:27 hera.rtrIm
-rw-rw-r-- 1 crayadm crayadm  1077 Feb  8 14:27 info.snx11029
-rw-rw-r-- 1 crayadm crayadm   662 Feb  8 14:27 snx11029n.ib
```

2. Create the persistent-storage file.

```
crayadm@hera-smw:~/working_dir> clcv generate
```

The `clcv` command does not print to `stdout` with successful completion; however, if there are errors when running the command, set the `--debug` flag to add debugging information.

Create ip2nets and routes Information for the Compute Nodes

1. Execute the `clcv` command with the `compute` flag to generate directives for the compute nodes.

```
crayadm@hera-smw:~/working_dir> clcv compute --split-routes=4
# Place the following line(s) in the appropriate 'modprobe' file.
#~~~~~
options lnet ip2nets=/path/to/ip2nets-loading/filename
options lnet routes=/path/to/route-loading/filename
#~~~~~
# Place the following line(s) in the appropriate ip2nets-loading file.
#~~~~~
gni1 10.128.*.*
#~~~~~
```

Access and Manage SMW Services

Change the Default iDRAC Password

1. Log into the web interface as `root`.
2. Select **iDRAC settings** on the left-hand bar.
3. Select **network/Security** on the main top bar.
4. Select **Users** on the secondary top bar.
5. Select the user whose password is changing. For example, `userid 2` and username `root`.
6. Select **Configure User**, then **Next**.
7. Type the new password into the **New Password** and **Confirm New Password** text boxes.
8. Select **Apply** to complete the password change.

Configure the SMW to Synchronize to a Site NTP Server

The components of the Cray system synchronize time with the System Management Workstation (SMW) through Network Time Protocol (NTP). By default, the NTP configuration of the SMW is configured to stand alone; however, the SMW can optionally be configured to synchronize with a site NTP server. Follow this procedure to configure the SMW to synchronize to a site NTP server.

1. Stop the NTP server by issuing the `/etc/init.d/ntp stop` command; this command must be executed as user `root`:

```
smw:~ # /etc/init.d/ntp stop
```

2. Edit the `/etc/ntp.conf` file on the SMW to point to the new server.
3. Restart the NTP server by issuing the `/etc/init.d/ntp restart` command:

```
smw:~ # /etc/init.d/ntp start
```

The SMW can continue to update the rest of the system by proxy. By default, the SMW qualifies as a stratum 3 (local) NTP server. For more information about NTP, refer to the Linux documentation.

Enable an Integrated Dell Remote Access Controller (iDRAC6) on a rack-mount SMW

Prerequisites

To enable an iDRAC6 on a rack-mount SMW, you need:

- Physical access to the SMW console
- The iDRAC6 IP address, subnet mask, and default gateway
- The SMW root account password

TIP: The following command retrieves the iDRAC IP information:

```
smw:~> /usr/bin/ipmitool -I open lan print 1
```

1. If the SMW is running, `su` to `root` and shut it down.

```
crayadm@smw:~> su - root
smw:~ # shutdown -h now;exit
```

2. Connect Ethernet cable to the iDRAC6 port. The cable is located on back of a rack-mount SMW in the lower left corner.
3. Power up the SMW.
4. After the BIOS, Dell PowerEdge Expandable RAID Controller (PERC) card, and disk map have displayed, the IPv4/IPv6 information displays. When the IPv4/IPv6 information displays, press **Ctrl-E**.
5. Using the arrow keys, select **LAN Parameters**, then press **Enter**.
6. Select **NIC Selection** and set it to **Dedicated**. Then press **Esc**.
7. Using the arrow keys, scroll down and select the **IPv4 settings** section.
 - a. Ensure that IPv4 is enabled.
 - b. Confirm that the IPv4 address source is set to static:

```
IPv4 Address Source: Static
```

- c. Enter your iDRAC6 IP addresses for the following:
 - Address:
 - Subnet Mask:
 - Default Gateway:
- d. Ensure that IPv6 is disabled.

- e. Press **Esc** and return to the **LAN Parameters** window.
8. Using the arrow keys, select **LAN User Configuration**, then press **Enter**.
 - NOTE:** This configuration is for both SSH and web browser access to the iDRAC.
 - a. Enter the `root` account name and iDRAC password.

```
Account User name: root
Enter Password: *****
Confirm Password: *****
```
 - a. Press **Esc**.
 9. Press **Esc** again.
 10. Select **Save Changes and Exit**, then press **Enter**.

The SMW will complete booting up; no user interaction is required.

R815 SMW: Change the BIOS and iDRAC Settings for an iDRAC

Follow this procedure to change the BIOS and iDRAC settings for a Dell R815 SMW.

NOTE: The output and figures shown in this procedure are used as examples. Actual output and display may vary.

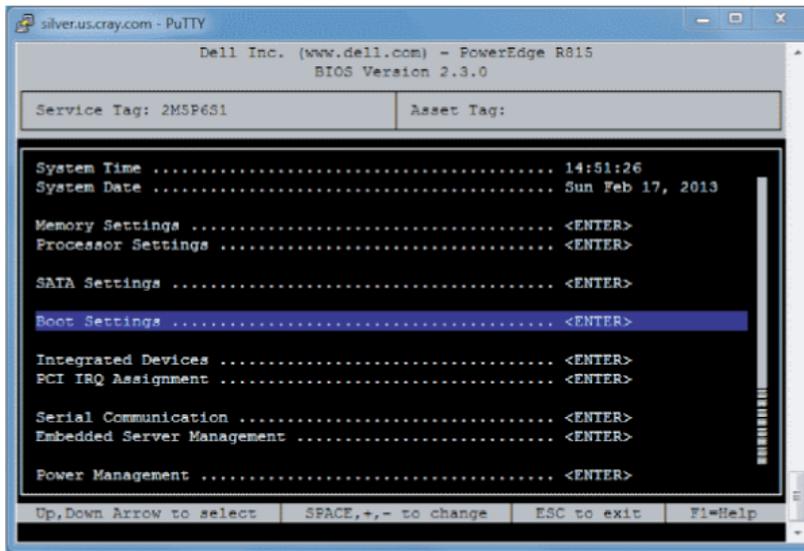
1. Power up the . When the BIOS power-on self-test (POST) process begins, **quickly press the F2 key** after the following messages appear in the upper-right of the screen.

```
F2 = System Setup
F10 = System Services
F11 = BIOS Boot Manager
F12 = PXE Boot
```

When the F2 keypress is recognized, the F2 = System Setup line changes to Entering System Setup.

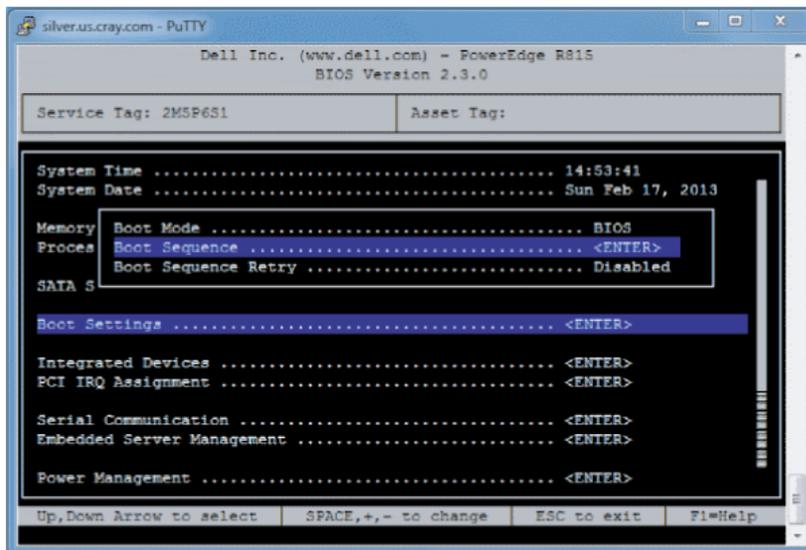
2. Select **Boot Settings**, then press **Enter**.

Figure 11. Dell R815 SMW Boot Settings Menu



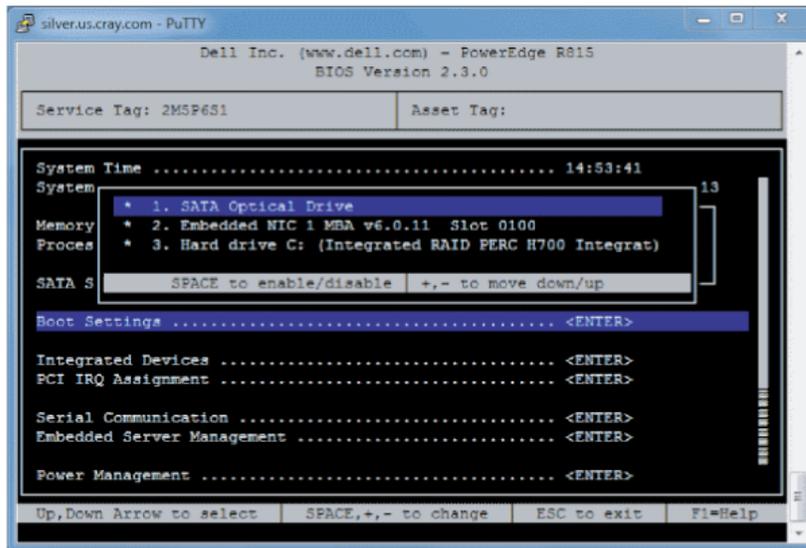
- a. Select **Boot Sequence**, then press `Enter` to view the boot settings.

Figure 12. Dell R815 SMW Boot Sequence Menu



- b. In the pop-up window, change the boot order so that the

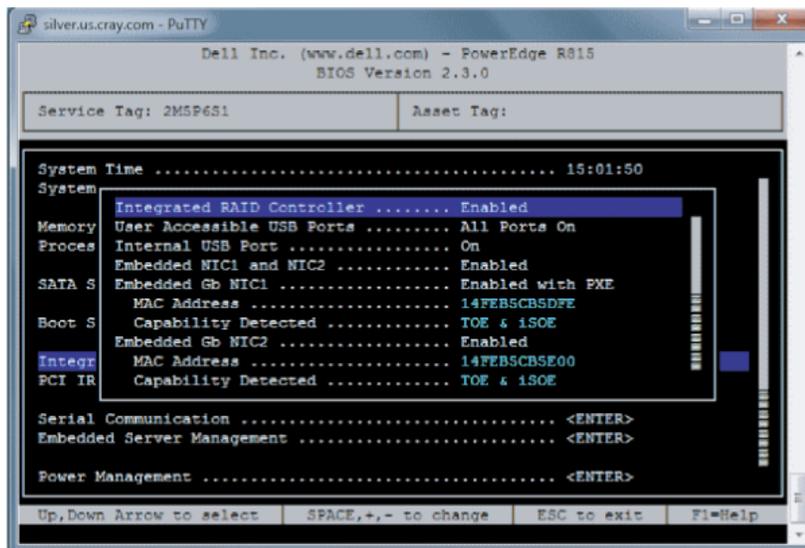
Figure 13. Dell R815 SMW Boot Sequence Settings



c. Press **Enter** to return to the **BIOS Boot Settings** screen.

3. Press **Esc** to return to the System Setup Menu, scroll down and select **Integrated Devices**.

Figure 14. Dell R815 SMW Integrated Devices (NIC) Settings



a. Set **Embedded Gb NIC 2** to **Enabled**.

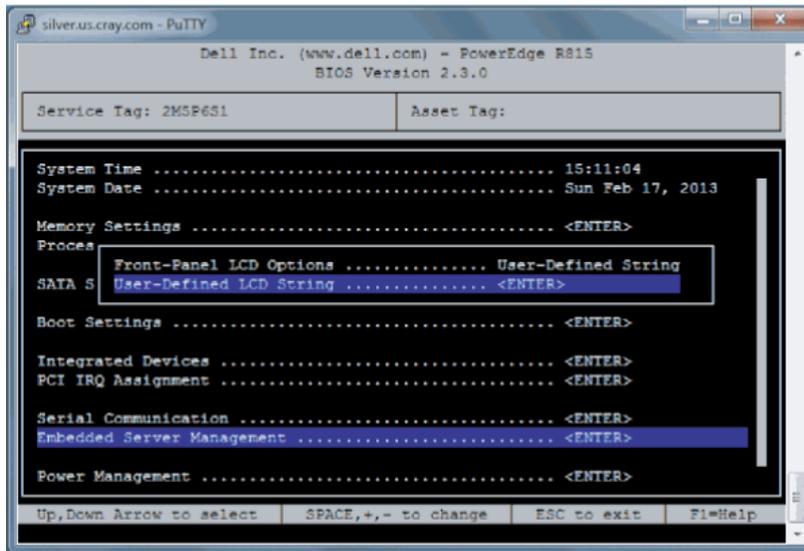
b. Scroll down and set **Embedded NIC 3** to **Enabled**.

c. Set **Embedded Gb NIC 4** to **Enabled**.

d. Press **Esc** to return to the System Settings Menu.

4. Select **Embedded Server Management**.

Figure 15. Dell R815 SMW Embedded Server Management Settings



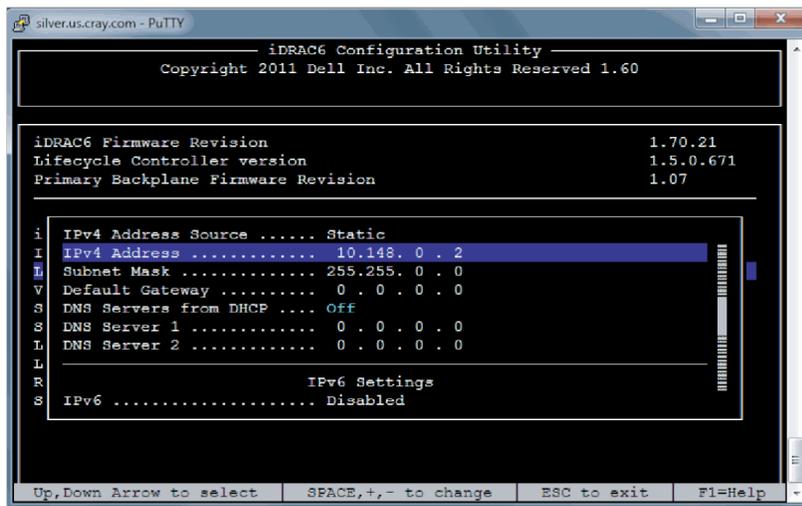
- a. Set **Front-Panel LCD Options** to **User-Defined LCD String**.
 - b. Set **User-Defined LCD String** to the login hostname, such as .
5. Save the changes and exit.
 - a. Press `Esc` to exit the System Setup Main Menu.
 - b. The utility displays the prompt "Are you sure you want to exit and reboot?" Select **Yes**.
 6. When the system reboots, press `Ctrl-E` to configure the iDRAC port settings.

www.dell.com

```
iDRAC6 Configuration Utility 1.60
Copyright 2011 Dell Inc. All Rights Reserved
Four 2.10 GHz Twelve-core Processors, L2/L3 Cache: 6 MB/10 MB
iDRAC6 FirmwareRevisionHversion: 1.70.21
.
.
.
IPv4 Stack      : Enabled
IP Address      : 10.148. 0 . 2
Subnet mask     : 255.255. 0 . 0
Default Gateway : 0 . 0 . 0 . 0
Press <Ctrl-E> for Remote Access Setup within 5 sec.....
```

- a. Set the **iDRAC LAN** to **ON**.
- b. Set **IPMI Over LAN** to **ON**.
- c. Select **LAN Parameters** and press `Enter`. Set the IPv4 address to the SMW DRAC IP address.
- d. Press `Esc` to return to the iDRAC menu, and `Esc` to exit and save.

Figure 16. Dell R815 SMW DRAC IPv4 Parameter Settings



R815 SMW: Enable an iDRAC

1. Connect Ethernet cable to the iDRAC port. The cable is located on back of the R815 SMW in the lower left corner.
2. Power up the SMW.
- 3.
4. Using the arrow keys, select **LAN Parameters**, then press `Enter`.
5. Select **NIC Selection** and set it to **Dedicated**. Then press `Esc`.
6. Using the arrow keys, scroll down and select the **IPv4 settings** section.
 - a. Ensure that IPv4 is enabled.
 - b. Confirm that the IPv4 address source is set to static:

IPv4 Address Source: Static

- c. Enter the iDRAC IP addresses for the following:
 - Address:
 - Subnet Mask:
 - Default Gateway:
 - d. Ensure that IPv6 is disabled.
 - e. Press `Esc` and return to the **LAN Parameters** window.
7. Using the arrow keys, select **LAN User Configuration**, then press `Enter`.

NOTE: This configuration is for both SSH and web browser access to the iDRAC.

- a. Enter the `root` account name and iDRAC password.

```
Account User name: root
Enter Password: *****
Confirm Password: *****
```

- b. Press `Esc`.
8. Press `Esc` again.
9. Select **Save Changes and Exit**, then press `Enter`. The SMW will complete booting up; no user interaction is required.

R630 SMW: Create an SMW Bootable Backup Drive

This procedure creates a bootable backup drive for a Dell R630 SMW in order to replace the primary drive if the primary drive fails. When these steps are completed, the backup drive on the SMW will be a bootable replacement for the primary drive when the backup drive is plugged in as or cabled as the primary drive.

IMPORTANT: The disk device names shown in this procedure are only examples. Substitute the actual disk device names for the actual system. The boot disk is `pci-0000:03:00.0-scsi-0:0:0:0` and is slot 0, and the bootable backup disk is `pci-0000:03:00.0-scsi-0:0:1:0` and is slot 1.

NOTICE: To create a clean backup, Cray recommends shutting down the Cray system before beginning this procedure.

Also, be aware that there may be a considerable load on the SMW while creating the SMW bootable backup drive.

1. Log on to the SMW as `crayadm` and `su` to `root`.

```
crayadm@smw> su -
Password:
smw#
```

2. Standardize the SMW's boot-time drive names with the Linux run-time drive names.

IMPORTANT: If the SMW configuration files on the SMW root drive have been modified already (because this site has completed this step at least once after installing the updated SMW base operating system), skip to step 3 on page 302; otherwise, complete this step to standardize the SMW's boot-time drive names with the Linux run-time drive names.

Set up ordered drives on the R630 SMW.

- a. Identify the installed SMW drive model numbers, serial numbers, and associated Linux device (`/dev`) names.

Execute `smwmapdrives` on the SMW to identify local (internal) drives mounted in the SMW and provide their Linux device (`/dev`) names.

NOTE: Effective with the SMW 7.2.UP00 release, the `smwmapdrives` script was provided both as a separate file in the release and in the base operating system RPM. If running that release or a later one, use the installed version of the script to back up the SMW.

```

smw# smwmapdrives
List of SMW-installed disk drives
-----
Physical slot 0:
  /dev/sda
  /dev/disk/by-id/scsi-35000c50079ab34b7
  /dev/disk/by-id/wwn-0x5000c50079ab34b7
  /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:0
Physical slot 1:
  /dev/sdb
  /dev/disk/by-id/scsi-35000c50079ab71c4
  /dev/disk/by-id/wwn-0x5000c50079ab71c4
  /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0
Physical slot 2:
  /dev/sdc
  /dev/disk/by-id/scsi-35000c50079ab313b
  /dev/disk/by-id/wwn-0x5000c50079ab313b
  /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:2:0
Physical slot 3:
  /dev/sdd
  /dev/disk/by-id/scsi-35000c50079ab4b4c
  /dev/disk/by-id/wwn-0x5000c50079ab4b4c
  /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:3:0
Physical slot 4:
  /dev/sde
  /dev/disk/by-id/scsi-35000c50079d05e70
  /dev/disk/by-id/wwn-0x5000c50079d05e70
  /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:4:0
Physical slot 5:
  NOT INSTALLED
Physical slot 6:
  NOT INSTALLED
Physical slot 7:
  NOT INSTALLED

```

The device names for `by-id` are persistent and will reference the drive, regardless of the slot in which the drive is installed.

`by-path` names reference a physical drive slot only and do not identify the drive installed in that slot. This is the naming used by default for the logging and database drives when the SMW was installed. This `by-path` name is used to specifically install logging and database file systems because the `by-id` device names refer to the physical drive slots expected to be used for those file systems and are provided as the default examples in the SMW installation configuration process.

The `/dev/sdX` drive names are not persistent; these names can change with each SMW boot and will change if drives are added, removed, or reordered in the SMW slots. For this reason, the `/dev/sda` drive name can only be used for the desk-side SMW.

Choose either the `by-id` naming or the `by-path` naming as the site administrative policy for managing the SMW-install disk drives. The following documentation provides the steps necessary to implement this selection on the SMW prior to creating an SMW bootable backup drive.

- b. Back up the following files before proceeding:

```

smw# cp -p /boot/grub/device.map /boot/grub/device.map-YYYYMMDD
smw# cp -p /boot/grub/menu.lst /boot/grub/menu.lst-YYYYMMDD
smw# cp -p /etc/fstab /etc/fstab-YYYYMMDD

```

Cray recommends that `/boot/grub/device.map`, `/etc/fstab` and `/boot/grub/menu.lst` changes use the "by-path" rather than the "by-id" device name because that would allow physically swapping the backup drive into the primary slot when there is a disk failure in the primary disk. If the backup disk is intended as backup only, rather than as a bootable backup, it is acceptable to use either device name.

- c. Edit the grub `device.map` file to reflect physical drive locations.

To provide a direct mapping of the SMW disk drive physical slots to the boot loader (BIOS and `grub`) drive names, the `device.map` mapping file used by `grub` should be replaced. Perform the following steps to install new `device.map` file entries to effect this mapping.

1. Edit the grub `device.map` file.
2. Delete all lines.
3. Enter the following lines into the file. These lines show each drive slot's physical location mapped to its boot-time `hd?` name.

NOTE: `by-id` names should not be used in the `device.map` file.

```
# Dell Rackmount r630 SMW
# grub(8) device mapping for boot-drive identification
# hd? numbers are being mapped to their physical
(hd0)  /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:0
(hd1)  /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0
(hd2)  /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:2:0
(hd3)  /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:3:0
(hd4)  /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:4:0
(hd5)  /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:5:0
```

- d. Modify the SMW boot drive `/etc/fstab` file to use `by-id` or `by-path` naming.

Modify the SMW file system mounting configuration file to use SMW disk `by-id` or `by-path` naming. Complete this step to replace any `/dev/sdX` disk partition references.

NOTE: Use the output of the `smwmapdrives` script in step 2.a on page 299 as a reference for drive names.

Edit `/etc/fstab`, replacing drive `/dev/sdX` references with either the `by-id` or `by-path` name's corresponding device name.

When a reference to `/dev/sda1` is being replaced, replace it with the corresponding "partition" file system suffixed with `-part1`. File system partitions for `/dev/sda` are indicated by the numeral appended to the device name; for example, `/dev/sda1` refers to partition 1 on `/dev/sda`.

For example, if the root and swap file systems are currently configured to mount `/dev/sda2`, they should be changed. Using the `by-path` device name from the example in step 2.a on page 307, the `fstab` lines would change from:

```
/dev/sda1 swap          swap          defaults      0 0
/dev/sda2 /                    ext3          acl,user_xattr 1 1
```

to:

```
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:0-part1 swap swap
defaults          0 0
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:0-part2 /      ext3
acl,user_xattr    1 1
```

- e. Modify `/boot/grub/menu.lst` to reflect the `device.map` BIOS/boot-up drive changes for the `sdX` remapping.

The same device name replacement performed on `/etc/fstab` should also be performed on the grub bootloader `/boot/grub/menu.lst` configuration file. All references to `/dev/sdX` devices should be replaced with corresponding `by-path` device names.

- f. Invoke the grub utility to reinstall the SMW boot loader on the primary boot drive.

Once the changes to `device.map`, `fstab`, and `menu.lst` have been completed, the grub bootloader boot blocks must be updated to reflect changes to the device names. Complete this step to update the boot loader on the boot drive.

Invoke the grub utility and reinstall SMW root-drive boot blocks.

```
smw# grub --no-curses
GNU GRUB version 0.97 (640K lower / 3072K upper memory)
[ Minimal BASH-like line editing is supported. For the first word, TAB
  lists possible command completions. Anywhere else TAB lists the possible
  completions of a device/filename. ]
grub> root (hd0,1)
      root (hd0,1)
      Filesystem type is ext2fs, partition type 0x83
grub> setup (hd0)
      Checking if "/boot/grub/stage1" exists... yes
      Checking if "/boot/grub/stage2" exists... yes
      Checking if "/boot/grub/e2fs_stage1_5" exists... yes
      Running "embed /boot/grub/e2fs_stage1_5 (hd0)"... 17 sectors \
are embedded. Succeeded
      Running "install /boot/grub/stage1 (hd0) (hd0)1+17 p (hd0,1)/boot/grub/
stage2 \
/boot/grub/menu.lst"... succeeded
      Done.
      grub> quit
```

3. If the backup drive disk partition table already exists and the partition table on the backup drive matches the partition table that is on the primary boot drive, skip this step; otherwise, create the backup drive disk partition table.

In this example, the partition table consists of two slices. Slice 1 is a 4 GB Linux swap partition. Slice 2 is the balance of disk space used for the root file system.

- a. Use the `fdisk` command to display the boot disk partition layout.

```
smw# fdisk -lu /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:0
Disk /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:0: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000c3cc8
```

Blocks	Id	System	Device	Boot	Start	End
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:0-part1					2048	67102719
33550336	82	Linux swap / Solaris				
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:0-part2				*	67102720	976773119
454835200	83	Linux				

- b. Use the `fdisk` command to configure the bootable backup disk partition layout. Set the bootable backup disk partition layout to match the boot disk partition layout. First, clear all of the old partitions using the `d` command within `fdisk`; next create a Linux swap and a Linux partition; and then write the changes to the disk. For help, type `m` within `fdisk`.

```

smw# fdisk -u /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0

Command (m for help): d
Partition number (1-4): 2

Command (m for help): d
Selected partition 1

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First sector (63-312581807, default 63): (Press the Enter key)
Using default value 63
Last sector, +sectors or +size{K,M,G} (63-312581807, default 312581807): 16771859
Partition number (1-4, default 1): 1
First sector (2048-976773167, default 2048): (Press the Enter key)
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-976773167, default 976773167): 67102719

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap / Solaris)

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4, default 2): 2
First sector (67102720-976773167, default 67102720): (Press the Enter key)
Using default value 67102720
Last sector, +sectors or +size{K,M,G} (67102720-976773167, default 976773167): (Press
the Enter key)
Using default value 976773167

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

```

- c. Display the boot backup disk partition layout and confirm it matches the `pci-0000:03:00.0-scsi-0:0:0:0` sector information.

```

smw# fdisk -lu /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0

Disk /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x7c334e96


```

Blocks	Id	System	Device	Boot	Start	End
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0-part1					2048	67102719
33550336	82	Linux swap / Solaris				
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0-part2					67102720	976773167
454835224	83	Linux				

4. Initialize the `swap` device.

```
smw# mkswap /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0-part1
Setting up swapspace version 1, size = 33550332 KiB
no label, UUID=8391498b-d159-469c-b766-66f00a28ff74
```

5. Create a new file system on the backup drive root partition by executing the mkfs command.

```
smw# mkfs -t ext3 /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0-part2
mke2fs 1.41.9 (22-Aug-2009)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
28434432 inodes, 113708806 blocks
5685440 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
3471 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872, 71663616, 78675968,
    102400000

Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 33 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
```

6. Mount the new backup root file system on /mnt.

```
smw# mount /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0-part2 /mnt
```

7. Confirm that the backup root file system is mounted.

```
smw# df
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/sda2       447696736 9180648 437606420 3% /
udev            66029308      744 66028564 1% /dev
tmpfs           66029308    39540 65989768 1% /dev/shm
/dev/sdae       309637120 1107516 292800964 1% /var/opt/cray/disk/1
/dev/sdac       206424760 1963664 193975336 2% /home
/dev/sdad       154818540 474696 146479524 1% /var/lib/mysql
/dev/drbd_r0    961405840 247180 912322076 1% /var/lib/pgsql
/dev/sdb2       447696760 202940 424752060 1% /mnt
```

The running root file system device is the one mounted on /.

8. Dump the running root file system to the backup drive.

```
smw# cd /mnt
smw# dump 0f - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:0-part2 | restore rf -
DUMP: WARNING: no file `/etc/dumpdates'
DUMP: Date of this level 0 dump: Wed Sep 16 15:40:41 2015
DUMP: Dumping /dev/sda2 (/) to standard output
DUMP: Label: none
DUMP: Writing 10 Kilobyte records
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 9129804 blocks.
DUMP: Volume 1 started with block 1 at: Wed Sep 16 15:43:08 2015
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
```

```

restore: ./lost+found: File exists
./tmp/rstmdir1442436041: (inode 27254928) not found on tape
./tmp/rstmodel442436041: (inode 27254931) not found on tape
DUMP: 77.64% done at 23626 kB/s, finished in 0:01
DUMP: Volume 1 completed at: Wed Sep 16 15:50:09 2015
DUMP: Volume 1 9132800 blocks (8918.75MB)
DUMP: Volume 1 took 0:07:01
DUMP: Volume 1 transfer rate: 21693 kB/s
DUMP: 9132800 blocks (8918.75MB)
DUMP: finished in 421 seconds, throughput 21693 kBytes/sec
DUMP: Date of this level 0 dump: Wed Sep 16 15:40:41 2015
DUMP: Date this dump completed: Wed Sep 16 15:50:09 2015
DUMP: Average transfer rate: 21693 kB/s
DUMP: DUMP IS DONE

```

9. Modify the backup drive's `fstab` and `menu.lst` files to reflect the backup drive's device, replacing the primary drive's device name.

NOTE: This step is necessary only if `by-id` names are used. If `by-path` names are being utilized for the `root` and `swap` devices, changes are not necessary; these devices reference physical slots, and the backup drive will be moved to the same physical slot (slot 0) when replacing a failed primary boot drive.

- a. Edit `/mnt/etc/fstab`. Replace the `root` and `swap` partitions' `by-id` device names with those used for this backup device, replacing the original disk device name.

```
smw# Vi /mnt/etc/fstab
```

For example, change

```

/dev/disk/by-id/scsi-35000c50079ab34b7-part1 swap      swap
defaults                                0 0
/dev/disk/by-id/scsi-35000c50079ab34b7-part2 /          ext3
acl,user_xattr                          1 1

```

to:

```

/dev/disk/by-id/scsi-35000c50079ab71c4-part1 swap      swap
defaults                                0 0
/dev/disk/by-id/scsi-35000c50079ab71c4-part2 /          ext3
acl,user_xattr                          1 1

```

- b. Edit `/mnt/boot/grub/menu.lst`. Replace the `root=` and `resume=` device names with those used for this backup device, replacing the original disk device name.

The `root=` entry normally refers to partition `-part2`, and the `resume=` entry normally refers to partition `-part1`; these partition references must be maintained.

For example, replace the `menu.lst` configuration references of:

```
root=/dev/disk/by-id/scsi-35000c50079ab34b7-part2
```

with:

```
root=/dev/disk/by-id/scsi-35000c50079ab71c4-part2
```

or similarly with the `by-id` device names, if those are preferred.

Replace the `resume=` references similarly.

10. Install the `grub` boot loader. To make the backup drive bootable, reinstall the `grub` boot facility on that drive.



CAUTION: Although all of the disks connected to the SMW are available to the system, `grub` detects only the first 16 devices. Therefore, if a disk is added to the SMW after the SMW is connected to the boot RAID, it is advisable to reboot the SMW before continuing.

a. Create a unique file on the backup drive to be used to identify that drive to `grub` boot facility.

```
smw# cd /
smw# touch /mnt/THIS_IS_1
```

b. Invoke the `grub` boot utility. Within the `grub` boot utility:

1. Execute the `find` command to locate the drive designation that `grub` uses.
2. Select the drive to which the boot blocks will be installed with the `root` command.
3. Use the `setup` command to set up and install the `grub` boot blocks on that drive. The Linux `grub` utility and boot system *always* refer to drives as `hd`, regardless of the actual type of drives. For example:

```
smw# grub --no-curses
GNU GRUB version 0.97 (640K lower / 3072K upper memory)
[ Minimal BASH-like line editing is supported. For the first word, TAB
  lists possible command completions. Anywhere else TAB lists the
  possible
  completions of a device/filename. ]
grub> find /THIS_IS_1
find /THIS_IS_1
  (hd1,1)
grub> root (hd1,1)
root (hd1,1)
  Filesystem type is ext2fs, partition type 0x83
grub> setup (hd1)
setup (hd1)
  Checking if "/boot/grub/stage1" exists... yes
  Checking if "/boot/grub/stage2" exists... yes
  Checking if "/boot/grub/e2fs_stage1_5" exists... yes
  Running "embed /boot/grub/e2fs_stage1_5 (hd1)"... 17 sectors are
  embedded.
  succeeded
  Running "install /boot/grub/stage1 (hd1) (hd1)1+17 p (hd1,1)/boot/grub/
  stage2 /boot/grub/menu.lst"... succeeded
  Done.
grub> quit
quit
```

IMPORTANT: For R630 SMWs, `grub` recreates `device.map` with the short names, not the persistent names. Do not trust the `/dev/sdx` names. Always use `find` when executing `grub` because it is possible that `grub root` may not be `hd2` the next time `grub` is executed.

11. Unmount the backup root partition.

```
smw# umount /mnt
```

The drive is now bootable once plugged in or cabled as the primary drive.

R815 SMW: Create an SMW Bootable Backup Drive

This procedure creates a bootable backup drive for a Dell R815 SMW in order to replace the primary drive if the primary drive fails. When these steps are completed, the backup drive on the SMW will be a bootable replacement for the primary drive when the backup drive is plugged in as or cabled as the primary drive.

IMPORTANT: The disk device names shown in this procedure are only examples. Substitute the actual disk device names for the actual system. The boot disk is phy7 and is slot 0, and the bootable backup disk is phy6 and is slot 1.

NOTICE: To create a clean backup, Cray recommends shutting down the Cray system before beginning this procedure.

Also, be aware that there may be a considerable load on the SMW while creating the SMW bootable backup drive.

1. Log on to the SMW as `crayadm` and `su` to `root`.

```
crayadm@smw> su -
Password:
smw#
```

2. Standardize the SMW's boot-time drive names with the Linux run-time drive names.

IMPORTANT: If the SMW configuration files on the SMW root drive have been modified already (because this site has completed this step at least once after installing the updated SMW base operating system), skip to step 3 on page 310; otherwise, complete this step to standardize the SMW's boot-time drive names with the Linux run-time drive names.

Set up ordered drives on the R815 SMW.

- a. Identify the installed SMW drive model numbers, serial numbers, and associated Linux device (`/dev`) names.

Execute `smwmapdrives` on the SMW to identify local (internal) drives mounted in the SMW and provide their Linux device (`/dev`) names.

NOTE: Effective with the SMW 7.2.UP00 release, the `smwmapdrives` script was provided both as a separate file in the release and in the base operating system RPM. If running that release or a later one, use the installed version of the script to back up the SMW.

```
smw# smwmapdrives
List of SMW-installed disk drives
-----
Physical slot 0:
/dev/sda
/dev/disk/by-id/ata-FUJITSU_MHZ2160BK_G2_K85DTB227RDS
/dev/disk/by-id/scsi-SATA_FUJITSU_MHZ2160_K85DTB227RDS
/dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0
Physical slot 1:
/dev/sdc
/dev/disk/by-id/ata-FUJITSU_MHZ2160BK_G2_K85DTB227RD7
/dev/disk/by-id/scsi-SATA_FUJITSU_MHZ2160_K85DTB227RF3
/dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0
Physical slot 2:
/dev/sdd
/dev/disk/by-id/ata-FUJITSU_MHZ2160BK_G2_K85DTB227RF3
/dev/disk/by-id/scsi-SATA_FUJITSU_MHZ2160_K85DTB227RF3
/dev/disk/by-path/pci-0000:05:00.0-sas-phy5-0x4433221105000000-lun-0
Physical slot 3:
/dev/sdb
```

```

/dev/disk/by-id/ata-ST9500620NS_9XF0665V
/dev/disk/by-id/scsi-SATA_ST9500620NS_9XF0665V
/dev/disk/by-path/pci-0000:05:00.0-sas-phy4-0x4433221104000000-lun-0
Physical slot 4:
    NOT INSTALLED
Physical slot 5:
    NOT INSTALLED

```

The device names for `by-id` are persistent and will reference the drive, regardless of the slot in which the drive is installed.

`by-path` names reference a physical drive slot only and do not identify the drive installed in that slot. This is the naming used by default for the logging and database drives when the SMW was installed. This `by-path` name is used to specifically install logging and database file systems because the `by-id` device names refer to the physical drive slots expected to be used for those file systems and are provided as the default examples in the SMW installation configuration process.

The `/dev/sdX` drive names are not persistent; these names can change with each SMW boot and will change if drives are added, removed, or reordered in the SMW slots. For this reason, the `/dev/sda` drive name can only be used for the desk-side SMW.

Choose either the `by-id` naming or the `by-path` naming as the site administrative policy for managing the SMW-install disk drives. The following documentation provides the steps necessary to implement this selection on the SMW prior to creating an SMW bootable backup drive.

- b. Back up the following files before proceeding:

```

smw# cp -p /boot/grub/device.map /boot/grub/device.map-YYYYMMDD
smw# cp -p /boot/grub/menu.lst /boot/grub/menu.lst-YYYYMMDD
smw# cp -p /etc/fstab /etc/fstab-YYYYMMDD

```

Cray recommends that `/boot/grub/device.map`, `/etc/fstab` and `/boot/grub/menu.lst` changes use the "by-path" rather than the "by-id" device name because that would allow physically swapping the backup drive into the primary slot when there is a disk failure in the primary disk. If the backup disk is intended as backup only, rather than as a bootable backup, it is acceptable to use either device name.

- c. Edit the grub `device.map` file to reflect physical drive locations.

To provide a direct mapping of the SMW disk drive physical slots to the boot loader (BIOS and grub) drive names, the `device.map` mapping file used by grub should be replaced. Perform the following steps to install new `device.map` file entries to effect this mapping.

1. Edit the grub `device.map` file.
2. Delete all lines.
3. Enter the following lines into the file. These lines show each drive slot's physical location mapped to its boot-time `hd?` name. Note that `by-id` names should not be used in the `device.map` file.

```

# Dell Rackmount r815 SMW
# grub(8) device mapping for boot-drive identification
# hd? numbers are being mapped to their physical
(hd0) /dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-
lun-0
(hd1) /dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-
lun-0
(hd2) /dev/disk/by-path/pci-0000:05:00.0-sas-phy5-0x4433221105000000-
lun-0
(hd3) /dev/disk/by-path/pci-0000:05:00.0-sas-phy4-0x4433221104000000-
lun-0
(hd4) /dev/disk/by-path/pci-0000:05:00.0-sas-phy3-0x4433221103000000-

```

```
lun-0
(hd5) /dev/disk/by-path/pci-0000:05:00.0-sas-phy2-0x4433221102000000-
lun-0
```

- d. Modify the SMW boot drive `/etc/fstab` file to use `by-id` or `by-path` naming.

Modify the SMW file system mounting configuration file to use SMW disk `by-id` or `by-path` naming. Complete this step to replace any `/dev/sdX` disk partition references.

NOTE: Use the output of the `smwmapdrives` script in step 2.a on page 307 as a reference for drive names.

Edit `/etc/fstab`, replacing drive `/dev/sdX` references with either the `by-id` or `by-path` name's corresponding device name.

When a reference to `/dev/sda1` is being replaced, replace it with the corresponding "partition" file system suffixed with `-part1`. File system partitions for `/dev/sda` are indicated by the numeral appended to the device name; for example, `/dev/sda1` refers to partition 1 on `/dev/sda`.

For example, if the root and swap file systems are currently configured to mount `/dev/sda2`, they should be changed. Using the `by-path` device name from the example in step 2.a on page 307, the `fstab` lines would change from:

```
/dev/sda1 swap                swap        defaults    0 0
/dev/sda2 /                        ext3        acl,user_xattr 1 1
```

to:

```
/dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0-
part1 swap swap        defaults    0 0
/dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0-
part2 /                        ext3        acl,user_xattr 1 1
```

- e. Modify `/boot/grub/menu.lst` to reflect the `device.map` BIOS/boot-up drive changes for the `sdX` remapping.

The same device name replacement performed on `/etc/fstab` should also be performed on the grub bootloader `/boot/grub/menu.lst` configuration file. All references to `/dev/sdX` devices should be replaced with corresponding `by-path` device names.

- f. Invoke the `grub` utility to reinstall the SMW boot loader on the primary boot drive.

Once the changes to `device.map`, `fstab`, and `menu.lst` have been completed, the grub bootloader boot blocks must be updated to reflect changes to the device names. Complete this step to update the boot loader on the boot drive.

Invoke the `grub` utility and reinstall SMW root-drive boot blocks.

```
smw# grub --no-curses
GNU GRUB version 0.97 (640K lower / 3072K upper memory)
[ Minimal BASH-like line editing is supported. For the first word, TAB
  lists possible command completions. Anywhere else TAB lists the possible
  completions of a device/filename. ]
grub> root (hd0,1)
      root (hd0,1)
      Filesystem type is ext2fs, partition type 0x83
grub> setup (hd0)
```

```

Checking if "/boot/grub/stage1" exists... yes
Checking if "/boot/grub/stage2" exists... yes
Checking if "/boot/grub/e2fs_stage1_5" exists... yes
Running "embed /boot/grub/e2fs_stage1_5 (hd0)"... 17 sectors \
are embedded. Succeeded
Running "install /boot/grub/stage1 (hd0) (hd0)1+17 p (hd0,1)/boot/grub/
stage2 \
/boot/grub/menu.lst"... succeeded
Done.
grub> quit

```

3. If the backup drive disk partition table already exists and the partition table on the backup drive matches the partition table that is on the primary boot drive, skip this step; otherwise, create the backup drive disk partition table.

In this example, the partition table consists of two slices. Slice 1 is a 4 GB Linux swap partition. Slice 2 is the balance of disk space used for the root file system.

- a. Use the `fdisk` command to display the boot disk partition layout.

```

smw# fdisk -lu /dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0
Disk /dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0: 250.0 GB, \
268435456000 bytes
255 heads, 63 sectors/track, 19457 cylinders, total 312581808 sectors
Units = sectors of 1 * 512 = 512 bytes
Disk identifier: 0x00000082

Device
/dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0-part1 \
Boot Start End Blocks Id System
63 16771859 8385898+ 82 Linux swap / Solaris
/dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0-part2 \
Boot Start End Blocks Id System
* 16771860 312576704 147902422+ 83 Linux

```

- b. Use the `fdisk` command to configure the bootable backup disk partition layout. Set the bootable backup disk partition layout to match the boot disk partition layout. First, clear all of the old partitions using the `d` command within `fdisk`; next create a Linux swap and a Linux partition; and then write the changes to the disk. For help, type `m` within `fdisk`.

```

smw# fdisk -u /dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0

The number of cylinders for this disk is set to 19457.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): p

Disk /dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0: 250.0 GB, \
268435456000 bytes
255 heads, 63 sectors/track, 19457 cylinders, total 312581808 sectors
Units = sectors of 1 * 512 = 512 bytes
Disk identifier: 0x00000080

Device
/dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0-part1 \
Boot Start End Blocks Id System
63 16771859 83828 82 Linux
swap / Solaris
Partition 1 does not end on cylinder boundary.

```

```

/dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0-part2 \
      Boot   Start       End     Blocks  Id System
      167719 312581807 156207044+ 83 Linux

```

```

Command (m for help): d
Partition number (1-4): 2

```

```

Command (m for help): d
Selected partition 1

```

```

Command (m for help): n
Command action
  e extended
  p primary partition (1-4)

```

```

p
Partition number (1-4): 1
First sector (63-312581807, default 63): (Press the Enter key)
Using default value 63
Last sector, +sectors or +size{K,M,G} (63-312581807, default 312581807): 16771859
Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap / Solaris)

```

```

Command (m for help): n
Command action
  e extended
  p primary partition (1-4)

```

```

p
Partition number (1-4): 2
First sector (16771860-312581807, default 16771860): (Press the Enter key)
Using default value 16771860
Last sector, +sectors or +size{K,M,G} (16771860-312581807, default 312581807): (Press
the Enter key)
Using default value 312581807

```

```

Command (m for help): w
The partition table has been altered!

```

```

Calling ioctl() to re-read partition table.
Syncing disks.

```

- c. Display the boot backup disk partition layout and confirm it matches the `phy7` sector information.

```

smw# fdisk -lu /dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0
Disk /dev/disk/by-path/pci-0000:05:00.0-sas-phy6:1-0x4433221106000000:0-lun0: 250.0
GB, \
268435456000 bytes
255 heads, 63 sectors/track, 19457 cylinders, total 312581808 sectors

```

4. Initialize the `swap` device.

```

smw# mkswap /dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0-part1
mkswap: /dev/disk/by-path/pci-0000:05:00.0-sas-phy6:1-0x4433221106000000:0-lun0-part1:
warning: don't erase bootbits sectors
        (DOS partition table detected). Use -f to force.
Setting up swapspace version 1, size = 8385892 KiB
no label, UUID=c0ef22ac-b405-4236-855b-e4a09b6e94ed

```

5. Create a new file system on the backup drive `root` partition by executing the `mkfs` command.

```

smw# mkfs -t ext3 /dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0-
part2
mke2fs 1.41.9 (22-Aug-2009)
Filesystem label=
OS type: Linux

```

```

Block size=4096 (log=2)
Fragment size=4096 (log=2)
9248768 inodes, 36976243 blocks
1848812 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
1129 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872

Writing inode tables:   done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 37 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.

```

- Mount the new backup root file system on /mnt.

```

smw# mount \
/dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0-part2 /mnt

```

- Confirm that the backup root file system is mounted.

```

smw# df
Filesystem          1K-blocks      Used Available Use% Mounted on
/dev/sda2            303528624    6438700 281671544   3% /
udev                 1030332         116  1030216   1% /dev
/dev/sdb2            306128812    195568 290505224   1% /mnt

```

The running root file system device is the one mounted on /.

- Dump the running root file system to the backup drive.

```

smw# cd /mnt
smw# dump 0f - \
/dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0-part2 | restore rf -
DUMP: WARNING: no file `/etc/dumpdates'
DUMP: Date of this level 0 dump: Tue Mar 15 13:43:17 2011
DUMP: Dumping /dev/sda2 (/) to standard output
DUMP: Label: none
DUMP: Writing 10 Kilobyte records
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 7898711 blocks.
DUMP: Volume 1 started with block 1 at: Tue Mar 15 13:44:40 2011
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
restore: ./lost+found: File exists
DUMP: 79.34% done at 20890 kB/s, finished in 0:01
DUMP: Volume 1 completed at: Tue Mar 15 13:52:13 2011
DUMP: Volume 1 7908080 blocks (7722.73MB)
DUMP: Volume 1 took 0:07:33
DUMP: Volume 1 transfer rate: 17457 kB/s
DUMP: 7908080 blocks (7722.73MB)
DUMP: finished in 453 seconds, throughput 17457 kBytes/sec
DUMP: Date of this level 0 dump: Tue Mar 15 13:43:17 2011
DUMP: Date this dump completed: Tue Mar 15 13:52:13 2011
DUMP: Average transfer rate: 17457 kB/s
DUMP: DUMP IS DONE

```

9. Modify the backup drive's `fstab` and `menu.lst` files to reflect the backup drive's device, replacing the primary drive's device name.

NOTE: This step is necessary only if `by-id` names are used. If `by-path` names are being utilized for the `root` and `swap` devices, changes are not necessary; these devices reference physical slots, and the backup drive will be moved to the same physical slot (slot 0) when replacing a failed primary boot drive.

- a. Edit `/mnt/etc/fstab`. Replace the `root` and `swap` partitions' `by-id` device names with those used for this backup device, replacing the original disk device name.
For example, change

```
/dev/disk/by-id/ata-FUJITSU_MHZ2160BK_G2_K85DTB227RDS-part1 swap swap defaults
/dev/disk/by-id/ata-FUJITSU_MHZ2160BK_G2_K85DTB227RDS-part2 / ext3 acl,user_xattr
```

to:

```
/dev/disk/by-id/ata-FUJITSU_MHZ2160BK_G2_K85DTB227RD7-part1 swap swap defaults
/dev/disk/by-id/ata-FUJITSU_MHZ2160BK_G2_K85DTB227RD7-part2 / ext3 acl,user_xattr
```

- b. Edit `/mnt/boot/grub/menu.lst`. Replace the `root=` and `resume=` device names with those used for this backup device, replacing the original disk device name.

The `root=` entry normally refers to partition `-part2`, and the `resume=` entry normally refers to partition `-part1`; these partition references must be maintained.

For example, replace the `menu.lst` configuration references of:

```
root=/dev/disk/by-id/ata-FUJITSU_MHZ2160BK_G2_K85DTB227RDS-part2
```

with:

```
root=/dev/disk/by-id/ata-FUJITSU_MHZ2160BK_G2_K85DTB227RD7-part2
```

or similarly with the `by-id` device names, if those are preferred.

Replace the `resume=` references similarly.

10. Install the `grub` boot loader. To make the backup drive bootable, reinstall the `grub` boot facility on that drive.



CAUTION: Although all of the disks connected to the SMW are available to the system, `grub` detects only the first 16 devices. Therefore, if a disk is added to the SMW after the SMW is connected to the boot RAID, it is advisable to reboot the SMW before continuing.

- a. Create a unique file on the backup drive to be used to identify that drive to `grub` boot facility.

```
smw# cd /
smw# touch /mnt/THIS_IS_6
```

- b. Invoke the `grub` boot utility. Within the `grub` boot utility:

1. Execute the `find` command to locate the drive designation that `grub` uses.
2. Select the drive to which the boot blocks will be installed with the `root` command.
3. Use the `setup` command to set up and install the `grub` boot blocks on that drive. The Linux `grub` utility and boot system *always* refer to drives as `hd`, regardless of the actual type of drives. For example:

```

smw# grub --no-curses
GNU GRUB version 0.97 (640K lower / 3072K upper memory)
[ Minimal BASH-like line editing is supported. For the first word, TAB
lists possible command completions. Anywhere else TAB lists the
possible
completions of a device/filename. ]
grub> find /THIS_IS_6
(hd2,1)
grub> root (hd2,1)
root (hd2,1)
Filesystem type is ext2fs, partition type 0x83
grub> setup (hd2)
Checking if "/boot/grub/stage1" exists... yes
Checking if "/boot/grub/stage2" exists... yes
Checking if "/boot/grub/e2fs_stage1_5" exists... yes
Running "embed /boot/grub/e2fs_stage1_5 (hd2)"... 17 sectors are
embedded.
succeeded
Running "install /boot/grub/stage1 (hd2) (hd2)1+17 p (hd2,1)/boot/grub/
stage2 \
/boot/grub/menu.lst"... succeeded
Done.
grub> quit

```

IMPORTANT: For R815 SMWs, grub recreates device.map with the short names, not the persistent names. Do not trust the /dev/sdx names. Always use find when executing grub because it is possible that grub root may not be hd2 the next time grub is executed.

11. Unmount the backup root partition.

```
smw# umount /mnt
```

The drive is now bootable once plugged in or cabled as the primary drive.

Desk-side SMW: Create an SMW Bootable Backup Drive

This procedure creates a System Management Workstation (SMW) bootable backup drive for a desk-side SMW. Its purpose is to replace the primary drive if the primary drive fails. When this procedure is completed, the backup drive on the SMW will be a bootable replacement for the primary drive when the backup drive is plugged in as or cabled as the primary drive.

IMPORTANT: The disk device names shown in this procedure are only examples; substitute the actual disk device names when executing this procedure. For example, on an SMW with three SMW disks, the boot disk is /dev/sda and the bootable backup disk is /dev/sdc; on an SMW with two SMW disks, the boot disk is /dev/sda and the bootable backup disk is /dev/sdb.

NOTICE: To create a clean backup, Cray recommends shutting down the Cray system before beginning this procedure.

Also be aware that there may be a considerable load on the SMW while creating the SMW bootable backup drive.

1. Log on to the SMW as `crayadm` and `su` to `root`.

```
crayadm@smw> su - root
smw#
```

2. If the backup drive disk partition table already exists and the partition table on the backup drive matches the partition table that is on the primary boot drive, skip this step; otherwise, create the backup drive disk partition table.

NOTE:

For optimal performance, the source and destination disks should be on different buses; drive slots 0 and 1 are on a different bus than drive slots 2 and 3.

In this example, the partition table consists of the following:

- Slice 1: 4 GB Linux swap partition
- Slice 2: Balance of disk space used for the root file system

- a. Use the `fdisk` command to display the boot disk partition layout.

```
smw# fdisk -lu /dev/sda
Disk /dev/sda: 320.0 GB, 320072933376 bytes
255 heads, 63 sectors/track, 38913 cylinders, total 625142448 sectors
Units = sectors of 1 * 512 = 512 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1		63	8401994	4200966	82	Linux swap / Solaris
/dev/sda2	*	8401995	625137344	308367675	83	Linux

- b. Use the `fdisk` command to configure the bootable backup disk partition layout. Set the bootable backup disk partition layout to match the boot disk partition layout. First, clear all of the old partitions using the `d` command within `fdisk`; next create a Linux swap and a Linux partition; and then write the changes to the disk. For help, type `m` within `fdisk` (see the following sample output).

```
smw# fdisk -u /dev/sdb

The number of cylinders for this disk is set to 38913.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
 1) software that runs at boot time (e.g., old versions of LILO)
 2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK).

Command (m for help): p
Disk /dev/sdb: 320.0 GB, 320072933376 bytes
255 heads, 63 sectors/track, 38913 cylinders, total 625142448 sectors
Units = sectors of 1 * 512 = 512 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb1		63	8401994	4200966	82	Linux swap
/dev/sdb2		8401995	625105214	308351610	83	Linux

```
Command (m for help): d
Partition number (1-5): 2
Command (m for help): d
Selected partition 1
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
```

```

Partition number (1-4): 1
First sector (63-625105215, default 63): (Press the Enter key)
Using default value 63
Last sector or +size or +sizeM or +sizeK (63-625105215, default 625105215):
8401994

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap / Solaris)

Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 2
First sector (8401995-625105215, default 8401995): (Press the Enter key)
Using default value 8401995
Last sector or +size or +sizeM or +sizeK (8401995-625105215, default
625105215): \
(Press the Enter key)
Using default value 625105215

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

```

- c. Display the boot backup disk partition layout.

```

smw# fdisk -lu /dev/sdb
Disk /dev/sdb: 320.0 GB, 320072933376 bytes
255 heads, 63 sectors/track, 38913 cylinders, total 625142448 sectors
Units = sectors of 1 * 512 = 512 bytes

   Device   Boot    Start        End    Blocks   Id  System
/dev/sdc1           63     8401994    4200966   82  Linux swap / Solaris
/dev/sdc2    *   8401995   625137344   308367675   83  Linux

```

3. Initialize the swap device.

```
smw# mkswap /dev/sdb1
```

4. (If the device names have already been standardized, skip this step.) Standardize the `/etc/fstab` and `grub` disk device names. The device names that the installation process writes into the `/boot/grub/menu.lst` file are UDEV-based names (for example, `/dev/disk/by-id/scsi-SATA_ST3320620AS_922J3-part2` or `/dev/disk/by-id/ata-ST3320620A_9QFA85PV-part2`) instead of the more commonly used device names (for example, `/dev/sda2` or `/dev/hda2`). In the following procedures, edit the `/boot/grub/menu.lst` file to change only the long UDEV-based name to the shorter, commonly used device name reflected in the output of the `df` command. Be aware that errors in the `/boot/grub/menu.lst` will affect the ability to boot the SMW.

- a. SLES 11 sets up `/etc/fstab` and `/boot/grub/menu.lst` with UDEV-based names for the root device. For example:

```

smw# head -2 /etc/fstab
/dev/disk/by-id/scsi-SATA_ST3320620AS_9QF922J3-part2 / ext3 acl,user_xattr 1
1
/dev/disk/by-id/scsi-SATA_ST3320620AS_9QF922J3-part1 swap swap defaults 0 0
smw# more /boot/grub/menu.lst
###Don't change this comment - YaST2 identifier: Original name: linux###
title SUSE Linux Enterprise Server 11 - 3.0.101-0.46
root (hd0,1)
kernel /boot/vmlinuz-3.0.101-0.46-default \
root=/dev/disk/by-id/ata-ST3320620AS_5QF00F84-part2 \
resume=/dev/sda1 splash=silent crashkernel=256M-:128M@16M
showopts vga=0x31a initrd /boot/initrd-3.0.101-0.46-default
###Don't change this comment - YaST2 identifier: Original name: failsafe###
title Failsafe -- SUSE Linux Enterprise Server 11 - 3.0.101-0.46
root (hd0,1)
kernel /boot/vmlinuz-3.0.101-0.46-default \
root=/dev/disk/by-id/ata-ST3320620AS_5QF00F84-part2 showopts \
ide=nodma apm=off noresume edd=off powersaved=off nohz=off highres=off
processor.max_cstate=1 x11failsafe vga=0x31a
initrd /boot/initrd-3.0.101-0.46-default

```

- b. Execute the `df` command to get the name of the device to use in the `/etc/fstab` and `/boot/grub/menu.lst` files to replace the long UDEV-based device name.
For example:

```

smw# df
Filesystem      1K-blocks    Used    Available    Use%    Mounted on
/dev/sda2       303528624 40652904  247457340   15%     /
udev            1030780    460      1030320     1%      /dev

```

- c. Create a backup copy of the `/etc/fstab` and `/boot/grub/menu.lst` files.

```

smw# cp -p /etc/fstab /etc/fstab.save
smw# cp -p /boot/grub/menu.lst /boot/grub/menu.lst.save

```

- d. Edit the `/etc/fstab` file appropriately, using the device name (`dev`) from the `df` command output. In this example, the "1" and "2" refer to the partition names on the device. Change the following lines, which changes the long name `disk/by-id/scsi-SATA_ST3320620AS_9QF922J3-part2` to `sda2` and changes `disk/by-id/scsi-SATA_ST3320620AS_9QF922J3-part1` to `sda1`. Ensure that the swap is on `sda1`:

```

/dev/disk/by-id/scsi-SATA_ST3320620AS_9QF922J3-part2 / ext3 acl,user_xattr 1
1
/dev/disk/by-id/scsi-SATA_ST3320620AS_9QF922J3-part1 swap swap defaults 0 0
to:
/dev/sda2 / ext3 acl,user_xattr 1 1
/dev/sda1 swap swap defaults 0 0

```

- e. Edit the `/boot/grub/menu.lst` file appropriately; use the device name (`dev`) from the `df` command output. Change the long name `disk/by-id/scsi-SATA_ST3320620AS_9QF922J3-part2` to `sda2`. Change the following lines:

```

title SUSE Linux Enterprise Server 11 - 3.0.101-0.46
kernel /boot/vmlinuz-3.0.101-0.46-default \
root=/dev/disk/by-id/ata-ST3320620AS_5QF00F84-part2 \
resume=/dev/sda1 splash=silent crashkernel=256M-:128M@16M showopts vga=0x31a

```

to

```
title SUSE Linux Enterprise Server 11 - 3.0.101-0.46
kernel /boot/vmlinuz-3.0.101-0.46-default \
root=/dev/sda2 resume=/dev/sda1 splash=silent \
crashkernel=256M-:128M@16M showopts vga=0x31a
```

and change the following lines:

```
title Failsafe -- SUSE Linux Enterprise Server 11 - 3.0.101-0.46
kernel /boot/vmlinuz-3.0.101-0.46-default \
root=/dev/disk/by-id/ata-ST3320620AS_5QF00F84part2 \
showopts ide=nodma apm=off noresume edd=off powersaved=off nohz=off \
highres=off processor.max_cstate=1 x11failsafe vga=0x31a
```

to

```
title Failsafe -- SUSE Linux Enterprise Server 11 - 3.0.101-0.46
kernel /boot/vmlinuz-3.0.101-0.46-default \
root=/dev/sda2 showopts ide=nodma apm=off noresume edd=off \
powersaved=off nohz=off highres=off processor.max_cstate=1 x11failsafe
vga=0x31a
```

- f. Verify that the edited files are correct and match the output of the `df` command.

```
smw# head -2 /etc/fstab
/dev/sda2 / ext3 acl,user_xattr 1 1
/dev/sda1 swap swap defaults 0 0
smw# more /boot/grub/menu.lst
###Don't change this comment - YaST2 identifier: Original name: linux###
title SUSE Linux Enterprise Server 11 - 3.0.101-0.46
root (hd0,1)
kernel /boot/vmlinuz-3.0.101-0.46-default root=/dev/sda2 \
resume=/dev/sda1 splash=silent crashkernel=256M-:128M@16M showopts
initrd /boot/initrd-3.0.101-0.46-default
###Don't change this comment - YaST2 identifier: Original name: failsafe###
title Failsafe -- SUSE Linux Enterprise Server 11 - 3.0.101-0.46
root (hd0,1)
kernel /boot/vmlinuz-3.0.101-0.46-default \
root=/dev/sda2 showopts ide=nodma apm=off noresume edd=off
powersaved=off nohz=off highres=off initrd /boot/initrd-3.0.101-0.46-default
```

5. Update the `grub` device table to recognize any new drives added since the initial operating system installation.

IMPORTANT: Although all of the disks connected to the SMW are available to the system, `grub` detects only the first 16 devices. Therefore, if a disk is added to the SMW after the SMW is connected to the boot RAID, it is advisable to reboot the SMW before continuing this procedure.

- a. Back up the current `grub` device mapping file.

```
smw# mv /boot/grub/device.map /boot/grub/device.map-YYYYMMDD
```

- b. Invoke the `grub` utility to create a new device mapping file.

```
smw# grub --device-map=/boot/grub/device.map
Probing devices to guess BIOS drives. This may take a long time.
GNU GRUB version 0.97 (640K lower / 3072K upper memory)
grub> quit
```

The file `/boot/grub/device.map` is now updated to reflect all drives, using the standardized drive naming. To verify the contents of the file:

```
smw# cat /boot/grub/device.map
(fd0) /dev/fd0
(hd0) /dev/sda
(hd1) /dev/sdc
```

6. Create a new file system on the backup drive root partition by executing the `mkfs` command.

```
smw# mkfs -t ext3 /dev/sdb2
mke2fs 1.41.1 (01-Sep-2008)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
19275776 inodes, 77091918 blocks
3854595 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
2353 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
4096000, 7962624, 11239424, 20480000, 23887872, 71663616
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
This filesystem will be automatically checked every 33 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
smw#
```

7. Mount the new backup root file system on `/mnt`.

```
smw# mount /dev/sdb2 /mnt
```

8. Confirm the running root file system device.

```
smw# df
Filesystem      1K-blocks    Used    Available    Use%    Mounted on
/dev/sda2       303528624  6438700  281671544    3%      /
udev            1030332     116     1030216     1%      /dev
/dev/sdb2       306128812  195568  290505224    1%      /mnt
```

The running root file system device is the one mounted on `/`.

9. Dump the running root file system to the backup drive.

```
smw# cd /mnt
smw# dump 0f - /dev/sda2 | restore rf -
DUMP: WARNING: no file `/etc/dumpdates'
DUMP: Date of this level 0 dump: Thu Nov 11 06:55:29 2010
DUMP: Dumping /dev/sda2 (/) to standard output
DUMP: Label: none
DUMP: Writing 10 Kilobyte records
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 4003398 blocks.
```

```
DUMP: Volume 1 started with block 1 at: Thu Nov 11 06:57:38 2010
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
restore: ./lost+found: File exists
DUMP: 81.99% done at 10941 kB/s, finished in 0:01
DUMP: Volume 1 completed at: Thu Nov 11 07:04:01 2010
DUMP: Volume 1 4008910 blocks (3914.95MB)
DUMP: Volume 1 took 0:06:23
DUMP: Volume 1 transfer rate: 10467 kB/s
DUMP: 4008910 blocks (3914.95MB)
DUMP: finished in 383 seconds, throughput 10467 kBytes/sec
DUMP: Date of this level 0 dump: Thu Nov 11 06:55:29 2010
DUMP: Date this dump completed: Thu Nov 11 07:04:01 2010
DUMP: Average transfer rate: 10467 kB/s
DUMP: DUMP IS DONE
```

10. To make the backup drive bootable, reinstall the grub boot facility on that drive.

- a. Create a unique file on the backup drive to be used to identify that drive to the grub boot facility.

```
smw# cd /
smw# touch /mnt/THIS_IS_SDX
```

- b. Invoke the grub boot utility. Within grub:

1. Execute the find command to locate the drive designation that grub uses.
2. Select the drive to which the boot blocks will be installed with the root command.
3. Use the setup command to set up and install the grub boot blocks on that drive.

```
smw# grub --no-curses
GNU GRUB version 0.97 (640K lower / 3072K upper memory)
[ Minimal BASH-like line editing is supported. For the first word, TAB^
lists possible command completions. Anywhere else TAB lists the possible
completions of a device/filename. ]
grub> find /THIS_IS_SDX
find /THIS_IS_SDX
(hd1,1)
grub> root (hd1,1)
root (hd1,1)
Filesystem type is ext2fs, partition type 0x83
grub> setup (hd1)
setup (hd1)
Checking if "/boot/grub/stage1" exists... yes
Checking if "/boot/grub/stage2" exists... yes
Checking if "/boot/grub/e2fs_stage1_5" exists... yes
Running "embed /boot/grub/e2fs_stage1_5 (hd1)"... 17 sectors are embedded.
succeeded
Running "install /boot/grub/stage1 (hd1) (hd1)1+17 p
(hd1,1)/boot/grub/stage2 /boot/grub/menu.lst"... succeeded
Done.
grub> quit
```

Note that the Linux grub utility and boot system always refer to drives as `hd`, regardless of the actual type of the drives.

11. Unmount the backup root partition.

```
smw# umount /dev/sdb2
```

The drive is now bootable once plugged in or cabled as the primary drive.

(Optional) R630 SMW: Set Up the Bootable Backup Drive as an Alternate Boot Device

This optional procedure modifies a bootable backup drive for a Dell R630 SMW in order to boot from and run the R630 SMW from the backup root partition.

IMPORTANT: In order to boot from this backup drive, the primary boot drive must still be operable and able to boot the grub boot blocks installed. If the backup drive is modified to boot as an alternate boot device, it will no longer function as a bootable backup if the primary drive fails.

The disk device names shown in this procedure are only examples. Substitute the actual disk device names for this system. The boot disk is `pci-0000:03:00.0-scsi-0:0:0:0` and is slot 0, and the bootable backup disk is `pci-0000:03:00.0-scsi-0:0:1:0` and is slot 1.

1. Mount the backup drive's root partition.

```
smw# mount /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0-part2 /mnt
```

2. Create a new boot entry in the `/boot/grub/menu.lst` file. This entry should be a duplicate of the primary boot entry with the following changes:
 - a. Modify the title to uniquely identify the backup boot entry.
 - b. Modify the `root (hd0,1)` directive to reflect the `grub` name of the backup drive.
 - c. Modify the `root=` and `resume=` specifications to reference the backup drive device.

This is an example `/boot/grub/menu.lst` file. Note the new entry for the backup drive. This example references `pci-0000:03:00.0-scsi-0:0:0:0` (slot 0) as the primary drive and `pci-0000:03:00.0-scsi-0:0:1:0` (slot 1) as the backup drive.

```
smw# cp -p /boot/grub/menu.lst /boot/grub/menu.lst.20150916
smw# vi /boot/grub/menu.lst
smw# cat /boot/grub/menu.lst

# Modified by YaST2. Last modification on Thu Aug 13 19:38:47 CDT 2015
default 0
timeout 8
##YaST - generic_mbr
gfxmenu (hd0,1)/boot/message
##YaST - activate

###Don't change this comment - YaST2 identifier: Original name: linux###
title SUSE Linux Enterprise Server 11 SP3 - 3.0.101-0.46
    root (hd0,1)
    kernel /boot/vmlinuz-3.0.101-0.46-default root=/dev/disk/by-path/
pci-0000:03:00.0-scsi-0:0:0:0-part2 pci=bfsort resume=/dev/disk/by-path/
pci-0000:03:00.0-scsi-0:0:0:0-part1 splash=silent crashkernel=256M-:128M@16M
showopts biosdevname=X vga=0x31a
    initrd /boot/initrd-3.0.101-0.46-default
```

```

### New entry allowing a boot of the back-up drive when the primary drive
### is still present
title BACK-UP DRIVE - SUSE Linux Enterprise Server 11 SP3 - 3.0.101-0.46
    root (hd1,1)
    kernel /boot/vmlinuz-3.0.101-0.46-default root=/dev/disk/by-path/
pci-0000:03:00.0-scsi-0:0:1:0-part2 pci=bfsort resume=/dev/disk/by-path/
pci-0000:03:00.0-scsi-0:0:1:0-part1 splash=silent crashkernel=256M-:128M@16M
showopts biosdevname=X vga=0x31a
    initrd /boot/initrd-3.0.101-0.46-default

###Don't change this comment - YaST2 identifier: Original name: failsafe###
title Failsafe -- SUSE Linux Enterprise Server 11 SP3 - 3.0.101-0.46
    root (hd0,1)
    kernel /boot/vmlinuz-3.0.101-0.46-default root=/dev/disk/by-path/
pci-0000:03:00.0-scsi-0:0:0:0-part2 showopts ide=nodma apm=off noresume edd=off
powersaved=off nohz=off highres=off processor.max_cstate=1 nomodeset
x11failsafe biosdevname=X vga=0x31a
    initrd /boot/initrd-3.0.101-0.46-default

```

3. Modify the backup drive's /etc/fstab file to reference the secondary drive slot rather than the first drive slot. Examine the backup drive's fstab file. Edit the /mnt/etc/fstab file, changing pci-0000:03:00.0-scsi-0:0:0:0 to pci-0000:03:00.0-scsi-0:0:1:0 device names to reference the backup drive. In the following example, the backup drive is pci-0000:03:00.0-scsi-0:0:1:0-....

```

smw# cp /mnt/etc/fstab /mnt/etc/fstab.20150916
smw# cat /mnt/etc/fstab
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:0-part1swap          swap
defaults                0 0
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:0-part2 /              ext3
acl,user_xattr          1 1
proc                    /proc          proc           defaults      0 0
sysfs                   /sys           sysfs          noauto        0 0
debugfs                 /sys/kernel/debug debugfs        noauto        0 0
usbfs                   /proc/bus/usb  usbfs          noauto        0 0
devpts                  /dev/pts       devpts         mode=0620,gid=5 0 0
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:3:0-part1/var/opt/cray/disk/1 ext3
defaults                1 0
none /var/lib/dhcp/db ramfs defaults 0 0

smw# vi /mnt/etc/fstab
smw# cat /mnt/etc/fstab
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0-part1swap          swap
defaults                0 0
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:1:0-part2 /              ext3
acl,user_xattr          1 1
proc                    /proc          proc           defaults      0 0
sysfs                   /sys           sysfs          noauto        0 0
debugfs                 /sys/kernel/debug debugfs        noauto        0 0
usbfs                   /proc/bus/usb  usbfs          noauto        0 0
devpts                  /dev/pts       devpts         mode=0620,gid=5 0 0
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:3:0-part1/var/opt/cray/disk/1 ext3
defaults                1 0
none /var/lib/dhcp/db ramfs defaults 0 0

```

4. Unmount the backup drive.

```
smw# umount /mnt
```

The SMW can now be shut down and rebooted. Upon display of the Please select boot device prompt, select the BACK-UP DRIVE - SLES 11 entry to boot the backup root partition.

(Optional) R815 SMW: Set Up the Bootable Backup Drive as an Alternate Boot Device

This optional procedure modifies a bootable backup drive for a Dell R815 SMW in order to boot from and run the R815 SMW from the backup root partition.

IMPORTANT: In order to boot from this backup drive, the primary boot drive must still be operable and able to boot the grub boot blocks installed. If the backup drive is modified to boot as an alternate boot device, it will no longer function as a bootable backup if the primary drive fails.

The disk device names shown in this procedure are only examples. Substitute the actual disk device names for this system. The boot disk is phy7 and is slot 0, and the bootable backup disk is phy6 and is slot 1.

1. Mount the backup drive's root partition.

```
smw# mount /dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0-part2 /mnt
```

2. Create a new boot entry in the /boot/grub/menu.lst file. This entry should be a duplicate of the primary boot entry with the following changes:
 - a. Modify the title to uniquely identify the backup boot entry.
 - b. Modify the root (hd0,1) directive to reflect the grub name of the backup drive.
 - c. Modify the root= and resume= specifications to reference the backup drive device.

This is an example /boot/grub/menu.lst file. Note the new entry for the backup drive. This example references phy7 (slot 0) and as the primary drive and phy6 (slot 1) as the backup drive.

```
smw# cp -p /boot/grub/menu.lst /boot/grub/menu.lst.20110317
smw# vi /boot/grub/menu.lst
smw# cat /boot/grub/menu.lst
# Modified by YaST2. Last modification on Wed Jun 27 12:32:43 CDT 2012
default 0
timeout 8
##YaST - generic_mbr
gfxmenu (hd0,1)/boot/message
##YaST - activate

###Don't change this comment - YaST2 identifier: Original name: linux###
title SUSE Linux Enterprise Server 11 SP3 - 3.0.26-0.7
  root (hd0,1)
  kernel /boot/vmlinuz-3.0.26-0.7-default \
  root=/dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0-part2 \
  resume=/dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0-part1 \
  splash=silent crashkernel=256M-:128M showopts vga=0x31a
  initrd /boot/initrd-3.0.26-0.7-default

### New entry allowing a boot of the back-up drive when the primary drive
### is still present.
```

```

title BACK-UP DRIVE - SUSE Linux Enterprise Server 11 SP3 - 3.0.26-0.7
    root (hd1,1)
    kernel /boot/vmlinuz-3.0.26-0.7-default \
    root=/dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0-
part2 \
    resume=/dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0-
part1 \
    splash=silent crashkernel=256M-:128M showopts vga=0x31a
    initrd (hd0,1)/boot/initrd-3.0.26-0.7-default

###Don't change this comment - YaST2 identifier: Original name: failsafe###
title Failsafe -- SUSE Linux Enterprise Server 11 SP3 - 3.0.26-0.7
    root (hd0,1)
    kernel /boot/vmlinuz-3.0.26-0.7-default \
    root=/dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0-
part2 \
    showopts ide=nodma apm=off noresume edd=off powersaved=off \
    nohz=off highres=off processor.max_cstate=1 nomodeset x11failsafe vga=0x31a
    initrd /boot/initrd-3.0.26-0.7-default

```

3. Modify the backup drive's `/etc/fstab` file to reference the secondary drive slot rather than the first drive slot. Examine the backup drive's `fstab` file. Edit the `/mnt/etc/fstab` file, changing `phy7` to `phy6` device names to reference the backup drive. In the following example, the backup drive is `phy6-....`

```

smw# cp -p /mnt/etc/fstab /mnt/etc/fstab.20110317
smw# cat /mnt/etc/fstab
/dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0-part1 \
swap swap defaults 0 0
/dev/disk/by-path/pci-0000:05:00.0-sas-phy7-0x4433221107000000-lun-0-part2 \
/ ext3 acl,user_xattr 1 1
proc /proc proc defaults 0 0
sysfs /sys sysfs noauto 0 0
debugfs /sys/kernel/debug debugfs noauto 0 0
usbfs /proc/bus/usb usbfs noauto 0 0
devpts /dev/pts devpts mode=0620,gid=5 0 0
smw# vi /mnt/etc/fstab
smw# cat /mnt/etc/fstab
/dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0-part1 \
swap swap defaults 0 0
/dev/disk/by-path/pci-0000:05:00.0-sas-phy6-0x4433221106000000-lun-0-part2 \
/ ext3 acl,user_xattr 1 1
proc /proc proc defaults 0 0
sysfs /sys sysfs noauto 0 0
debugfs /sys/kernel/debug debugfs noauto 0 0
usbfs /proc/bus/usb usbfs noauto 0 0
devpts /dev/pts devpts mode=0620,gid=5 0 0

```

4. Unmount the backup drive.

```
smw# umount /mnt
```

The SMW can now be shut down and rebooted. Upon display of the `Please select boot device` prompt, select the `BACK-UP DRIVE - SLES 11` entry to boot the backup root partition.

R630 SMW: Replace a Failed LOGDISK

This procedure describes how to replace a failed LOGDISK on a Dell R630 SMW. To replace a failed LOGDISK on a Dell R815 SMW, see [R815 SMW: Replace a Failed LOGDISK](#) on page 326.

1. Replace the failed drive with the new drive.
2. Reboot the SMW.

```
smw# reboot
```

3. Reconfigure LOGDISK.

```
smw# /sbin/fdisk /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:3:0
Command (m for help): p
```

- a. Delete all the current partitions, if there are any.

```
Command (m for help): d
Partition number 4
Command (m for help): d
Partition number 3
Command (m for help): d
Partition number 2
Command (m for help): d
Partition number 1
Command (m for help): p
```

- b. Create the new, single partition.

```
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-121601, default 1): # Hit return, take the default
Using default value 1
Last cylinder, +cylinders or +size{K,M,G} (1-121601, default 121601):
# Hitreturn, take the default
Using default value 121601

Command (m for help): p

Disk /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:3:0:1000.2 GB, \
1000204886016 bytes
255 heads, 63 sectors/track, 121601 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00000083

Device Boot
Start   End   Blocks  Id System
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:3:0-part1 1 121601 976760001 83
Linux

Command (m for help): w
The partition table has been altered!
```

```
Calling ioctl() to re-read partition table.
Syncing disks.
```

4. Recreate the file system.

```
smw# mkfs -t ext3 -b 4096
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:3:0-part1
```

5. Mount the newly created file system.

```
smw# mount /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:3:0-part1/ var/opt/cray/
disk/1
```

6. The symbolic links should already be there to link this to /var/opt/cray/log.

```
smw# ls -al /var/opt/cray
```

7. Create the following new directories:

```
smw# mkdir /var/opt/cray/disk/1/log
smw# mkdir /var/opt/cray/disk/1/debug
smw# mkdir /var/opt/cray/disk/1/dump
```

8. Restart the `rsms` daemon.

```
smw# /etc/init.d/rsms restart
smw# /etc/init.d/dbMonitor restart
```

R815 SMW: Replace a Failed LOGDISK

This procedure describes how to replace a failed LOGDISK on a Dell R815 SMW. To replace a failed LOGDISK on a Dell R630 SMW, see [R630 SMW: Replace a Failed LOGDISK](#) on page 324.

1. Replace the failed drive with the new drive.
2. Reboot the SMW.

```
smw# reboot
```

3. Reconfigure LOGDISK.

```
smw# /sbin/fdisk /dev/disk/by-path/pci-0000:05:00.0\
-sas-phy4-0x4433221104000000-lun-0
Command (m for help): p
```

- a. Delete all the current partitions, if there are any.

```
Command (m for help): d
Partition number 4
Command (m for help): d
Partition number 3
Command (m for help): d
Partition number 2
```

```
Command (m for help): d
Partition number 1
Command (m for help): p
```

- b. Create the new, single partition.

```
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-121601, default 1): # Hit return, take the default
Using default value 1
Last cylinder, +cylinders or +size{K,M,G} (1-121601, default 121601):
# Hitreturn, take the default
Using default value 121601

Command (m for help): p

Disk /dev/disk/by-path/pci-0000:05:00.0-sas-phy4-0x4433221104000000-
lun-0:1000.2 GB, \
1000204886016 bytes
255 heads, 63 sectors/track, 121601 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00000083

Device Boot
Start   End     Blocks  Id System
/dev/disk/by-path/pci-0000:05:00.0-sas-phy4-0x4433221104000000-lun-0-part1 1
121601 976760001 83   Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

4. Recreate the file system.

```
smw# mkfs -t ext3 -b 4096
/dev/disk/by-path/pci-0000:05:00.0-sas-phy4-0x4433221104000000-lun-0-part1
```

5. Mount the newly created file system.

```
smw# mount /dev/disk/by-path/pci-0000:05:00.0-sas-phy4-0x4433221104000000\
-lun-0-part1/ var/opt/cray/disk/1
```

6. The symbolic links should already be there to link this to /var/opt/cray/log.

```
smw# ls -al /var/opt/cray
```

7. Create the following new directories:

```
smw# mkdir /var/opt/cray/disk/1/log
smw# mkdir /var/opt/cray/disk/1/debug
smw# mkdir /var/opt/cray/disk/1/dump
```

8. Restart the `rsms` daemon.

```
smw# /etc/init.d/rsms restart
smw# /etc/init.d/dbMonitor restart
```

Reboot a Stopped SMW

The SMW is an integral player in monitoring and maintaining optimal High Speed Network (HSN) traffic. If the SMW is down or being rebooted (i.e., not fully working), the blade controllers will automatically throttle the high-speed network because they are no longer hearing SMW heartbeats. This is done in order to prevent possible network congestion, which normally requires the SMW to be up in order to respond to such congestion. Once the SMW is up again, the blade controllers will unthrottle the network. (No attempt is made to prevent loss of data or to carry out operations that occur when the SMW is offline.) The consequences of throttling are that the network will perform much more slowly than normal.

When the SMW comes up, it restarts, establishes communications with all external interfaces, restores the proper state in the state manager, and continues normal operation without user intervention.

For a scheduled or unscheduled shutdown and reboot of the SMW, it is necessary to have a backup of the files so that if one or more of the files in use becomes corrupted, a clean set of files is available with which to reboot.

1. Verify that the configuration files contain the most recent system configuration.
2. Boot the SMW.

SMW Primary Disk Failure Recovery

To recover an SMW, an administrator must reorder the drives at the front of the SMW. No BIOS or software configuration changes are required.



CAUTION: Booting from the bootable backup disk is intended only for emergency use in the event of failure or loss of data on the primary disk.

1. Shutdown the OS on the SMW, if possible.
2. Power the SMW off.
3. Unplug the power cord.
4. For a desk-side SMW, open the disk drive access door, which is on the front of the SMW.
5. Remove the primary disk from its slot. For a desk-side SMW, the primary disk is located at the bottom of the column of disk drives at the front of the SMW. For a Rack-mount SMW, remove the disk drive that is in slot 0.
6. Remove the bootable backup disk and place it in the primary disk slot.
7. Press the reset button (front), if required.
8. Boot the SMW.

Remote Access to SMW

Virtual Network Computing (VNC) software enables the administrator to view and interact with the SMW from another computer. The Cray system provides a VNC server, `xvnc`. A VNC client must be downloaded in order to connect to it. See RealVNC (<http://www.realvnc.com/>) or TightVNC (<http://www.tightvnc.com/>) for more information.

The VNC software requires a TCP/IP connection between the server and the viewer. Some firewalls may not allow this connection due to security policies. VNC is considered to be an insecure protocol. See [For Workstation or Laptop Running Linux: Connect to the VNC Server Through an SSH Tunnel](#) for a procedure to create a secure connection to the SMW.

Cray supplies a VNC account `cray-vnc`.

Start the VNC server

1. Log on to the SMW as `root` user.
2. Use the `chkconfig` command to check the current status of the server:

```
smw# chkconfig vnc
vnc off
```

3. If the `chkconfig` command executed in step 2 reports that `Xvnc` was started by INET services (`xinetd`) execute the following commands to disable `xinetd` startup of `Xvnc`. Startup of `Xvnc` via `xinetd` is the SLES 11 default, but it usually is disabled by `chkconfig`:

```
smw# chkconfig vnc off
smw# /etc/init.d/xinetd reload
Reload INET services (xinetd). done
```

If no other `xinetd` services have been enabled, the `reload` command will return `failed` instead of `done`. If the `reload` command returns a `failed` notification, this is normal and can be ignored.

4. Use the `chkconfig` command to start `Xvnc` at boot time:

```
smw# chkconfig vnc on
```

5. Start the `Xvnc` server immediately:

```
smw# /etc/init.d/vnc start
```

If the password for `cray-vnc` has not already been established, the system prompts for one. Enter a password to access the server.

```
Password: *****
Verify:
Would you like to enter a view-only password (y/n)? n
xauth: creating new authority file /home/cray-vnc/.Xauthority
```

```
New 'X' desktop is smw:1
```

```
Creating default startup script /home/cray-vnc/.vnc/xstartup
Starting applications specified in /home/cray-vnc/.vnc/xstartup
Log file is /home/cray-vnc/.vnc/smw:1.log
```

To access the Xvnc server, use a VNC client, such as `vncviewer`, `tight_VNC`, `vnc4`, or a web browser. Direct it to the SMW that is running Xvnc. With many clients, it is possible to specify whether to connect in view-only or in an active mode. If active participation is specified, every mouse movement and keystroke made in the client is sent to the server. If more than one client is active at the same time, typing and mouse movements are intermixed.

Commands entered through the VNC client affect the system as if they were entered from the SMW. However, the main SMW window and the VNC clients cannot detect each other. It is a good idea for the administrator who is sitting at the SMW to access the system through a VNC client.

The startup script starts the Xvnc server for display :1.

6. Verify that Xvnc started:

```
smw# ps -e | grep vnc
1839 pts/0    00:00:00 Xvnc
```

For Workstation or Laptop Running Linux: Connecting to the VNC Server through an SSH Tunnel, using the `vncviewer -via` Option

IMPORTANT: This procedure is for use with the TightVNC client program.

Verify that the `vncviewer -via` option is available. If not, use [For Workstation or Laptop Running Linux: Connecting to the VNC Server through an SSH Tunnel](#) on page 330.

To connect from a workstation or laptop running Linux, enter the `vncviewer` command shown below.

The first password to enter is for `crayadm` on the SMW. The second password to enter is for the VNC server on the SMW, which was set the first time the VNC server was started using `/etc/init.d/vnc start`.

```
/home/mary> vncviewer -via crayadm@smw localhost:1
Password: *****
VNC server supports protocol version 3.130 (viewer 3.3)
Password: *****
VNC authentication succeeded
Desktop name "cray-vnc's X desktop (smw:1)"
Connected to VNC server, using protocol version 3.3
```

For Workstation or Laptop Running Linux: Connecting to the VNC Server through an SSH Tunnel

NOTE: This procedure assumes that the VNC server on the SMW is running with the default port of 5901.

1. This `ssh` command starts an `ssh` session between the local Linux computer and the SMW, and it also creates an SSH tunnel so that port 5902 on the `localhost` is forwarded through the encrypted SSH tunnel to port 5901 on the SMW. The system will prompt for the `crayadm` password on the SMW.

```
local_linux_prompt> ssh -L 5902:localhost:5901 smw -l crayadm
Password:
crayadm@smw>
```

2. Now `vncviewer` can be started using the local side of the SSH tunnel, which is port 5902. The system will prompt for the password of the VNC server on the SMW. This password was set when the VNC server was started for the first time using `/etc/init.d/vnc start` on the SMW.

```
local_linux_prompt> vncviewer localhost:2
Connected to RFB server, using protocol version 3.7
Performing standard VNC authentication
Password:
```

The VNC window from the SMW appears. All traffic between the `vncviewer` on the local Linux computer and the VNC server on the SMW is now encrypted through the SSH tunnel.

For Workstation or Laptop Running Mac OS X: Connecting to the VNC Server through an SSH Tunnel

NOTE: This procedure assumes that the VNC server on the SMW is running with the default port of 5901.

1. This `ssh` command starts an `ssh` session between the local Mac OS X computer and the SMW, and it also creates an SSH tunnel so that port 5902 on the localhost is forwarded through the encrypted SSH tunnel to port 5901 on the SMW. The system will prompt for the `crayadm` password on the SMW.

```
local_mac_prompt> ssh -L 5902:localhost:5901 smw -l crayadm
Password:
crayadm@smw>
```

2. Now `vncviewer` can be started using the local side of the SSH tunnel, which is port 5902. The system will prompt for the password of the VNC server on the SMW. This password was set when the VNC server was started for the first time using `/etc/init.d/vnc start` on the SMW.

The SSH tunnel is now ready for use. To bring up the `vncviewer` on a Mac OS X computer, type the following on the command line:

```
local_mac_prompt% open vnc://localhost:5902
```

All traffic between the `vncviewer` on the local Mac computer and the VNC server on the SMW is encrypted through the SSH tunnel.

For Workstation or Laptop Running Windows: Connecting to the VNC Server through an SSH Tunnel

NOTE: To connect from a computer running Windows, both a VNC client program, such as TightVNC, and an SSH program, such as PuTTY, SecureCRT, or OpenSSH are recommended.

1. The same method described in [For Workstation or Laptop Running Linux: Connecting to the VNC Server through an SSH Tunnel](#) can be used for computers running the Windows operating system.

Although TightVNC encrypts VNC passwords sent over the network, the rest of the traffic is sent unencrypted. To avoid a security risk, install and configure an SSH program that creates an SSH tunnel between TightVNC on the local computer (localhost port 5902) and the remote VNC server (localhost port 5901).

NOTE: Because the procedure for creating the SSH tunnel varies among different SSH programs for Windows computers, no instructions are provided here.

2. After installing TightVNC, start the VNC viewer program by double-clicking on the TightVNC icon. Enter the hostname and VNC screen number, `localhost: number` (such as, `localhost:2` or `localhost:5902`), and then select the Connect button.

SMW and CLE System Administration Commands

In addition to the SUSE Linux Enterprise Server (SLES) commands available, this table lists the Cray developed commands for administering Cray Linux Environment (CLE) on the Cray system.

The system provides the following types of commands for the system administrator:

- Hardware Supervisory System (HSS) commands invoked from the System Management Workstation (SMW) to control HSS operations; HSS commands are provided with SMW release packages.
- Cray Lightweight Log Management (LLM) System commands invoked from the SMW or on a CLE service node; the LLM commands are provided with both the SMW release packages and the CLE release packages.
- CLE commands invoked from a node to control the service and compute partitions; CLE commands are provided with CLE release packages.

Table 12. HSS Commands

Command	Description
dbMonitor	Controls the monitor process script that starts during system boot to watch <code>mysqld</code> and restart <code>mysqld</code> if it should crash
getSedcLogValues	Displays specified <code>sedc_manager</code> log file records
hss_make_default_initrd	Creates the default HSS controller boot image
hssbootlink	Links a Linux kernel <code>bzImage</code> file, an <code>initramfs</code> file, and a <code>parameters</code> file so that they can be booted on a Controller by using PXE boot on an SMW
hssclone	Clones the master image directory
hssds_init	Creates the Hardware Supervisory System (HSS) data store; ensures the proper HSS data store user credentials are created and that the data store is ready for operation
hsspackage	Facilitates creation of controller boot images
nid2nic	Prints all <code>nid-to-nic</code> address mappings
rtr	Routes the Cray network
sedc_manager	Invokes the System Environment Data Collections (SEDC) SMW manager
SMWconfig	Automatically configures software on SMW
SMWinstall	Automatically installs and configures software on SMW
SMWinstallCLE	Updates the SMW software on <code>bootroot</code> and <code>sharedroot</code> for system sets with CLE software installed
xtalive	Gets a response from an HSS daemon
xtbootdump	Parses a <code>bootinfo-file</code> to determine if <code>xtdumpsys</code> needs to be invoked

Command	Description
xtbootimg	Creates, extracts, or updates a Cray bootable image file
xtbootsys	Boots specified components in a Cray system
xtbounce	Powers components of the Cray system down then up
xtcreboot	Reboots specified cabinet or blade controllers on Cray XC series systems
xtcheckhss	Initiates a series of tests that validate the health of the HSS on Cray XC series systems
xtcheckmac	Checks for duplicate MAC addresses among L1 and L0 controllers on Cray XE and Cray XK systems
xtclass	Displays the network topology class for this system
xtclear	Clears component flags in the state manager
xtcli	Runs the HSS command line
xtcli boot	Specifies the types of components to boot
xtcli clear	Clears flag status in component state
xtcli part	Updates partition configurations
xtcli power	Powers a component up or down
xtcli set	Sets flag status in the component state
xtcon	Provides a two-way connection to the console of any running service node
xtconsole	Displays console text from a node
xtconsumer	Displays HSS events
xtcreport	Parses <code>xtn1rd</code> log file and display system network congestion protection information
xtcptop	Parses specified <code>xtn1rd</code> log file and displays real-time system network congestion protection information as it is written to the file
xtdaemonconfig	Configures HSS daemons dynamically
xtdimminfo	Collects and displays summaries from hardware errors reported in the console file on Cray XE and Cray XK systems
xtdiscover	Discovers and configures the Cray system hardware
xtdumpsys	Gathers information when a system stops responding or fails
xterrorcode	Displays event error codes
xtfileio	Reads or writes a file on an L1 or L0 controller
xtflash	Performs automated reflashing and rebooting of L1s and L0s on Cray XE and Cray XK systems
xtgenid	Generates HSS physical IDs
xtgetsyslog	Retrieves the <code>/var/log/messages</code> file from L1 and L0 controllers on Cray XE and Cray XK systems
xthb	Node heartbeat checker
xthwerrlog	Reports hardware errors

Command	Description
xthwerrlogd	Logs Gemini network errors
xthwinv	Retrieves hardware component information for selected modules
xtlogfilter	Filters information from event router log files
xtlogin	Logs on to cabinet and blade control processors
xtmcinfo	Gets microcontroller information from cabinet and blade control processors
xtmem2file	Reads CPU or Cray Gemini memory and saves it in a file
xtmemio	Reads or writes 32-bit or 64-bit words from CPU or Cray Gemini memory
xtmemwatch	Watches a memory location change on Cray XE and Cray XK systems
xtnetwatch	Watches the Cray system interconnection network for link control block (LCB) and router errors
xtnid2str	Converts node identification numbers to physical names
xtnlrd	Responds to fatal link errors by rerouting the system
xtnmi	Collects debug information from unresponsive nodes
xtpcimon	Monitors health of PCIe channels for Cray XC series systems
xtpget	Displays current system power usage and applied capping parameters for Cray XC series systems
xtpmaction	Implements power management actions for Cray XC series systems
xtpmdbconfig	Modifies power management configuration parameters and provides a mechanism for an operator to hook into database rotation events for Cray XC series systems
xtpscan	Controls power data collection and logging for Cray XC series systems
xtresview	Displays the current state of cabinets, blades, and links, and whether any have failed or been warm swapped out
xtrsh	Invokes a diagnostic utility that concurrently executes programs on batches of cabinet control processors and/or blade control processors
xtsedviewer	Command-line interface for SEDC
xtshow	Shows components with selected characteristics
xtwarmswap	Allows Cray system blades, chassis, or cabinets to be warm swapped
xtwatchsyslog	Shows all log messages for cabinet control processors (L1 controllers) and blade control processors (L0 controllers)

Table 13. LLM Commands

Command	Description
cray-syslog	Starts, stops, restarts, or checks the status of the log system
xtlog	Delivers messages to the Cray Lightweight Log Management (LLM) system
xtsession	Displays the current boot <i>sessionid</i>

Command	Description
xttail	Outputs the last part of Cray Lightweight Log Management (LLM) files
xttoday	Provides today's date in same format that is used to time stamp Cray Lightweight Log Management (LLM) log files
xttrim	Provides a simple and configurable method to automate the compression and deletion of old log files

Table 14. CLE Commands

Command	Description
apmgr	Provides interface for ALPS to cancel pending reservations.
apconf	A utility for manipulating and modifying ALPS configuration files.
cdump	Dumps node memory.
clcvf	A utility for configuring and validating Fine-grained Routing (FGR) on Cray systems.
crash	Analyzes Linux crash dump data or a live system (Red Hat utility).
csacon	Condenses records from the sorted pacct file.
csanodeacct	Initiates the end of application accounting on a node.
csanodemerg	Initiates collection of individual compute node accounting files.
csanodesum	Reads and consolidates application node accounting records.
dumpd	Initiates automatic dump and reboot of nodes when requested by Node Health Checker (NHC).
lastlogin	Records last date on which each user logged in
lbcd	Invokes the load balancer client daemon.
lbname	Invokes the load balancer service daemon.
lustre_control	Manages direct-attached and external Lustre file systems using standard Lustre commands and site specific Lustre file system definition and tuning files.
nhc_recovery	Releases compute nodes on a crashed login node that will not be rebooted.
pdsh	Issues commands to groups of hosts in parallel.
projdb	Creates and updates system project database for CSA.
rca-helper	Used in various administrative scripts to retrieve information from the Resiliency Communication Agent (RCA).
rsipd	Invokes the Realm-Specific IP Gateway Server.
sdbwarmswap	Updates the Service Database (SDB) when blades are replaced or added.
xt-lustre-proxy	Invokes the Lustre startup/shutdown, health monitor, and automatic failover utility.

Command	Description
xtalloc2db	Converts a text file to the <code>alloc_mode</code> table in the Service Database (SDB).
xtattr2db	Converts a text file to the <code>attributes</code> table in the Service Database (SDB).
xtauditctl	Distributes <code>auditctl</code> requests to nodes on a Cray system.
xtaumerge	Merges audit logs from multiple nodes into a single audit log file.
xtcdr2proc	Gets information from the RCA.
xtcheckhealth	Executes the Node Health Checker.
xtcleanup_after	Called by ALPS to check node health.
xtclone	Clones the master image directory and overlays a site-specific template.
xtcloneshared	Clones node or class directory in shared root hierarchy.
xtdb2alloc	Converts the <code>alloc_mode</code> table in the Service Database (SDB) to a text file.
xtdb2attr	Converts the <code>attributes</code> table in the Service Database (SDB) to a text file.
xtdb2etchosts	Converts service information in the SDB to a text file.
xtdb2filesystem	Converts the <code>filesystem</code> table of the SDB to a text file.
xtdb2gpus	Converts the <code>gpus</code> table in the Service Database (SDB) to a text file.
xtdb2lustrefailover	Converts the <code>lustre_failover</code> table in the SDB to a text file.
xtdb2lustreserv	Converts the <code>lustre_serv</code> table of the SDB to a text file.
xtdb2nodeclasses	Converts the <code>service_processor</code> table of the SDB to a text file.
xtdb2order	Converts the processor table <code>od_allocator_id</code> field in the Service Database (SDB) to a text file.
xtdb2proc	Converts the <code>processor</code> table of the SDB to a text file.
xtdb2segment	Converts <code>segment</code> table in the Service Database (SDB) to a text file.
xtdb2servcmd	Converts the <code>service_cmd</code> table of the SDB to a text file.
xtdb2servconfig	Converts the <code>service_config</code> table of the SDB to a text file.
xtdbsyncd	Invokes the HSS/SDB synchronization daemon.
xtfilesystem2db	Converts a text file to the SDB <code>filesystem</code> table.
xtfsck	Checks file systems on a system set defined in <code>/etc/sysset.conf</code> .
xtgetconfig	Gets configuration information from <code>/etc/sysconfig/xt</code> file.
xtgetdslroot	Returns compute node root path used within an environment.
xtgpus2db	Converts a text file to the SDB <code>gpus</code> table.
xthotbackup	Creates a backup copy of a system set on the boot RAID.
xthowspec	Displays file specialization in the shared root directory.

Command	Description
xtlusfoevntsndr	Sends failover events to clients for Lustre imperative recovery.
xtlusfoadmin	Displays Lustre automatic failover database tables and enables/disables Lustre server failover.
xtlustrefailover2db	Converts a text file to the SDB <code>lustre_failover</code> table.
xtlustreserv2db	Converts a text file to the SDB <code>lustre_service</code> table.
xtmount	Allows administrators to mount storage devices on the SMW based on their LABEL and FUNCTION roles in the <code>sysset.conf</code> file instead of long <code>/dev/disk/by-id</code> names.
xtnce	Displays or changes the class of a node.
xtnodeclasses2db	Converts a text file to the <code>service_processor</code> table in the SDB.
xtnodestat	Provides current job and node status summary information on a CNL compute node.
xtoparchive	Performs archive operations on shared root files from a given specification list.
xtopco	Checks out RCS versioned shared root specialized files.
xtopcommit	Commits changes made inside an <code>xtopview</code> session.
xtoprdump	Lists shared root file specification and version information.
xtoprlog	Provides RCS log information about shared root specialized files.
xtopview	Views file system as it would appear on any node, class of nodes, or all service nodes.
xtorder2db	Converts a text file to values in the <code>od_allocator_id</code> field of the processor table in the Service Database (SDB).
xtpackage	Facilitates creation of boot images.
xtpkgvar	Creates a skeleton structure of <code>/var</code> .
xtproc2db	Converts a text file to the <code>processor</code> table of the SDB.
xtprocadmin	Gets/sets the processor flag in the SDB.
xtrelswitch	Performs release switching by manipulating symbolic links in the file system and by setting the default version of modulefiles that are loaded at login.
xtrsipcfg	Generates and optionally installs the necessary RSIP client and server configuration files.
xtsegment2db	Converts a text file to <code>segment</code> table in the Service Database (SDB).
xtservcmd2db	Converts a text file to the <code>service_cmd</code> table of the SDB.
xtservconfig	Adds, removes, or modifies the <code>service_config</code> table of the SDB.
xtservconfig2db	Converts a text file to the <code>service_config</code> table of the SDB.
xtshutdown	Shuts down the service nodes in an orderly fashion.
xtspec	Specializes files for nodes or classes.

Command	Description
xtunspec	Unspecializes files for nodes or classes.
xtverifyconfig	Verifies the coherency of <code>/etc/init.d</code> files across all shared root views.
xtverifydefaults	Verifies and fixes inconsistent system default links within the shared root.
xtverifyshroot	Checks the configuration of the shared-root file system.

System Component States

Component state definitions are designated by uppercase letters. The state of `OFF` means that a component is present on the system. If the component is a blade controller, node, or ASIC, then this will also mean that the component is powered off. If the administrator disables a component, the state shown becomes `disabled`. When the `xtcli enable` command is used to enable that component for use once again, its state switches from `disabled` to `off`. In the same manner, enabling an empty component means that its state switches from `empty` to `off`.

The state of `EMPTY` components does not change when using the `xtcli enable` or the `xtcli disable` command, unless the force option (`-f`) is used.

Disabling of a cabinet, chassis, or blade will fail if any nodes under the component are in the `ready` state, unless the force option (`-f`) is used. An error message will indicate the reason for the failure.

Table 15. State Definitions

State	Cabinet Controller	Blade Controller	Cray ASIC	CPU	Link
OFF	Powered off	Powered off	Powered off	Powered off	Link is down
ON	Powered on	Powered on	Powered on and operational	Powered on	Link is up
HALT	--	--	--	CPU halted	--
STANDBY	--	--	--	Booting was initiated	--
READY	Operational	Operational	Operational	Booted	Operational

Table 16. Additional State Definitions (Common to all components)

State	Description
DISABLED	Operator disabled this component.
EMPTY	Component does not exist.
N/A	Component cannot be accessed by the system.
RESVD	Reserved; new jobs are not allocated to this component.

There are two notification flags, which can occur with any state.

WARNING A condition of the component was detected that is outside the normal operating range but is not yet dangerous.

ALERT A dangerous condition or fatal error has been detected for the component.

The state manager links the state of a node and the state of its GPU (if present), so that the state of the node and GPU are equal, except when a GPU has been disabled. If the GPU is disabled, it does not take part in any further state transitions and no flags are set on the GPU until the GPU is reenabled.

Administrative states are hierarchal, so disabling or enabling a component has a cascading effect on that component's children. A component may not be enabled if its parent component is disabled, but a subcomponent may be disabled without affecting its parents.

Table 17. xtcli Commands and Valid States

xtcli Command	Subcommand	Cabinet Controller	Blade Controller	Node
power	up	ON	OFF	OFF
	down	READY	ON	ON, HALT, DIAG
	up_slot (an alias for up)			
	down_slot (an alias for down)			
	force_down (an alias for down)			
halt		N/A	N/A	STANDBY, READY
boot		N/A	N/A	ON, HALT

Update the Time Zone

When the Cray Linux Environment (CLE) operating system is installed, the Cray system time is set at US/Central Standard Time (CST), which is six hours behind Greenwich Mean Time (GMT). An administrator can change this time.

When a Cray system is initially installed, the time zone set on the SMW is copied to the boot root, shared root and CNL boot images.

To change the time zone on the SMW, L0 controller, L1 controller, boot root, shared root, or for the compute node image, follow the appropriate procedure below.

Change the time zone for the SMW and the blade and cabinet controllers on XE systems



CAUTION: Perform this procedure while the Cray system is shut down; do not flash blade and cabinet controllers while the Cray system is booted.

You must be logged on as `root`. In this example, the time zone is changed from "America/Chicago" to "America/New_York".

1. Ensure the blade and cabinet controllers are responding. For example:

```
smw:~ # xtalive -a l0sysd s0
```

2. Optional: Check the current time zone setting for the SMW and controllers.

```
smw:~ # date
Wed Aug 01 21:30:06 CDT 2012

smw:~ # xtrsh -l root -s /bin/date s0
c0-0c0s2 : Wed Aug 01 21:30:51 CDT 2012
c0-0c0s5 : Wed Aug 01 21:30:51 CDT 2012
c0-0c0s7 : Wed Aug 01 21:30:51 CDT 2012
c0-0c1s1 : Wed Aug 01 21:30:51 CDT 2012
.
.
.
c0-0 : Wed Aug 01 21:30:52 CDT 2012
```

3. Verify that the `zone.tab` file in the `/usr/share/zoneinfo` directory contains the time zone you want to set.

```
smw:~ # grep America/New_York /usr/share/zoneinfo/zone.tab
US          +404251-0740023 America/New_York      Eastern Time
```

4. Create the time conversion information files.

```
smw:~ # date
Wed Aug 01 21:32:52 CDT 2012
smw:~ # /usr/sbin/zic -l America/New_York
smw:~ # date
Wed Aug 01 22:33:05 EDT 2012
```

5. Modify the `clock` file in the `/etc/sysconfig` directory to set the `DEFAULT_TIMEZONE` and the `TIMEZONE` variables to the new time zone.

```
smw:/etc/sysconfig # grep TIMEZONE /etc/sysconfig/clock
TIMEZONE="America/Chicago"
DEFAULT_TIMEZONE="US/Eastern"
smw:~ # vi /etc/sysconfig/clock
make changes
smw:~ # grep TIMEZONE /etc/sysconfig/clock
TIMEZONE="America/New_York"
DEFAULT_TIMEZONE="US/Eastern"
```

6. Copy the `/etc/localtime` file to `/opt/tftpboot`, and then restart the log system and `rsms`.

```
smw:~ # cp /etc/localtime /opt/tftpboot
smw:~ # /etc/init.d/cray-syslog restart
smw:~ # /etc/init.d/rsms restart
```

7. If this is the first time the time zone has been modified, complete this step. If the time zone has been changed already, skip this step and proceed to the next step.

- a. Exit from the root login.

```
smw:~ # exit
```

- b. Erase the flash memory of the L1s and flash the updated time zone.

```
crayadm@smw:~> fm -w -t 11
crayadm@smw:~> xtflash -t 11
```

- c. Erase the flash memory of the L0s and flash the updated time zone.

```
crayadm@smw:~> fm -w -t 10
crayadm@smw:~> xtflash -t 10
```

- d. Check the current time zone setting for the SMW and controllers.

```
crayadm@smw:~> date
Wed Aug 01 23:07:07 EDT 2012
crayadm@smw:~> xtrsh -l root -s /bin/date s0
c0-0c1s1 : Wed Aug 01 23:07:16 EDT 2012
c0-0c0s7 : Wed Aug 01 23:07:16 EDT 2012
c0-0c1s3 : Wed Aug 01 23:07:16 EDT 2012
.
.
.
c0-0 : Wed Aug 01 23:07:17 EDT 2012
```

Skip the next step and proceed with bouncing the system, below.

8. If the time zone has been changed already, complete this step.

- a. To update the L1's time zone:

```
smw:~ # xtrsh -l root -m ^c[0-9]+-[0-9]+$ -s 'atftp -g -r localtime \
-l $(readlink /etc/localtime) router && cp /etc/localtime /var/tftp'
```

- b. To update the L0's time zone:

```
smw:~ # xtrsh -l root -m s -s 'atftp -g -r localtime \
-l $(readlink /etc/localtime) router'
```

9. Bounce the system.

```
crayadm@smw:~> xtbounce s0
```

NOTE: An incompatibility exists between the current version of `/etc/localtime` and earlier versions that may be on the system. This incompatibility causes the `date` command to report an incorrect time on the compute nodes. To resolve this incompatibility, after updating the SMW software you will also need to update the time zone on the compute nodes as described in the procedure *Changing the time zone for compute nodes* in *Installing and Configuring Cray Linux Environment (CLE) Software*.

Change the Time Zone on the Boot Root and Shared Root

Prerequisites

User `root` privileges are required for this procedure.

Perform the following steps to change the time zone. In this example, the time zone is changed from "America/Chicago" to "America/New_York".

1. Confirm the time zone setting on the SMW.

```
smw:~ # cd /etc/sysconfig
smw:/etc/sysconfig # grep TIMEZONE clock
TIMEZONE="America/New_York"
DEFAULT_TIMEZONE="US/Eastern"
```

2. Log on to the boot node.

```
smw:/etc/sysconfig # ssh root@boot
```

3. Verify that the `zone.tab` file in the `/usr/share/zoneinfo` directory contains the appropriate time zone.

```
boot:~ # cd /usr/share/zoneinfo
boot:/usr/share/zoneinfo # grep America/New_York zone.tab
US          +404251-0740023 America/New_York      Eastern Time
```

4. Create the time conversion information files.

```
boot:/usr/share/zoneinfo # date
Mon Jul 30 22:50:52 CDT 2012
boot:/usr/share/zoneinfo # /usr/sbin/zic -l America\New_York
boot:/usr/share/zoneinfo # date
Mon Jul 30 23:59:38 EDT 2012
```

5. Modify the clock file in the /etc/sysconfig directory to set the `DEFAULT_TIMEZONE` and the `TIMEZONE` variables to the new time zone.

```
boot:/usr/share/zoneinfo # cd /etc/sysconfig
boot:~ # grep TIMEZONE clock
TIMEZONE="America/Chicago"
DEFAULT_TIMEZONE="US/Eastern"
boot:~ # vi clock
make changes
boot:~ # grep TIMEZONE clock
TIMEZONE="America/New_York"
DEFAULT_TIMEZONE="US/Eastern"
```

```
boot:/usr/share/zoneinfo # cd /etc/sysconfig
boot:~ # grep TIMEZONE clock
TIMEZONE="America/Chicago"
DEFAULT_TIMEZONE="US/Eastern"
boot:~ # vi clock
```

Make site-specific changes.

```
boot:~ # grep TIMEZONE clock
TIMEZONE="America/New_York"
DEFAULT_TIMEZONE="US/Eastern"
```

6. Switch to the default view by using `xtopview`.

IMPORTANT: If the SDB node has not been started, the `-x /etc/opt/cray/sdb/node_classes` option must be included when invoking the `xtopview` command.

```
boot:~ # xtopview
```

7. Verify that the `zone.tab` file in the `/usr/share/zoneinfo` directory contains the appropriate time zone.

```
default:// # grep America/New_York /usr/share/zoneinfo zone.tab
US          +404251-0740023 America/New_York          Eastern Time
```

8. Create the time conversion information files.

```
default:// # date
Mon Jul 30 23:10:52 CDT 2012
default:// # /usr/sbin/zic -l America/New_York
default:// # date
Tue Jul 31 00:11:38 EDT 2012
```

9. Modify the clock file in the /etc/sysconfig directory to set the `DEFAULT_TIMEZONE` and the `TIMEZONE` variables to the new time zone.

```
default:// # cd /etc/sysconfig
default://etc/sysconfig # grep TIMEZONE clock
TIMEZONE="America/Chicago"
DEFAULT_TIMEZONE="US/Eastern"
default://etc/sysconfig # vi clock
make changes
default://etc/sysconfig # grep TIMEZONE clock
TIMEZONE="America/New_York"
DEFAULT_TIMEZONE="US/Eastern"
```

10. Exit xtopview.

```
default:/etc/sysconfig # exit
```

Change the Time Zone for Compute Nodes

1. Exit from the boot node and confirm the time zone setting on the SMW.

```
boot:/usr/share/zoneinfo # exit
smw:/etc/sysconfig # grep TIMEZONE clock
TIMEZONE="America/New_York"
DEFAULT_TIMEZONE="US/Eastern"
```

2. Copy the new /etc/localtime file from the SMW to the bootimage template directory.

```
smw:/etc/sysconfig # cp -p /etc/localtime \
/opt/xt-images/templates/default/etc/localtime
```

3. Copy the new /usr/share/zoneinfo file from the SMW to the bootimage template directory. The directory to contain the time zone file must be created in the bootimage template area.

NOTE: This procedure enables a single time zone for the compute nodes. If users will be setting the `TIMEZONE` variable to time zones which are not the system default, you may wish to either copy a few of the common time zones used by the user community or the entire `/usr/share/zoneinfo` directory to the `/opt/xt-images/templates/default/` area.

```
smw:/etc/sysconfig # mkdir -p \
/opt/xt-images/templates/default/usr/share/zoneinfo/America
smw:~# cp -p /usr/share/zoneinfo/America/New_York \
/opt/xt-images/templates/default/usr/share/zoneinfo/America/New_York
```

4. Update the boot image to include these changes; follow the steps in [Prepare Compute and Service Node Boot Images](#) on page 37.

The time zone is not changed until you boot the compute nodes with the updated boot image.

PBS Professional Licensing for Cray Systems

PBS Professional uses a licensing scheme based on a central license server that allows licenses to float between servers. This reduces the complexity of managing environments with multiple, independent PBS installations and simplifies configuration when running other software packages that use the same license manager.

The PBS server and scheduler run on the Cray service database (SDB) node. By default, the SDB node is only connected to the Cray system high-speed network (HSN) and cannot access an external license server. Various options to set up network connectivity between the license server and the SDB node are detailed below. Determine which option is best suited to the site's needs and implement that solution prior to installing the PBS Professional software from Altair.

NOTE: Regardless of the option chosen, a PBS Professional MOM daemon must run on each login node where users may launch jobs.

PBS Professional configuration options on a Cray system include:

- **Running the PBS Professional server and scheduler on a Cray system service node.** If running the PBS Professional scheduler and server on a login node, be aware that these daemons consume processor and memory resources and have the potential to impact other processes running on the login node. In addition, any service running on a node where users are allowed to run processes increases the potential for interruption of that service. While these risks are generally low, it is important to consider them before selecting this option. Refer to [Migrate the PBS Professional Server and Scheduler](#) on page 348 to configure PBS Professional using this strategy.
- **Moving the PBS Professional server and scheduler external to the Cray system.** The PBS Professional scheduler requests MPP data from one of the MOM daemons running on the Cray system login nodes. The volume of this data is dependent upon the size and utilization of the Cray system. If running the PBS Professional scheduler outside of the Cray system, the scheduler cycle time could increase due to decreased bandwidth and increased latency in network communication. In most cases, the difference in cycle time is negligible. However, if the system has larger node counts (> 8192), consider avoiding this option. To configure PBS Professional for this strategy, refer to [Migrate the PBS Professional Server and Scheduler](#) on page 348.
- **Configuring the SDB node as an RSIP client.** This options allows the administrator to leave the PBS Professional scheduler and server on the SDB node. If RSIP is already running, this may be an attractive option. Cray recommends a dedicated network node for the RSIP server, which may not be desirable if RSIP is not already running. Follow the appropriate procedure in [Configure RSIP to the SDB Node](#) on page 349 to configure the SDB node as an RSIP client.
- **Configuring Network Address Translation (NAT) to forward IP packets to and from the SDB node.** This may be the best choice if intending to use packet forwarding exclusively for PBS Professional licensing and do not mind running NAT services on a login node. The steps to configure NAT IP forwarding to the SDB node are described in [Configure Network Address Translation \(NAT\) IP Forwarding](#) on page 352.
- **Installing a network card in the SDB node to connect it to the external network.** With this option it is not necessary to configure RSIP or NAT, but a PCIe network interface card (NIC) must be purchased for a modest cost. This is an attractive option if the goal is to access the SDB node directly from the external network. This procedure does not require connection through another node on the

Cray system. The steps to configure this option are covered in [Install and Configure a NIC](#) on page 354.

Cray recommends that system administrators consult their local networking and security staff prior to selecting one of these options. After a method for accessing the license server has been chosen and configured, complete the PBS Professional license manager configuration as described in the *Altair License Management System Installation Guide*. For additional information about using the `qmgr` command to set up the `pbs_license_file_location` resource, see the *PBS Professional Installation and Upgrade Guide* from Altair Engineering, Inc.

For more information, see: <http://www.pbsworks.com>.

Migrate the PBS Professional Server and Scheduler

Before migrating the PBS Professional server and scheduler off of the SDB node, select the target host. PBS Professional versions 9.2 and beyond are MPP aware, meaning they are capable of scheduling jobs to Cray systems. If a central PBS Professional server and scheduler already exist, simply add the Cray system to the list of nodes.

The first step is to install PBS Professional on the Cray system as described in the *PBS Professional Installation and Upgrade Guide*. The install procedure configures the SDB node as the PBS Professional server and scheduler host. Modify the default configuration to ensure that the PBS Professional scheduler and server do not start automatically on the SDB node.

1. If the PBS scheduler and server are running on the SDB node, log on to the SDB and stop the services.

```
sdb:~ # /etc/init.d/pbs stop
```

2. Log on to the Cray system boot node as root and unspecialize the PBS Professional configuration file for the SDB node. For example, if the SDB is node 3, type the following commands:

```
boot:~ # xtopview -m "Unspecialize pbs.conf on the SDB" -n 3
node/3:/ # xtunspec /etc/pbs.conf
node/3:/ # exit
boot:~ #
```

3. Edit the PBS Professional configuration file for the login nodes to point to the new server. The new server may be one of the login nodes or a host external to the Cray system. Set `PBS_SERVER` in `/etc/pbs.conf` to the new PBS Professional server host. For example, if the server is named `myserver`, type the following commands:

```
boot:~ #xtopview -m "Update pbs.conf for new server" -c login
class/login/: # vi /etc/pbs.conf
```

```
PBS_SERVER=myserver.domain.com
```

```
class/login/: # exit
```

- To migrate the server and scheduler to a login node and start PBS Professional automatically at boot time, specialize the `/etc/pbs.conf` file for that node. If the services are being moved to an external host, skip this step. For example, if the node ID of the login node is 4, type the following commands:

```
boot:~ # xtopview -m "Specialize pbs.conf for new server" -n 4
node/4:/ # xtspec /etc/pbs.conf
```

- Modify the `/etc/pbs.conf` file to start all of the PBS Professional services; for example:

```
node/4:/ # vi /etc/pbs.conf
```

```
PBS_START_SERVER=1
PBS_START_SCHED=1
PBS_START_MOM=1
```

```
node/4:/ # exit
```

- Log on to each of the login nodes as root and modify the PBS Professional MOM configuration file `/var/spool/PBS/mom_priv/config`. Change the `$clienthost` value to the name of the new PBS Professional server. For example, if the server is named `myserver`, type the following commands:

```
login2:~ # vi /var/spool/PBS/mom_priv/config
```

```
$clienthost myserver.domain.com
```

- After the configuration file has been updated, restart PBS Professional on each login node.

```
login2:~ # /etc/init.d/pbs restart
```

NOTE: This command starts the PBS Professional scheduler and server if they have been migrated to a login node.

- Log on to the new PBS Professional server host and add a host entry for each of the login nodes.

```
myserver:~ # qmgr
Qmgr: create node mycrayxt1
Qmgr: set node mycrayxt1 resources_available.mpphost=xhostname
Qmgr: create node mycrayxt2
Qmgr: set node mycrayxt2 resources_available.mpphost=xhostname
Qmgr: exit
```

At this point, the login nodes should be visible to the PBS Professional server.

Configure RSIP to the SDB Node

Follow the procedures in this section to configure the SDB node as an RSIP client. After the SDB node is configured as an RSIP client, refer to the *Altair License Management System Installation Guide* for detailed instructions about obtaining and installing the appropriate license manager components.

If RSIP is not configured on the system, follow the first procedure to generate a simple RSIP configuration with a single server and only the SDB node as a client.

[Install and Configure RSIP Using CLEinstall](#) on page 160 includes procedures to configure RSIP on a Cray system using the CLEinstall installation program. If RSIP is already configured using these procedures during the Cray Linux Environment (CLE) installation or upgrade, follow [Add the SDB Node as an RSIP Client to an Existing RSIP Configuration](#) on page 351 to add the SDB node as an RSIP client for one of the existing RSIP servers.

For additional information on configuring RSIP services, see [Configure Realm-specific IP Addressing \(RSIP\)](#).

IMPORTANT: Cray strongly recommends [Install, Configure, and Start RSIP Clients and Servers](#) on page 161 for RSIP configuration.

1. Boot the system as normal. Ensure all the service nodes are available, and ensure that the system is setup to allow password-less ssh access for the root user.
2. Select a service node to run the RSIP server. The RSIP server node must have external Ethernet connectivity and must not be a login node. In this example the physical ID for the RSIP server is `c0-0c0s6n1`.
3. Specialize the `rsipd.conf` file by node ID and install the `rsip.conf` file to the shared root. Additionally, tune the RSIP servers by updating the associated `sysctl.conf` file. Invoke the following steps for the RSIP server node.

- a. Log on to the boot node and invoke `xtopview` in the node view.

```
boot:~ # xtopview -n c0-0c0s6n1
```

- b. Specialize `/etc/opt/cray/rsipd/rsipd.conf` for the specified node.

```
node/c0-0c0s6n1:/ # xtspec /etc/opt/cray/rsipd/rsipd.conf
```

- c. Copy the `rsip.conf` template file from the SMW to the shared root.

```
node/c0-0c0s6n1:/ # scp \
crayadm@smw:/opt/cray-xt-rsipd/default/etc/rsipd.conf.example \
/etc/opt/cray/rsipd/rsipd.conf
```

- d. Modify the `port_range`, `ext_if` and `max_clients` parameters in the `rsipd.conf` file as follows: If the external Ethernet interface is not `eth0`, modify `ext_if` accordingly. For example,

```
ext_if eth1
```

```
node/c0-0c0s6n1:/ # vi /etc/opt/cray/rsipd/rsipd.conf
```

```
port_range 8192-60000
max_clients 2
Uncomment:
ext_if eth0
```

- e. Specialize the `/etc/sysctl.conf` file and modify the OS port range so that it does not conflict with the RSIP server.

```
node/c0-0c0s6n1:/ # xtspec /etc/sysctl.conf
node/c0-0c0s6n1:/ # vi /etc/sysctl.conf
```

```
net.ipv4.ip_local_port_range = 60001 61000
```

- f. If the specified RSIP server is using a 10GbE interface, update the default socket buffer settings by modifying the following lines in the `sysctl.conf` file.

```
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 0
net.core.rmem_max = 524287
net.core.wmem_max = 524287
net.core.rmem_default = 131072
net.core.wmem_default = 131072
net.ipv4.tcp_rmem = 131072 1000880 9291456
net.ipv4.tcp_wmem = 131072 1000880 9291456
net.ipv4.tcp_mem = 131072 1000880 9291456
```

- g. Update the `udev` rules to skip the `ifup` of the `rsip` interfaces as they are created. Add `rsip*` to the list of interface names for `GOTO="skip_ifup"`.

```
node/c0-0c0s6n1:/ # xtspec /etc/udev/rules.d/31-network.rules
node/c0-0c0s6n1:/ # vi /etc/udev/rules.d/31-network.rules
```

```
SUBSYSTEM=="net", ENV{INTERFACE}=="rsip*|ppp*|ipp*|isdn*|plip*|lo*|irda*|
\
dummy*|ipsec*|tun*|tap*|bond*|vlan*|modem*|dsl*", GOTO="skip_ifup"
```

- h. Exit the `xtopview` session.

```
node/c0-0c0s6n1:/ # exit
```

4. Update the boot automation script to start the RSIP client on the SDB node. This line is simply a `modprobe` of the `krrip` module with the IP argument pointing to the HSN IP address of the RSIP server node and specifying the requested number of ports; place the new line towards the end of the file, immediately before any `'motd'` or `'ip route add'` lines. For example, if the IP address of the RSIP server is `10.128.0.14` and 32 ports are requested, type the following commands.

```
crayadm@smw:~> vi /opt/cray/etc/auto.xthostname
```

```
# RSIP client startup
lappend actions { crms_exec_via_bootnode "sdb" "root" "modprobe krrip
ip=10.128.0.14use_rsip_local_ports=1 num_ports=32" }
```

Add the SDB Node as an RSIP Client to an Existing RSIP Configuration

1. Select one of the RSIP servers to provide RSIP access for the SDB node. In this example, the RSIP server has the physical ID `c0-0c0s6n1`.
2. Log on to the boot node and invoke `xtopview` in the node view for the selected RSIP server.

```
boot:~ # xtopview -n c0-0c0s6n1
```

3. Modify the `max_clients` parameters in the `rsipd.conf` file; Add two more clients to make room for the new SDB node. For example, if configuring 300 RSIP clients (compute nodes), type the following:

```
node/c0-0c0s6n1:/ # vi /etc/opt/cray/rsipd/rsipd.conf
```

```
max_clients 302
```

4. Update the boot automation script to start the RSIP client on the SDB node. Do this after the line that starts the RSIP server. This line is simply a `modprobe` of the `krsip` module with the IP argument pointing to the HSN IP address of the RSIP server node. For example, if the IP address of the RSIP server is `10.128.0.14`, type the following commands.

```
crayadm@smw:~> vi /opt/cray/etc/auto.xthostname
```

```
# RSIP client startup
lappend actions { crms_exec_via_bootnode "sdb" "root" "modprobe krsip
ip=10.128.0.14 rsip_local_ports=1" }
```

Configure Network Address Translation (NAT) IP Forwarding

1. Select a login node to act as the NAT router. Cray recommends that selecting the node with the lowest load or network latency. For this example the login node is named `login2`.
2. Log on to the selected node and invoke the `ifconfig` command to obtain the IP address of the routing node. If the Cray system has a Gemini-based, system interconnection network, type this command:

```
login2:/ # ifconfig ipogif0
ipogif0  Link encap:Ethernet  HWaddr 00:01:01:00:00:04
         inet addr:10.128.0.3  Mask:255.252.0.0
         inet6 addr: fe80::201:1ff:fe00:4/64 Scope:Link
         UP RUNNING NOARP  MTU:16000  Metric:1
         RX packets:1543290 errors:0 dropped:0 overruns:0 frame:0
         TX packets:1640783 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:1643894879 (1567.7 Mb)  TX bytes:1446996661 (1379.9 Mb)
```

The IP address of the routing node is indicated as `inet addr` (in this case, `10.128.0.3`).

3. Record the Ethernet interface used on this login node. For example:

```
login2:/ # netstat -r | grep default
default  cfgw-12-hsrp.us 0.0.0.0    UG        0 0        0 eth0
```

Following this example, use the Ethernet interface, `eth0`, in the NAT startup script that is created in the next step.

4. Edit `/etc/hosts` on the shared root to include the external license server(s). Add these entries prior to the first local Cray IP addresses; that is, before the `10.128.x.y` entries. For example:

```
boot:~# xtopview
default:/: # vi /etc/hosts
```

```
10.0.0.55    tic tic.domain.com
10.0.0.56    tac tac.domain.com
10.0.0.57    toe toe.domain.com
```

```
default:/: # exit
```

5. In the same manner, edit `/etc/hosts` on the boot root to include entries for the external license server(s).

```
boot:~# vi /etc/hosts
```

```
10.0.0.55    tic tic.domain.com
10.0.0.56    tac tac.domain.com
10.0.0.57    toe toe.domain.com
```

6. In the default `xtopview` view, create and/or edit the `/etc/init.d/local.start-nat` file on the shared root, adding the following text:

```
boot:~# xtopview
default:/: # vi /etc/init.d/local.start-nat
```

```
#!/bin/bash
### BEGIN INIT INFO
# Provides:          local.start-nat
# Required-Start:
# Required-Stop:
# Default-Start:
# Default-Stop:
# Description:       Set up NAT IP forwarding
### END INIT INFO

echo "Setting up NAT IP forwarding."
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -F
iptables -A FORWARD -i eth0 -o ipogif0 -m state --state ESTABLISHED,RELATED -
j ACCEPT
iptables -A FORWARD -i ipogif0 -o eth0 -j ACCEPT
iptables -A FORWARD -j LOG
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

7. Add execute permissions to the `local.start-nat` file:

```
default:/: # chmod 755 /etc/init.d/local.start-nat
```

8. Exit the `xtopview` session.

```
default:/: # exit
```

9. Log on as `root` to the selected router node and start the NAT service. Use the `iptables` command to verify that forwarding is active.

```
login2:~ # /etc/init.d/local.start-nat
login2:~ # iptables -L -n
```

```
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination      state
ACCEPT     all  --  0.0.0.0/0              0.0.0.0/0
RELATED,ESTABLISHED
ACCEPT     all  --  0.0.0.0/0              0.0.0.0/0
LOG        all  --  0.0.0.0/0              0.0.0.0/0          LOG flags 0 level 4

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
login2:~ #
```

10. Add a new default route on the SDB node. Ensure that this route does not currently exist. For example, if the routing node IP interface identified in step 2 on page 352 is 10.128.0.3, type this command:

```
login2:~ #ssh sdb /sbin/route add default gw 10.128.0.3
```

11. Test the new route by invoking the ping command and ensuring the service node can access external servers by name.
12. Edit the boot automation script to Configure NAT services. For example, if the IP address identified in step 2 on page 352 is 10.128.0.3:

```
smw:~# vi /opt/cray/etc/auto.xthostname
```

Add the following lines just prior to the ALPS/PBS startup:

```
lappend actions { crms_exec_via_bootnode "login2" "root" \
"/etc/init.d/local.start-nat" }
lappend actions { crms_exec_via_bootnode "sdb" "root" \
"/sbin/route add default gw 10.128.0.3" }
```

NAT services should now restart automatically upon the next reboot of the Cray system.

Install and Configure a NIC

Obtain a PCIe compliant NIC. Intel 82546 based cards have been verified with Cray system SDB nodes. Follow this procedure to install the network card in the SDB node and connect it to the external network. Note that rebooting the system is required as part of this procedure.

1. Prior to shutting the system down, perform the following steps on the boot node to ensure the new NIC is configured upon the ensuing reboot. Invoke `xtopview` in the node view for the SDB node. For example, if the SDB is node 3, the IP address to assign on the external network is 172.30.10.100, the appropriate netmask is 255.255.0.0, and the default gateway IP is 172.30.10.1, type these commands.

```
boot:~# xtopview -m "add eth0 interface" -n 3
node/3:/ # cd /etc/sysconfig/network
node/3:/ # xtspec ifcfg-eth0
node/3:/ # vi ifcfg-eth0
```

Add the following content:

```
DEVICE="eth0"
BOOTPROTO="static"
STARTMODE="onboot"
IPADDR=172.30.10.100
NETMASK=255.255.0.0
```

```
node/3:/ # touch routes
node/3:/ # xtspec routes
node/3:/ # echo 'default 172.30.10.1 - -' >routes
node/3:/ # exit
```

2. Edit the `/etc/hosts` file on the shared root and add entries for the external license server(s). For example:

```
boot:~# xtopview
default:/ # vi /etc/hosts
```

Add these entries prior to the first local Cray system IP addresses:

```
10.0.0.55    tic tic.domain.com
10.0.0.56    tac tac.domain.com
10.0.0.57    toe toe.domain.com
```

```
default:/ # exit
default:/ # exit
```

3. Shut down the system.

```
smw:~# xtbootsys -s last -a auto_xtshutdown
```

4. Power down the slot where the SDB node is installed. For example:

```
smw:~# xtcli power down_slot -f c0-0c0s0
```

5. Pull the blade, physically insert the new NIC into the PCIe slot of the SDB node and reinsert the blade into the slot.

6. Power up the slot where the SDB node is installed. For example:

```
smw:~# xtcli power up_slot -f c0-0c0s0
```

7. Connect the NIC to the Ethernet network on which the license server is accessible.

8. Boot the Cray system.

9. Log on to the SDB node and invoke the `ifconfig` command to confirm that the SDB shows the new `eth0` interface.

```
nid00003:~ # /sbin/ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:04:23:DF:4C:56
          inet addr:172.30.10.100  Bcast:172.30.10.1  Mask:255.255.0.0
          inet6 addr: 2001:408:4000:40c:204:23ff:fedf:4c56/64  Scope:Global
          inet6 addr: 2600:805:100:40c:204:23ff:fedf:4c56/64  Scope:Global
          inet6 addr: fe80::204:23ff:fedf:4c56/64  Scope:Link
```

```
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:428807 errors:0 dropped:0 overruns:0 frame:0
TX packets:10400 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:34426088 (32.8 Mb) TX bytes:1292332 (1.2 Mb)
Base address:0x2fc0 Memory:fece0000-fed00000
```

10. Ping the license server from the SDB node.

```
nid00003:~ # ping tic.domain.com
```