Managing System Software for Cray XE™ Systems

S–2393–4001

RECORD OF REVISION

S–2393–4001 Published September 2011 Supports the Cray Linux Environment (CLE) 4.0.UP01 release and the System Management Workstation (SMW) 6.0.UP01 release.

S–2393–4001 Published August 2011 Limited Availability draft; supports the Cray Linux Environment (CLE) 4.0.UP00 release and the System Management Workstation (SMW) 6.0.UP00 release.

S–2393–3102 Published December 2010 Supports the Cray Linux Environment (CLE) 3.1.02 release and the System Management Workstation (SMW) 5.1.02 release.

3.1 Published June 2010 Supports the Cray Linux Environment (CLE) 3.1 release and the System Management Workstation (SMW) 5.1 release.

3.0 Published March 2010 Supports the Cray Linux Environment (CLE) 3.0 release and the System Management Workstation (SMW) 5.0 release.

2.2 Published July 2009 Supports the general availability (GA) release of Cray XT systems running the Cray Linux Environment (CLE) 2.2 release and the general availability (GA) release of the System Management Workstation (SMW) 4.0 release.

2.1 Published November 2008 Supports the general availability (GA) release of Cray XT systems running the Cray Linux Environment (CLE) 2.1 release and the System Management Workstation (SMW) 3.1 release as of the SMW 3.1.09 update.

2.0 Published October 2007 Supports general availability (GA) release of Cray XT series systems running the Cray XT series Programming Environment 2.0, UNICOS/lc 2.0, and System Management Workstation 3.0.1 releases.

1.5 Published October 2006 Supports general availability (GA) release of Cray XT series systems running the Cray XT series Programming Environment 1.5, UNICOS/lc 1.5, and System Management Workstation 1.5 releases.

1.4 Published May 2006 Supports Cray XT3 systems running the Cray XT3 Programming Environment 1.4, Cray XT3 RAS and Management System (CRMS) 1.4, and UNICOS/lc 1.4 releases.

1.3 Published November 2005 Supports Cray XT3 systems running the Cray XT3 Programming Environment 1.3, System Management Workstation (SMW) 1.3, and UNICOS/lc 1.3 releases.

1.2 Published September 2005 Supports Cray XT3 systems running the Cray XT3 Programming Environment 1.2, System Management Workstation (SMW) 1.2, and UNICOS/lc 1.2 releases.

1.1 Published June 2005 Supports Cray XT3 systems running the Cray XT3 Programming Environment 1.1, System Management Workstation (SMW) 1.1, and UNICOS/lc 1.1 releases.

1.0 Published February 2005 Draft documentation to support Cray XT3 limited availability (LA) systems.

Changes to this Document

S–2393–4001

Added information

- The following procedures were added:
  - Procedure 56 on page 222
  - Procedure 58 on page 233
  - Procedure 59 on page 235
  - Procedure 77 on page 307
  - Procedure 29 on page 144
  - Procedure 30 on page 145
- The following section was added: Configuring Memory Control Groups on page 143.

Revised information

- The HSS file systems hierarchy changed as of the SMW 6.0.UP00 release to adhere to Filesystem Hierarchy Standard (Linux Standard Base, LSB, compliant). This guide was updated throughout to reflect the new paths, with are: `/opt/cray/hss/default/{bin, etc, lib, lib64, man, sys}`; `/var/opt/cray/{log, dump, debug}`.
- Table 1, including adding the accelerator module (GPU) naming convention.
- The boot blade was renamed to system blade; documentation changed accordingly.
- The `xtcli power up` and `power down` commands are now only used for cabinets and blades. In addition, the `xtcli power up` command will not attempt to power up network mezzanine cards or nodes, which are handled by the `xtbounce` command during system boot. Also the `xtcli up_slot` command is now an alias for the `xtcli power up` command, and the `xtcli down_slot` and `force_down` commands are now aliases for the `xtcli power down` command.
- Managing User Accounts on Service Nodes on page 114 was corrected.
- Enabling LDAP Support for User Authentication on page 116.
- Setting Node Attributes Using the `/etc/opt/cray/sdb/attr.xthwinv.xml` and `/etc/opt/cray/sdb/attr.defaults` Files on page 198 was revised to reflect the use of the `attr.xthwinv.xml` file, rather than the `attr.xthwinv` file.
- During new system installations and any updates or upgrades, the `CLEinstall` program disables execute permissions on all `cron` scripts in the `/etc/cron.*` directories. You must manually enable any scripts you want to use following the installation.

- Configuring Realm-specific IP Addressing (RSIP) on page 206 has been updated to support the configuration of service node RSIP clients with the `CLEinstall` utility.
- `ALPS_SHARED_DIR_PATH` is removed from `/etc/sysconfig/alps` and is set in `/etc/alps.conf` by the `sharedDir` option.
- The following procedures were revised:
  - Procedure 57 on page 228.
  - Procedure 15 on page 94.
  - Procedure 16 on page 94.
  - Procedure 26 on page 133.
  - The procedure titled "Finding files in `/etc` that are specialized by class" has been replaced by Example 64.
  - Procedure 95 on page 359.
  - The procedure titled "Finding specialization of a file on a node" has been replaced by Example 65.
  - Procedure 27 on page 141.
  - Procedure 50 on page 211.
  - Procedure 51 on page 214.
  - Procedure 55 on page 221.
  - Appendix B, System States on page 327.
  - Appendix C, Error Codes on page 329.

Deleted information

- The following previously deprecated commands were removed: `xtps`, `xtkill`, `xtuname`, and `xtwho`.
- The `xtcli -f` and `-x` options were removed.

Contents

Page

Introduction [1] **27**

1.1 Audience for This Guide 28

1.2 Cray System Administration Publications 28

1.3 Related Publications 29

Introducing System Components [2] **31**

2.1 System Management Workstation (SMW) 33

2.2 CLE 34

2.3 Boot Root File System 35

2.4 Shared Root File System 35

2.5 Service Nodes 35

2.5.1 Boot Node 36

2.5.2 Service Database (SDB) Node 37

2.5.3 Syslog Node 37

2.5.4 Login Nodes 38

2.5.5 Network Nodes 38

2.5.6 I/O Nodes 38

2.5.7 Services on the Service Nodes 38

2.5.7.1 Resiliency Communication Agent (RCA) 38

2.5.7.2 Lustre File System 39

2.5.7.3 Cray Data Virtualization Service (Cray DVS) 39

2.5.7.4 Application Level Placement Scheduler (ALPS) for Compute Nodes 40

2.5.7.5 Cluster Compatibility Mode 41

2.5.7.6 Repurposing CNL Compute Nodes as Service Nodes 41

2.5.7.7 IP Implementation 41

2.6 Compute Nodes 42

2.7 Job Launch Commands 43

2.8 Node Health Checker (NHC) 43

2.9 Comprehensive System Accounting (CSA) 44

2.10 Checkpoint/Restart (CPR) 44

Page

2.11 Optional Workload-management (Batch) System Software Products 44
2.12 Hardware Supervisory System (HSS) 45
2.12.1 HSS Network 45
2.12.2 HSS Interface 46
2.12.3 Blade Control Processor (L0 Controller) and Cabinet Control Processors (L1 Controller) . . . 46
2.12.4 NTP Server 47
2.12.5 Event Router 47
2.12.6 HSS Managers 47
2.12.6.1 State Manager 48
2.12.6.2 Boot Manager 48
2.12.6.3 System Environmental Data Collections (SEDC) Manager 48
2.12.6.4 Flash Manager 49
2.12.6.5 NID Manager 49
2.12.7 Automatically Discover and Configure Cray System Hardware 50
2.12.8 Cray System Network Routing Utility 51
2.12.9 Event Logs 51
2.12.10 Boot Logs 51
2.12.11 Dump Logs 51
2.13 Cray Management Services (CMS) 51
2.14 Storage 51
2.15 Other Administrative Information 52
2.15.1 Identifying Components 52
2.15.1.1 Physical ID 53
2.15.1.2 Node ID (NID) 56
2.15.1.3 Class Name 57
2.15.2 Topology Class 58
2.15.3 Persistent `/var` Directory 58
2.15.4 Default Network IP Addresses 58
2.15.5 `/etc/hosts` Files 58
2.15.6 Realm-Specific IP Addressing (RSIP) for CNL Compute Nodes 60
2.15.7 Security Auditing 60
2.15.8 Logging Failed Login Attempts 60
2.15.9 Logical Machines 60

Managing the System [3] 63

3.1 Connecting the SMW to the Console of a Service Node 63
3.2 Logging On to the Boot Node 63
3.3 Preparing a Service Node and Compute Node Boot Image 64

Page

3.3.1 Using `shell_bootimage_`*label*`.sh` to Prepare Boot Images 66
3.3.2 Changing Boot Parameters 69
3.4 Booting Nodes 69
3.4.1 Booting the System 69
3.4.2 Using the `xtcli boot` Command to Boot a Node or Set of Nodes 72
3.4.3 Rebooting a Single Compute Node 72
3.4.4 Rebooting Login or Network Nodes 72
3.5 Requesting and Displaying System Routing 73
3.6 Shutting Down the System Using the `auto.xtshutdown` File 74
3.7 Shutting Down Service Nodes Using the `xtshutdown` Command 74
3.8 Shutting Down the System or Part of the System Using the `xtcli shutdown` Command . . . 75
3.9 Stopping System Components 76
3.9.1 Reserving a Component 76
3.9.2 Powering Down Blades or Cabinets 76
3.9.3 Halting Selected Nodes 77
3.10 Restarting a Blade or Cabinet 77
3.11 Aborting Active Sessions on the HSS Boot Manager 78
3.12 Displaying and Changing Software System Status 78
3.12.1 Displaying the Status of Nodes from the Operating System 78
3.12.2 Viewing and Changing the Status of Nodes 79
3.12.3 Marking a Compute Node as a Service Node 80
3.12.4 Finding Node Information 80
3.12.4.1 Translating Between Physical ID Names and Integer NIDs 80
3.12.4.2 Finding Node Information Using the `xtnid2str` Command 80
3.12.4.3 Finding Node Information Using the `nid2nic` Command 81
3.13 Displaying and Changing Hardware System Status 81
3.13.1 Generating HSS Physical IDs 82
3.13.2 Disabling Hardware Components 82
3.13.3 Enabling Hardware Components 83
3.13.4 Setting Components to Empty 84
3.13.5 Locking Components 84
3.13.6 Unlocking Components 84
3.14 Performing Parallel Operations on Nodes 85
3.15 `xtbounce` Error Message Indicating L1 and L0s Not in Sync 86
3.16 Handling Bus Errors 86
3.17 Handling Component Failures 87
3.18 Capturing and Analyzing System-level and Node-level Dumps 87

Page

3.18.1 Dumping Information Using the `xtdumpsys` Command . . . 87
3.18.2 `ldump` and `lcrash` Utilities for Node Memory Dump and Analysis . . . 88
3.18.3 Using `dumpd` to Automatically Dump and Reboot Nodes . . . 88
3.18.3.1 Enabling `dumpd` . . . 89
3.18.3.2 `/etc/opt/cray-xt-dumpd/dumpd.conf` Configuration File . . . 90
3.18.3.3 Using the `dumpd-dbadmin` Tool . . . 91
3.18.3.4 Using the `dumpd-request` Tool . . . 92
3.19 Using `xtnmi` Command to Collect Debug Information from Hung Nodes . . . 92

Monitoring System Activity [4] **93**

4.1 Monitoring the System with the System Environmental Data Collector (SEDC) . . . 93
4.2 Displaying Installed SMW Release Level . . . 93
4.3 Displaying Installed CLE Release Level . . . 93
4.4 Displaying Boot Configuration Information . . . 93
4.5 Managing Log Files Using CLE and HSS Commands . . . 94
4.5.1 Filtering the Event Log . . . 94
4.5.2 Adding Entries to Log Files . . . 95
4.5.3 Examining Log Files . . . 95
4.5.4 Removing Old Log Files . . . 95
4.6 Managing Log Files Using the Cray Management Services (CMS) Log Manager . . . 97
4.7 Checking the Status of System Components . . . 97
4.8 Checking the Status of Compute Processors . . . 97
4.9 Checking CNL Compute Node Connection . . . 98
4.10 Checking Link Control Block and Router Errors . . . 99
4.11 Monitoring the Status of Jobs Started Under a Third-party Batch System . . . 100
4.12 Using the `cray_pam` Module to Monitor Failed Login Attempts . . . 100
4.13 Monitoring DDN RAID . . . 100
4.14 Monitoring LSI Engenio RAID . . . 100
4.15 Monitoring HSS Managers . . . 100
4.15.1 Examining Activity on the HSS Boot Manager . . . 100
4.15.2 Polling a Response from an HSS Daemon, Manager, or the Event Router . . . 101
4.16 Monitoring Events . . . 101
4.17 Monitoring Node Console Messages . . . 101
4.18 Showing the Component Alert, Warning, and Location History . . . 102
4.19 Displaying Component Information . . . 103
4.20 Displaying Alerts and Warnings . . . 106
4.21 Clearing Flags . . . 106

Page

Managing User Access [5] **107**

5.1 Load Balancing Across Login Nodes . . . 107

5.2 Passwords . . . 107

5.2.1 Changing Default SMW Passwords After Completing Installation . . . 108

5.2.2 Changing `root` and `crayadm` Passwords on Boot and Service Nodes . . . 108

5.2.3 Changing the `root` Password on CNL Compute Nodes . . . 109

5.2.4 Changing the HSS Data Store (MySQL) Password . . . 109

5.2.5 Changing Default MySQL Passwords on the SDB . . . 110

5.2.6 Assigning and Changing User Passwords . . . 113

5.2.7 Logins That Do Not Require Passwords . . . 113

5.3 Administering Accounts . . . 113

5.3.1 Managing Boot Node Accounts . . . 114

5.3.2 Managing User Accounts on Service Nodes . . . 114

5.3.2.1 Adding a User or Group . . . 114

5.3.2.2 Removing a User or Group . . . 115

5.3.2.3 Changing User or Group Information . . . 115

5.3.2.4 Assigning Groups of CNL Compute Nodes to a User Group . . . 115

5.3.2.5 Associating Users with Projects . . . 115

5.3.2.6 Enabling LDAP Support for User Authentication . . . 116

5.3.3 Setting Disk Quotas for a User on the Cray Local, Non-Lustre File System . . . 117

5.4 System-wide Default Modulefiles . . . 118

5.5 User Access to a Compiler Environment Using Modulefiles . . . 118

5.6 Maintaining `*rc.local` Scripts . . . 119

5.7 Using the `pam_listfile` Module in the Shared Root Environment . . . 120

5.8 `ulimit` Stack Size Limit . . . 121

5.9 Stopping a User's Job . . . 121

5.9.1 Stopping a CNL Job Running in Interactive Mode . . . 121

5.9.2 Stopping a Job Running Under a Batch System . . . 121

Modifying an Installed System [6] **123**

6.1 Configuring the Shared-root File System on Service Nodes . . . 123

6.1.1 Specialization . . . 124

6.1.2 Visible Shared-root File System Layout . . . 125

6.1.3 How Specialization Is Implemented . . . 127

6.1.4 Working with the Shared-root File System . . . 128

6.1.4.1 Managing System Configuration With the `xtopview` Tool . . . 129

6.1.4.2 Updating Specialized Files From Within the `xtopview` Shell . . . 131

6.1.4.3 Specializing Files . . . 131

Page

6.1.4.4 Determining which Files are Specialized . . . 133
6.1.4.5 Checking Shared-root Configuration . . . 135
6.1.4.6 Verifying the Coherency of `/etc/init.d` Files Across All Shared Root Views . . . 135
6.1.4.7 Cloning a Shared-root Hierarchy . . . 136
6.1.4.8 Changing the Class of a Node . . . 136
6.1.4.9 Removing Specialization . . . 137
6.1.4.10 Displaying RCS Log Information for Shared Root Files . . . 137
6.1.4.11 Checking Out an RCS Version of Shared Root Files . . . 138
6.1.4.12 Listing Shared Root File Specification and Version Information . . . 139
6.1.4.13 Performing Archive Operations on Shared Root Files . . . 139
6.1.5 Logging Shared-root Activity . . . 140
6.2 PBS Professional Licensing Requirements for Cray Systems . . . 140
6.3 Disabling Secure Shell (SSH) on Compute Nodes . . . 140
6.4 Modifying SSH Keys for Compute Nodes . . . 141
6.5 Configuring the System Environmental Data Collector (SEDC) . . . 143
6.6 Configuring Optional RPMs in the CNL Boot Image . . . 143
6.7 Configuring Memory Control Groups . . . 143
6.8 Configuring Cray Enhanced Linux Security Features . . . 145
6.8.1 Security Auditing and Cray Audit Extensions . . . 145
6.8.1.1 Lustre File System Requirements for Cray Audit . . . 149
6.8.1.2 System Performance Considerations for Cray Audit . . . 151
6.8.2 Using the `cray_pam` PAM to Log Failed Login Attempts . . . 151
6.9 Configuring `cron` Services . . . 155
6.10 Configuring the Load Balancer . . . 158
6.11 Configuring Node Health Checker (NHC) . . . 160
6.11.1 `/etc/opt/cray/nodehealth/nodehealth.conf` Configuration File . . . 160
6.11.2 Configuring Node Health Checker Tests . . . 161
6.11.2.1 Guidance About NHC Tests . . . 164
6.11.2.2 Global Configuration Variables That Affect All NHC Tests . . . 167
6.11.2.3 Standard Variables That Affect Individual NHC Tests . . . 168
6.11.3 Suspect Mode . . . 169
6.11.4 NHC Messages . . . 171
6.11.5 What if a Login Node Crashes While `xtcheckhealth` Binaries are Monitoring Nodes? . . . 172
6.11.6 Disabling NHC . . . 173
6.11.7 `nodehealth` Modulefile . . . 174
6.11.8 Configuring the Node Health Checker to Use SSL . . . 174
6.12 Activating Process Accounting for Service Nodes . . . 176

Page

6.13 Configuring Failover for Boot and SDB Nodes 176
6.13.1 Configuring Boot-node Failover 177
6.13.2 Configuring SDB Node Failover 180
6.13.3 Compute Node Failover Manager 182
6.14 Creating Logical Machines 182
6.14.1 Creating Routable Logical Machines 182
6.14.1.1 Topology Class 0 182
6.14.1.2 Topology Class 1 183
6.14.1.3 Topology Class 2 184
6.14.1.4 Topology Class 3 185
6.14.2 Configuring a Logical Machine 185
6.14.3 Booting a Logical Machine 186
6.15 Updating Boot Configuration 186
6.16 Modifying Boot Automation Files 187
6.17 Callout to `rc.local` During Boot 188
6.18 Changing the System Software Version to be Booted 188
6.18.1 Minor Release Switching within a System Set 189
6.18.2 Major Release Switching using Separate System Sets 190
6.19 Changing the Service Database (SDB) 191
6.19.1 Service Database Tables 191
6.19.2 Database Security 192
6.19.3 Updating Database Tables 192
6.19.3.1 Changing Nodes and Classes 194
6.19.3.2 Changing Services 195
6.20 Viewing the Service Database Contents with MySQL Commands 196
6.21 Configuring the Lustre File System 197
6.22 Configuring Cray Data Virtualization Service (Cray DVS) 197
6.23 Enabling File-locking for Lustre Clients 198
6.24 Setting and Viewing Node Attributes 198
6.24.1 Setting Node Attributes Using the `/etc/opt/cray/sdb/attr.xthwinv.xml` and `/etc/opt/cray/sdb/attr.defaults` Files 198
6.24.1.1 Generating the `/etc/opt/cray/sdb/attributes` File 198
6.24.2 SDB `attributes` Table 200
6.24.3 Setting Attributes Using the `xtprocadmin` Command 201
6.24.4 Viewing Node Attributes 202
6.25 Using the XTAdmin Database `segment` Table 202
6.26 Configuring Networking Services 203
6.26.1 Changing the High-speed Network (HSN) 203

Page

6.26.2 Network File System (NFS) . . . 203
6.26.3 Configuring Ethernet Link Aggregation (Bonding, Channel Bonding) . . . 204
6.26.4 Configuring a Virtual Local Area Network (VLAN) Interface . . . 205
6.26.5 Increasing Size of ARP Tables . . . 206
6.26.6 Configuring Realm-specific IP Addressing (RSIP) . . . 206
6.26.6.1 Using the `CLEinstall` Program to Install and Configure RSIP . . . 207
6.26.7 IP Routes for CNL Nodes in the `/etc/routes` File . . . 211
6.27 Updating the System Configuration After A Hardware Change . . . 211
6.28 Changing the Location to Log `syslog-ng` Information . . . 216

Managing Services [7] **217**

7.1 Configuring the SMW to Synchronize to a Site NTP Server . . . 217
7.2 Synchronizing Time of Day on Compute Node Clocks with the Clock on the Boot Node . . . 217
7.3 Adding and Starting a Service Using Standard Linux Mechanisms . . . 218
7.4 Adding and Starting a Service Using RCA . . . 218
7.4.1 Adding a Service to List of Services Available under RCA . . . 218
7.4.2 Indicating Nodes on Which the Service Will Be Started . . . 219
7.5 Creating a Snapshot of `/var` . . . 219
7.6 Setting Soft and Hard Limits to Prevent Login Node Hangs . . . 220
7.7 Rack-mount SMW: Creating a Cray System Management Workstation (SMW) Bootable Backup Drive 222
7.8 Desk-side SMW: Creating an System Management Workstation (SMW) Bootable Backup Drive . . . 227
7.9 Rack-mount SMW: Setting Up the Bootable Backup Drive as an Alternate Boot Device . . . 233
7.10 Desk-side SMW: Setting Up the Bootable Backup Drive as an Alternate Boot Device . . . 235
7.11 Archiving the SDB . . . 238
7.12 Backing Up Limited Shared-root Configuration Data . . . 238
7.12.1 Using the `xtoparchive` Utility to Archive the Shared-root File System . . . 238
7.12.2 Using Linux Utilities to Save the Shared-root File System . . . 239
7.13 Backing Up Boot Root and Shared Root . . . 240
7.13.1 Using the `xthotbackup` Command to Back Up Boot Root and Shared Root . . . 240
7.13.2 Using `dump` and `restore` Commands to Back Up Boot Root and Shared Root . . . 241
7.14 Backing Up User Data . . . 242
7.15 Rebooting a Stopped SMW . . . 242
7.15.1 SMW Recovery . . . 243
7.16 Recovering from Service Database Failure . . . 243
7.16.1 Database Server Failover . . . 243
7.16.2 Rebuilding Corrupted SDB Tables . . . 244
7.17 Using Persistent SCSI Device Names . . . 244
7.17.1 Using `cray-scscidev-emulation` Device Naming . . . 244

Page

7.18 Using a Linux `iptables` Firewall to Limit Services . . . 245
7.19 Handling Single-node Failures . . . 245
7.20 Increasing the Boot Manager Time-out Value . . . 246
7.21 RAID Failure . . . 246

Using the Application Level Placement Scheduler (ALPS) [8] 247

8.1 ALPS Functionality . . . 247
8.2 ALPS Architecture . . . 248
8.2.1 ALPS Clients . . . 249
8.2.1.1 The `aprun` Client . . . 250
8.2.1.2 The `apstat` Client . . . 250
8.2.1.3 The `apkill` Client . . . 251
8.2.1.4 The `apmgr` Client . . . 251
8.2.1.5 The `apbasil` Client . . . 251
8.2.2 ALPS Daemons . . . 252
8.2.2.1 The `apbridge` Daemon . . . 252
8.2.2.2 The `apsched` Daemon . . . 252
8.2.2.3 The `apsys` Daemon . . . 252
8.2.2.4 The `apwatch` Daemon . . . 253
8.2.2.5 The `apinit` Daemon . . . 253
8.2.2.6 The `apres` Daemon . . . 254
8.2.2.7 ALPS Log Files . . . 254
8.2.2.8 Changing Debug Message Level of `apsched` and `apsys` Daemons . . . 254
8.3 Configuring ALPS . . . 255
8.3.1 `/etc/sysconfig/alps` Configuration File . . . 255
8.3.2 `/etc/alps.conf` Configuration File . . . 259
8.4 Resynchronizing ALPS and the SDB Command After Manually Changing the SDB . . . 262
8.5 Identifying Reserved Resources . . . 263
8.6 Terminating a Batch Job . . . 263
8.7 Setting a Compute Node to Batch or Interactive Mode . . . 264
8.8 Manually Starting and Stopping ALPS Daemons on Service Nodes . . . 264
8.9 Manually Cleaning ALPS and PBS After Downed Login Node . . . 265
8.10 Verifying that ALPS is Communicating with Cray System Compute Nodes . . . 266
8.11 ALPS and Node Health Monitoring Interaction . . . 266
8.11.1 `aprun` Actions . . . 267
8.11.2 `apinit` Actions . . . 268
8.11.3 `apsys` Actions . . . 269
8.11.4 Cleanup Version 1 Actions (`apmgrcleanup`) . . . 270

Page

8.11.5 Cleanup Version 2 Actions . . . 272
8.11.6 Node Health Checker Actions . . . 273
8.11.7 Verifying Application Cleanup . . . 273

Using Comprehensive System Accounting [9] 275

9.1 Interacting with Batch Entry Systems or the PAM `job` Module . . . 276
9.2 CSA Configuration File Values . . . 276
9.3 Configuring CSA . . . 278
9.3.1 Obtaining File System and Node Information . . . 278
9.3.2 Editing the `csa.conf` File . . . 279
9.3.3 Editing Other System Configuration Files . . . 282
9.3.4 Creating a CNL Image with CSA Enabled . . . 283
9.3.5 Setting Up Project Accounting . . . 283
9.3.5.1 Disabling Project Accounting . . . 285
9.3.6 Setting Up Job Accounting . . . 285
9.4 Creating Accounting `cron` Jobs . . . 286
9.4.1 `csanodeacct` `cron` Job for Login Nodes . . . 286
9.4.2 `csarun` `cron` Job . . . 287
9.4.3 `csaperiod` `cron` Job . . . 287
9.5 Enabling CSA . . . 287
9.6 Using LDAP with CSA . . . 288

Using Checkpoint/Restart on Cray Systems (Deferred implementation) [10] 289

10.1 Requirements and/or Limitations for Checkpoint/Restart . . . 289
10.1.1 Using Current Cray MPT Libraries . . . 289
10.1.2 Specifying Batch System Software Releases . . . 289
10.1.3 Setting File System Access Pattern . . . 290
10.1.4 Disabling `mmap` Mechanism . . . 290
10.2 Installation and Configuration . . . 290
10.2.1 Cray Installation and Configuration Options . . . 290
10.2.2 Configuring TORQUE and Moab to Work with CPR . . . 291
10.2.3 Configuring PBS Professional to Work with CPR . . . 292
10.3 Using Checkpoint/Restart . . . 292
10.3.1 Compiling Applications . . . 292
10.3.2 Using Checkpoint/Restart with TORQUE and Moab . . . 293
10.3.2.1 Common Checkpoint/Restart Error Messages . . . 293
10.3.3 Using Checkpoint/Restart with PBS Professional . . . 295

Page

Dynamic Shared Objects and Cluster Compatibility Mode in the Cray Linux Environment [11] **297**

11.1 Configuring the Compute Node Root Runtime Environment (CNRTE) Using `CLEinstall` . . 297

11.2 Configuring Cluster Compatibility Mode 299

11.2.1 Preconditions 301

11.2.2 Configuration Options Relevant to Installation 302

11.2.3 Post-install Options and Configuration 303

OpenFabrics Interconnect Drivers for CLE Systems [12] **309**

12.1 OFED Overview 309

12.2 Using InfiniBand 310

12.2.1 Storage Area Networking 310

12.2.2 Lustre Routing 311

12.2.3 IP Connectivity 312

12.3 Configuration 312

12.4 InfiniBand Configuration 313

12.5 Subnet Manager (OpenSM) Configuration 314

12.5.1 Starting OpenSM at Boot Time 315

12.6 Internet Protocol over InfiniBand (IPoIB) Configuration 315

12.7 Configuring SCSI RDMA Protocol (SRP) on Cray Systems 316

12.8 Lustre Networking (LNET) Router 317

12.8.1 Configuring the LNET Router 317

12.8.2 Configuring the InfiniBand Lustre Server 319

12.8.3 Configuring the LNET Clients 320

Appendix A SMW and CLE System Administration Commands **321**

A.1 HSS Commands 321

A.2 CLE System Administration Commands 323

Appendix B System States **327**

Appendix C Error Codes **329**

Appendix D Remote Access to the SMW **341**

Appendix E Updating the Time Zone **345**

Appendix F Creating Modulefiles **351**

F.1 Modulefile Template 351

F.2 Sharing Your Modulefile 354

F.3 Modulefile Help 354

Page

Appendix G PBS Professional Licensing for Cray Systems **355**

G.1 Introduction 355

G.2 Migrating the PBS Professional Server and Scheduler 356

G.3 Configuring RSIP to the SDB Node 358

G.4 Network Address Translation (NAT) IP Forwarding 361

G.5 Installing and Configuring a NIC 363

Appendix H Installing RPMs **367**

H.1 Generic RPM Usage 367

Appendix I Sample LNET Router Controller Script **369**

Procedures

Procedure 1. Logging on to the boot node 63

Procedure 2. Creating a Cray boot image from existing file system images 65

Procedure 3. Preparing a boot image for CNL compute nodes and service nodes 66

Procedure 4. Manually booting the boot node and service nodes 69

Procedure 5. Booting CNL compute nodes 71

Procedure 6. Shutting down service nodes 74

Procedure 7. Reserving a component 76

Procedure 8. Powering down a specified blade 76

Procedure 9. Halting a node 77

Procedure 10. Power up blades in a cabinet 77

Procedure 11. Disabling a Gemini ASIC 82

Procedure 12. Enabling a Gemini ASIC 83

Procedure 13. Power-cycling a component 86

Procedure 14. Enabling `dumpd` 89

Procedure 15. Showing boot configuration information for the entire system 94

Procedure 16. Showing boot configuration information for a partition of a system 94

Procedure 17. Showing the status of a component 97

Procedure 18. Displaying the location history for component `c0-0c0s0n1` 102

Procedure 19. Changing the `root` and `crayadm` passwords on boot and service nodes 108

Procedure 20. Changing the `root` password on CNL compute nodes 109

Procedure 21. Changing default MySQL passwords on the SDB 110

Procedure 22. Stopping a CNL job running in interactive mode 121

Procedure 23. Specializing a file by class login 132

Procedure 24. Specializing a file by node 132

Procedure 25. Specializing a file by node without entering `xtopview` 133

Procedure 26. Finding files in `/etc` that are specialized by a node 133

Page

Procedure 27. Disabling SSH daemon (`sshd`) on CNL compute nodes 141
Procedure 28. Using `dropbear` to generate site-specific SSH keys 141
Procedure 29. Adjusting the memory control group limit 144
Procedure 30. Disabling memory control groups 145
Procedure 31. Configuring Cray Audit 147
Procedure 32. Configuring `cray_pam` to log failed login attempts 152
Procedure 33. Configuring `cron` for the SMW and the boot node 155
Procedure 34. Configuring `cron` for the shared root with persistent `/var` 156
Procedure 35. Configuring `cron` for the shared root without persistent `/var` 157
Procedure 36. Configuring `lbnamed` on the SMW 159
Procedure 37. Installing the load balancer on an external "white box" server 159
Procedure 38. Recovering from a login node crash when a login node will not be rebooted 172
Procedure 39. Configuring the Node Health Checker (NHC) to use SSL 175
Procedure 40. Configuring boot-node failover 179
Procedure 41. Disabling boot-node failover 180
Procedure 42. Configuring a logical machine 185
Procedure 43. Booting a system set 190
Procedure 44. Updating the `service_config` table when services change 195
Procedure 45. Examining the service databases with MySQL commands 196
Procedure 46. Configuring an I/O service node bonding interface 204
Procedure 47. Configuring a Virtual Local Area Network (VLAN) interface 205
Procedure 48. Installing, configuring, and starting RSIP clients and servers 208
Procedure 49. Adding isolated service nodes as RSIP clients 210
Procedure 50. Updating the system configuration after hardware changes 211
Procedure 51. Updating the system configuration after services nodes were added or removed . . . 214
Procedure 52. Configuring `syslog-ng` system message logs 216
Procedure 53. Configuring the SMW to synchronize to a site NTP server 217
Procedure 54. Adding a service to list of services available under RCA 219
Procedure 55. Preventing login node hangs by setting soft and hard limits 221
Procedure 56. Rack-mount SMW: Creating an SMW bootable backup drive 222
Procedure 57. Desk-side SMW: Creating an SMW bootable backup drive 228
Procedure 58. Rack-mount SMW: Setting up the bootable backup drive as an alternate boot device . . 233
Procedure 59. Desk-side SMW: Setting up the bootable backup drive as an alternate boot device . . . 235
Procedure 60. Backing up limited shared-root configuration data 239
Procedure 61. Backing up the boot root and shared root using the `dump` and `restore` commands . 241
Procedure 62. Rebooting a stopped SMW 242
Procedure 63. SMW primary disk failure recovery 243

Page

Procedure 64. Starting and stopping ALPS daemons on a specific service node 264
Procedure 65. Restarting ALPS daemon on a specific service node 265
Procedure 66. Manually cleaning up ALPS and PBS after a login node goes down 265
Procedure 67. Obtaining file system and node information 278
Procedure 68. Editing CSA parameters for the example system 279
Procedure 69. Setting up CSA project accounting 283
Procedure 70. Disabling project accounting 285
Procedure 71. Setting up CSA job accounting 285
Procedure 72. Using DVS to mount home directories on the compute nodes for CCM 303
Procedure 73. Modifying CCM and Platform-MPI system configurations 304
Procedure 74. Setting up files for the `cnos` class 305
Procedure 75. Linking the CCM prologue/epilogue scripts for use with PBS and Moab TORQUE on login nodes 306
Procedure 76. Using `qmgr` to create a general CCM queue and queues for separate ISV applications . 307
Procedure 77. Configuring Platform LSF for use with CCM 307
Procedure 78. Configuring InfiniBand on service nodes 313
Procedure 79. Starting a single instance of OpenSM on a service node at boot time 315
Procedure 80. Configuring IP Over InfiniBand (IPoIB) on Cray systems 315
Procedure 81. Configuring and enabling SRP on Cray Systems 316
Procedure 82. Configuring the LNET routers 317
Procedure 83. Manually controlling LNET routers 319
Procedure 84. Configuring the InfiniBand Lustre Server 319
Procedure 85. Configuring the LNET clients 320
Procedure 86. Starting the VNC server 341
Procedure 87. For workstation or laptop running Linux: Connecting to the VNC server through an `ssh` tunnel, using the `vncviewer -via` option 342
Procedure 88. For workstation or laptop running Linux: Connecting to the VNC server through an `ssh` tunnel 343
Procedure 89. For workstation or laptop running Mac OS X: Connecting to the VNC server through an `ssh` tunnel 343
Procedure 90. For workstation or laptop running Windows: Connecting to the VNC server through an `ssh` tunnel 344
Procedure 91. Changing the time zone for the SMW and the L1 and L0 controllers 345
Procedure 92. Changing the time zone on the boot root and shared root 347
Procedure 93. Changing the time zone for compute nodes 348
Procedure 94. Migrating PBS off the SDB node 357
Procedure 95. Creating a simple RSIP configuration with the SDB node as a client 359
Procedure 96. Adding the SDB node as an RSIP client to an existing RSIP configuration 360
Procedure 97. Configuring NAT IP forwarding for the SDB node 361

Page

Procedure 98. Installing and configuring a NIC on the SDB node 363

Examples

Example 1. Sample `/etc/opt/cray/sdb/node_classes` file 57
Example 2. Establishing a two-way connection between the SMW and console of service node `c0-0c0s0n1` 63
Example 3. Making a boot image with new parameters for service and CNL compute nodes 69
Example 4. Booting all service nodes with a specific image 72
Example 5. Booting all compute nodes with a specific image 72
Example 6. Booting compute nodes using a load file 72
Example 7. Rebooting a single compute node 72
Example 8. Rebooting login or network nodes 73
Example 9. Displaying routing information 73
Example 10. Routing the entire system 73
Example 11. Shutting down the system using the `auto.xtshutdown` file 74
Example 12. Shutting down all compute nodes 75
Example 13. Shutting down specified compute nodes 75
Example 14. Shutting down all nodes of a system 75
Example 15. Forcing nodes to shut down 75
Example 16. Aborting a session running on the boot manager 78
Example 17. Looking at node characteristics 79
Example 18. Viewing all node attributes 79
Example 19. Viewing selected node attributes of selected nodes 79
Example 20. Disabling a node 80
Example 21. Disabling all processors 80
Example 22. Finding the physical ID for node 38 81
Example 23. Finding the physical ID for nodes 0, 1, 2, and 3 81
Example 24. Finding the physical IDs for Gemini IDs 0-7 81
Example 25. Printing the *nid*-to-*nic* address mappings for the node with NID `31`. 81
Example 26. Printing the *nid*-to-*nic* address mappings for the same node as shown in Example 25, but specifying the NIC value in the command line 81
Example 27. Creating a list of node identifiers that are not in the `DISABLE`, `EMPTY`, or `OFF` state . 82
Example 28. Disabling the Gemini ASIC `c0-0c1s3g0` 82
Example 29. Enabling Gemini ASIC `c0-0c1s3g0` 83
Example 30. Setting a blade to the `empty` state 84
Example 31. Locking cabinet `c0-0` 84
Example 32. Show all session (lock) data 84
Example 33. Unlocking cabinet `c0-0` 85
Example 34. Restarting the NTP service 85

Page

Example 35. Dumping information about a working component 87
Example 36. Displaying installed SMW release level 93
Example 37. Displaying installed CLE release level 93
Example 38. Finding information in the event log 94
Example 39. Adding entries to `syslog` file 95
Example 40. Identifying nodes in `down` or `admindown` state 97
Example 41. Display current allocation and status of each compute processing element and the application that it is running 98
Example 42. Verifying that a compute node is connected to the network 99
Example 43. Running `xtnetwatch` to monitor the system interconnection network 99
Example 44. Looking at a session running on the boot manager 101
Example 45. Checking the boot manager 101
Example 46. Monitoring for specific events 101
Example 47. Checking events except heartbeat: 101
Example 48. Obtaining node console messages 102
Example 49. Identifying all service nodes 103
Example 50. Showing compute nodes in the `disabled` state 105
Example 51. Adding a group 114
Example 52. Adding a user account 114
Example 53. Removing a user account 115
Example 54. Creating a `pam_listfile` list file 120
Example 55. Adding a line to `/etc/pam.d/sshd` to enable `pam_listfile` 120
Example 56. Stopping a job running under PBS Professional 121
Example 57. Shared-root links 127
Example 58. Starting the `xtopview` shell for a node 130
Example 59. Starting the `xtopview` shell for a class of nodes 130
Example 60. Starting the `xtopview` shell for a directory other than `/rr/current` 130
Example 61. Sample `xtopview` session 130
Example 62. Starting `xtopview` using `node_classes` for information 131
Example 63. Updating a file within `xtopview` shell 131
Example 64. Finding files in `/etc` that are specialized by class 134
Example 65. Finding specialization of a file on a node 134
Example 66. Finding nodes on which a file is specialized 134
Example 67. Finding specialization of a file on a node without invoking the `xtopview` shell . . . 134
Example 68. Finding specialization of files by class without invoking the `xtopview` shell 135
Example 69. Finding the class of a node 136
Example 70. Adding a node to a class 136
Example 71. Removing node specialization 137

Page

Example 72. Removing class specialization . . . 137
Example 73. Printing the latest version of a file . . . 138
Example 74. Printing the RCS log for `/etc/fstab` in the node 3 view . . . 138
Example 75. Displaying differences between two versions of the `/etc/fstab` file . . . 138
Example 76. Checking out a version 1.2 copy of `/etc/fstab` . . . 138
Example 77. Recreating the file link for `/etc/fstab` to the current view's `/etc/fstab` file . . . 138
Example 78. Printing specifications for login class specialized files . . . 139
Example 79. Printing specifications for files modified in the default view and include any warning messages . . . 139
Example 80. Adding files specified by specifications listed in `specfile` to an archive file . . . 140
Example 81. Listing specifications for files currently in the `archive.20110422` archive file . . . 140
Example 82. Default `/etc/auditd.conf` file . . . 150
Example 83. Modified PAM configuration files configured to report failed login by using an alternate path . . . 154
Example 84. Creating a logical machine with a boot node and SDB node specifying the boot image path 185
Example 85. Updating boot configuration . . . 186
Example 86. Identifying services in the `service_config` table . . . 195
Example 87. Sample mount line from compute node `/etc/fstab` . . . 198
Example 88. Using node attribute labels to assign nodes to user groups . . . 200
Example 89. Adding the `PBS-MOM` service for a specific node . . . 219
Example 90. Force the `fomd` to update its configuration information about a new or updated service on a node . . . 219
Example 91. Effect a change for a new or updated service on a group of nodes . . . 219
Example 92. Using the `xtoparchive` utility to archive the shared-root file system . . . 239
Example 93. Using the `xthotbackup` command to create a bootable backup system set . . . 240
Example 94. Using the `xthotbackup` command to copy selected file systems from source to the backup system set . . . 241
Example 95. Recovering from an SDB failure . . . 243
Example 96. Increasing the `boot_timeout` value . . . 246
Example 97. Sample `/etc/sysconfig/alps` configuration file . . . 258
Example 98. Sample `/etc/alps.conf` configuration file . . . 262
Example 99. Retrieving node allocation status . . . 264
Example 100. Verifying that ALPS is communicating with Cray system compute nodes . . . 266
Example 101. Running a `csanodeacct` `cron` job on each login node to move local accounting files 286
Example 102. Executing the `csarun` script . . . 287
Example 103. Running periodic accounting at different intervals than the regular system accounting interval . . . 287
Example 104. Submit a job to TORQUE . . . 293
Example 105. Submit a job to TORQUE that checkpoints every 30 minutes . . . 293

Page

Example 106. Checkpoint and terminate a job using TORQUE 293
Example 107. Restart a held job using TORQUE 293
Example 108. Restart a checkpointed job using TORQUE 293
Example 109. Submit a job to PBS Professional 295
Example 110. Submit a job to PBS Professional that checkpoints every 3 minutes of CPU time . . . 295
Example 111. Checkpoint and terminate a job using PBS Professional 295
Example 112. Restart a held job using PBS Professional 295
Example 113. Restart a checkpointed job using PBS Professional 295
Example 114. Location of queue configuration files 307
Example 115. Modulefile example 353
Example 116. Installing an RPM on the SMW 368
Example 117. Installing an RPM on the boot node root 368
Example 118. Installing an RPM on the shared root 368

Figures

Figure 1. Administrative Components of a Cray System 31
Figure 2. Types of Specialization 123
Figure 3. Shared-root Implementation 126
Figure 4. ALPS Process 249
Figure 5. Cray System Job Distribution Cross-section 300
Figure 6. CCM Job Flow Diagram 301
Figure 7. The OFED Stack (source: OpenFabrics Alliance) 310
Figure 8. Cray System Connected to Storage Using SRP 311
Figure 9. Cray Service Node Acting as an Infiniband Lustre Router 311
Figure 10. Cray Service Node in IP over IB Configuration 312

Tables

Table 1. Physical ID Naming Conventions 53
Table 2. File Specialization by Class 124
Table 3. File Specialization by Node 125
Table 4. Shared-root Commands 128
Table 5. Topology 0 Chassis Layout 183
Table 6. Service Database Tables 191
Table 7. Database Privileges 192
Table 8. Service Database Update Commands 192
Table 9. CSA Parameters That Must Be Specific to Your System 277
Table 10. Project Accounting Parameters That Must Be Specific to Your System 284
Table 11. BLCR Reported Checkpoint Error Messages 294

Page

Table 12. Checkpoint/Restart Error Messages 294
Table 13. Hardware Error Messages 295
Table 14. Upper Layer InfiniBand I/O Protocols for Cray Systems 313
Table 15. LNET Network Address Configuration for Cray XE Systems 317
Table 16. HSS Commands 321
Table 17. CLE Commands 324
Table 18. State Definitions 327
Table 19. Additional State Definitions 327
Table 20. `xtcli` Commands and Allowed States 328
Table 21. System Error Codes 329

Introduction [1]

Note: In this guide, references to Cray systems mean Cray XE systems unless otherwise noted.

The release documents provided with your Cray Linux Environment (CLE) operating system and Cray System Management Workstation (SMW) release packages state the specific Cray platforms supported with each release package.

A Cray system is a massively parallel processing (MPP) system that has a shared-root file system available to all service-processing elements nodes). Cray has combined commodity and open-source components with custom-designed components to create a system that can operate efficiently at immense scale.

The Cray Linux Environment (CLE) operating system includes Cray's customized version of the SUSE Linux Enterprise Server (SLES) 11 Service Pack 1 (SP1) operating system, with a Linux 2.6.32.36 kernel. This full-featured operating system runs on the Cray system's service nodes. Service nodes perform the functions needed to support users, administrators, and applications running on compute nodes. Above the operating system level are specialized daemons and applications that perform functions unique to each service node.

Compute nodes on Cray XE systems run the *CNL* compute node operating system, which runs a Linux 2.6.32.36 kernel. The kernel provides support for application execution without the overhead of a full operating-system image. The kernel interacts with an application process in very limited ways. It provides virtual memory addressing and physical memory allocation, memory protection, access to the message-passing layer, and a scalable job loader. Support for I/O operations is limited inside the compute node's kernel. For a more complete description, see Compute Nodes on page 42.

Note: Functionality marked as deferred in this documentation is planned to be implemented in a later release.

1.1 Audience for This Guide

The audience for this guide is system administrators and those who manage the operation of a Cray system. Prerequisites for using this guide include a working knowledge of Linux to administer the system and a review of the Cray system administration documentation listed in Cray System Administration Publications and in Related Publications on page 29, of this guide. This guide assumes that you have a basic understanding of your Cray system and the software that runs on it.

1.2 Cray System Administration Publications

This publication is one of a set of related manuals that cover information about the structure and operation of your Cray system. See also:

- *Cray System Management Workstation (SMW) Software Release Overview* (S–2482)
- *Installing Cray System Management Workstation (SMW) Software* (S–2480)
- *Cray System Management Workstation (SMW) Software Release Errata*
- *Cray Linux Environment (CLE) Software Release Overview* (S–2425)
- *Cray Linux Environment (CLE) Software Release Overview Supplement* (S–2497)
- *Installing and Configuring Cray Linux Environment (CLE) Software* (S–2444)
- *Limitations for the CLE Release*
- *CLE Release Errata*
- *Using Cray Management Services (CMS)* (S–2484)
- *Using and Configuring System Environment Data Collections (SEDC)* (S–2491)
- *Managing Lustre for the Cray Linux Environment (CLE)* (S–0010)
- *Repurposing Compute Nodes as Service Nodes on Cray XE and Cray XT Systems* (S–0029)
- *Introduction to Cray Data Virtualization Service* (S–0005)
- *Cray Application Developer's Environment Installation Guide* (S–2465)
- *Cray Application Developer's Environment Supplement Installation Guide* (S–2485)
- *Workload Management and Application Placement for the Cray Linux Environment* (S–2496)
- *Writing a Node Health Checker (NHC) Plugin Test* (S–0023)

1.3 Related Publications

Because your Cray system runs a combination of software developed by Cray, other vendors' software, and open-source software, the following websites may be useful:

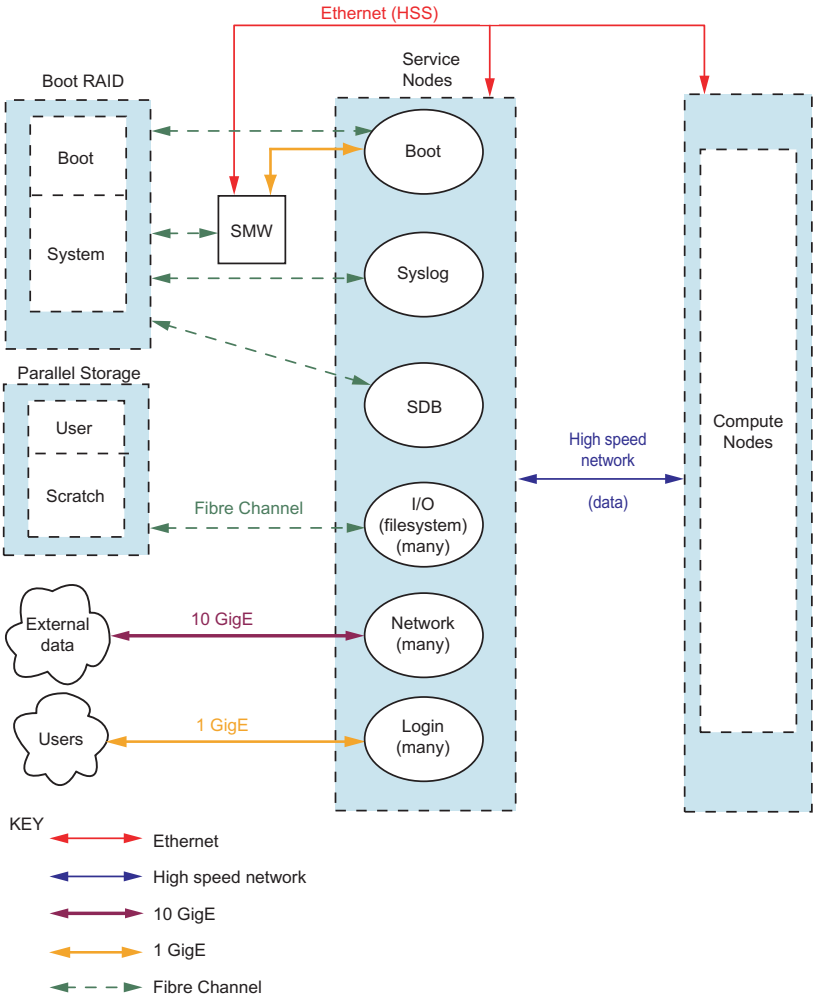
- Linux Documentation Project — See http://www.tldp.org
- SLES 11 and Linux documentation — See http://www.novell.com/linux
- Data Direct Networks documentation — See http://www.ddn.com/support/product-downloads-and-documentation
- LSI Engenio storage system documentation — See http://www.lsi.com/storage_home/products_home/external_raid/index.html
- MySQL documentation — See http://www.mysql.com/documentation/
- Lustre File System documentation — See http://www.lustre.org and http://www.sun.com/software/products/lustre/
- Batch system documentation:

PBS Professional:	Altair Engineering, Inc.	http://www.pbsworks.com
Moab and TORQUE:	Adaptive Computing Enterprises Inc.	http://www.adaptivecomputing.com
Platform LSF:	Platform Computing Corporation	http://www.platform.com

Introducing System Components [2]

Cray systems separate calculation and monitoring functions. Figure 1 shows the components of a Cray system that an administrator manages.

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. (For additional information about the SMW, see System Management Workstation (SMW) on page 33.)
- The Hardware Supervisory System (HSS) monitors the system and handles component failures. The 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. (For additional information about HSS, see Hardware Supervisory System (HSS) on page 45.)
- Cray Management Services (CMS) provides the infrastructure to the Application Level Placement Scheduler (ALPS) for a fast cache of node attributes, reservations, and claims. ALPS reservation and claim information is forwarded to CMS on the SMW to enable system administrators to search the history of jobs, error messages which occurred during a job, and the job utilization on the system. (For additional information about CMS, see *Using Cray Management Services (CMS)*, S–2484.)
- The Cray Linux Environment (CLE) operating system is the operating system for Cray systems. (For additional information about CLE, see CLE on page 34.)
- Service nodes perform the management functions that enable the computations to occur. (For additional information about service nodes, see Service Nodes on page 35.)
- Compute nodes are primarily dedicated to computation. (For additional information about compute nodes, see Compute Nodes on page 42.)
- RAID is partitioned for a variety of storage functions such as boot RAID, database storage, and parallel and user-file system storage. (For additional information about RAID, see Boot Root File System on page 35 and Storage on page 51.)

A Cray system has six 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 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 disk farm 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.

2.1 System Management Workstation (SMW)

The SMW is the 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 Service Pack 1 (SLES 11 SP1) operating system, Cray developed software, and third-party software. The SMW is also a single point of control for the HSS. The HSS data is stored on an internal hard drive of the SMW. For more information about the HSS, see Hardware Supervisory System (HSS) on page 45. CMS provides the infrastructure to ALPS for a fast cache of node attributes, reservations, and claims. For additional information about CMS, see *Using Cray Management Services (CMS)* (S–2484).

SMW software installation consists of the `SMWinstall`, `SMWconfig`, and `SMWinstallCLE` system administrator commands. These SMW installation commands automate the SMW software installation process significantly. 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 process ID (PID) of the `SMWinstall` or the `SMWinstallCLE` command; the exact names are displayed when the command is invoked. Using the `SMWconfig` and `SMWinstall` commands, you can specify the SMW configuration file that you want to use for the installation. The `SMWinstall.conf` file contains information about the type of Cray system, the days-to-keep for logging, the Network Time Protocol (NTP) server names, and variables for separate log and database devices in rack-mount SMWs. This information is required for initial installations and is otherwise detected or reused for upgrades and updates of the SMW. An example `SMWinstall.conf` file is included on the SMW installation media, but a customized `SMWinstall.conf` file is expected to be located by default in `/home/crayadm/`. For additional information, see the `SMWinstall`(8), `SMWconfig`(8), and `SMWinstallCLE`(8) man pages.

You log on to an SMW window on the console to perform SMW functions. From the SMW, you can log on to a disk controller or use a web-based interface from the SMW to configure a RAID controller or Fibre Channel switch. You can log on to the boot node from the SMW as well. From the SMW, you cannot 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, you can initiate the boot process, access the database that keeps track of system hardware, and perform standard administrative tasks.

2.2 CLE

CLE is the operating system for Cray systems. CLE is the Cray customized version of the SLES 11 SP1 operating system with a Linux 2.6.32.36 kernel. This full-featured operating system runs on the Cray service nodes. CNL 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, 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.

For a complete list of Cray developed CLE administrator commands, see Appendix A, SMW and CLE System Administration Commands on page 321.

2.3 Boot Root File System

The boot node has its own root file system, `bootroot`, which is created on the boot RAID during installation. You install and configure the boot RAID from the SMW before you boot the boot node. The boot node mounts the `bootroot` from the boot RAID.

2.4 Shared Root File System

A Cray system has 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, so if you modify the single copy, it affects all service nodes. This makes the administration process similar to that of a single system.

You manage the shared-root file system from the boot node through the `xtopview` command (see Managing System Configuration With the `xtopview` Tool on page 129).

If you need unique files on a specific node or class of nodes (that is, nodes of a certain type), you can set up a modified directory structure. This process, called *specialization*, creates a new directory hierarchy that overlays the existing root directory on the specified nodes and contains symbolic links that point to the unique files. For information about the shared root and file specialization, see Configuring the Shared-root File System on Service Nodes on page 123.

2.5 Service Nodes

Service nodes can be specialized and categorized by the services that run on them:

- Boot node
- SDB node
- Syslog node
- Login nodes
- Network nodes
- I/O nodes

Service nodes run the CLE operating system. The administrator commands for these nodes are standard Linux commands and Cray system-specific commands.

Documentation may use the terms *SIO node*, *XIO node* (Service and I/O node with Gemini application-specific integrated circuits (ASICs), or I/O node as a generic reference to the SDB and I/O nodes.

You log on to the boot node through the SMW console, then from the boot node you can log on to the other service nodes.

Service nodes perform the functions needed to support users, administrators, and applications running on compute nodes. As the system administrator, you define service node classes by the service they perform. 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 SDB node.

You can start services system-wide or on specific nodes. You can start services during the boot process or later on specific nodes of a running system. How you start a service 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.

A full-featured operating system runs on the service nodes. Service nodes run a version of Linux. 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. For more information about Cray system commands, see Appendix A, SMW and CLE System Administration Commands on page 321. After the system is up, you can access any service node from any other service node, provided you have the correct permissions.

2.5.1 Boot Node

Use the boot node 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.

The boot node is the first node to be booted, which is done through the boot node's blade control processor (L0 controller) (see Blade Control Processor (L0 Controller) and Cabinet Control Processors (L1 Controller) on page 46). You can bring up an `xterm` window on the SMW to log on to the boot node.

Note: 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.

You can configure two boot nodes 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 Configuring Boot-node Failover on page 177.

2.5.2 Service Database (SDB) Node

The SDB node hosts the service database (SDB), which is a MySQL database that resides on a separate file system on the boot RAID. The SDB is accessible to every service node (see Changing the Service Database (SDB) on page 191). The SDB provides a central location for storing information so that it does not need to be stored on each node. You can access the SDB from any service node after the system is booted, provided you have the correct authorizations.

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 CNL. For more information about ALPS, see Application Level Placement Scheduler (ALPS) for Compute Nodes on page 40.
- System configuration tables that list and describe processor and service information.

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

You can configure two SDB nodes 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 Configuring SDB Node Failover on page 180.

2.5.3 Syslog Node

The syslog node connects to the HSN. A syslog daemon, `syslog-ng`, runs on all service nodes and directs log file information to the `syslog-ng` on the syslog node. The syslog data from the `syslog-ng` daemons on the boot node and the syslog node (commonly the SDB node) is forwarded to the CMS log manager daemon on the SMW. The CMS log manager aggregates the logs to enable system-wide searches and association of events and log messages. You can modify the `/etc/syslog-ng/syslog-ng-conf.in` file to change where the log information is saved. For more information, see Changing the Location to Log `syslog-ng` Information on page 216.

The syslog services may be run on a dedicated syslog node, on the same node as the SDB node, or on the boot node.

2.5.4 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 the compute nodes.

You can use the Linux `lbnamed` load balancer software provided to distribute user logins across login nodes (see Configuring the Load Balancer on page 158). 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.

2.5.5 Network Nodes

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

2.5.6 I/O Nodes

I/O nodes host the Lustre file system; see Lustre File System on page 39. The I/O nodes connect to the RAID subsystems that contain the Lustre file system. Two I/O nodes connect to each RAID device for resiliency; each I/O node has full accessibility to all storage on the connected RAID device. Cray provides support for RAID subsystems from two different vendors, Data Direct Networks (DDN) and LSI Logic Corporation.

Cray Data Virtualization Service (Cray DVS) servers run on an I/O node; see Cray Data Virtualization Service (Cray DVS) on page 39. DVS servers cannot run on the same I/O nodes as Lustre servers. On I/O nodes, DVS servers act as external file system clients. DVS will project the external file systems to service and compute node clients within the system.

2.5.7 Services on the Service Nodes

Service nodes provide the services described in this section.

2.5.7.1 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 `service_config` table of the SDB maintains a list of services that RCA starts. For the services listed in the `service_config` table, the RCA daemon (`rcad_svcs`) starts and restarts all services that must run on a node. You can determine or modify services available through the SDB `service_config` table by using the `xtservconfig` command. For additional information about using this command, see Changing Services on page 195.

Note: Services can also be started manually or automatically by using standard Linux mechanisms (see Adding and Starting a Service Using Standard Linux Mechanisms on page 218).

The SDB `serv_cmd` table stores information about each service, such as, service type, service instance, heartbeat interval, and restart policy.

The configuration file for service nodes is `/etc/opt/cray/rca/rcad_svcs.service.conf`. By default, this configuration file starts the `rca_dispatcher` and the failover manager.

The RCA consists of a kernel-mode driver and a user-mode daemon on CLE. The Cray 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 (see Blade Control Processor (L0 Controller) and Cabinet Control Processors (L1 Controller) on page 46) of any programs that have registered with it and to handle event traffic between the HSS and the applications that register to receive events. The RCA driver starts as part of the kernel boot, and the RCA daemon starts as part of the initialization scripts.

2.5.7.2 Lustre File System

Cray systems running CLE support the Lustre file system that provides a high-performance, highly scalable, POSIX-compliant shared file system. You can configure Lustre file systems 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 nodes.

For more information, see *Managing Lustre for the Cray Linux Environment (CLE)* (S–0010) and *Installing and Configuring Cray Linux Environment (CLE) Software* (S–2444).

2.5.7.3 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 the *Installing and Configuring Cray Linux Environment (CLE) Software* (S–2444) and *Introduction to Cray Data Virtualization Service* (S–0005).

2.5.7.4 Application Level Placement Scheduler (ALPS) for Compute Nodes

For compute nodes running CNL, the Application Level Placement Scheduler (ALPS) is provided. ALPS provides application placement, launch, and management functionality and cooperates with third-party batch systems for application scheduling. The 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.

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.

Note: Only one application can be placed per node; two different executables cannot be run on the same node at the same time.

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

When a job is running on CNL compute nodes, the `aprun` process (see Job Launch Commands on page 43) interacts with ALPS to keep track of the processors that the job uses.

For more information about ALPS, see Chapter 8, Using the Application Level Placement Scheduler (ALPS) on page 247.

2.5.7.5 Cluster Compatibility Mode

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 Chapter 11, Dynamic Shared Objects and Cluster Compatibility Mode in the Cray Linux Environment on page 297.

2.5.7.6 Repurposing CNL Compute Nodes as Service Nodes

Some services on Cray systems have resource requirements or limitations (for example, memory, processing power or response time) that you 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, you can configure the service to run on a single, dedicated compute node and avoid using traditional service I/O node hardware.

When you configure a node on a compute blade 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 *Repurposing Compute Nodes as Service Nodes on Cray XE and Cray XT Systems*.

2.5.7.7 IP Implementation

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

Note: 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.

2.6 Compute Nodes

Cray XE 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 distribution, with a Linux 2.6.32.36 kernel and with 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.

Several libraries and compilers are linked at the user level to support I/O and communication service. PGI, PathScale, and the GNU Compiler Collection (GCC) C, C++, and Fortran 90 compilers are supported. Applications statically link to these libraries. Users can set their desired compiler target architecture environment by loading the `xtpe-target-cnl` modulefile. For information about using modulefiles, see User Access to a Compiler Environment Using Modulefiles on page 118. 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) daemon, `rcad-svcs`, handles node services (see Services on the Service Nodes on page 38).

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

The following user-level BusyBox commands are functional on CNL compute nodes: `ash`, `BusyBox`, `cat`, `chmod`, `chown`, `cp`, `cpio`, `free`, `grep`, `gunzip`, `kill`, `killall`, `ln`, `ls`, `mkdir`, `mktemp`, `more`, `ps`, `rm`, `sh`, `tail`, `test`, `vi`, and `zcat`. For information about supported command 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.

The `CLEinstall` program creates `/var/opt/cray/install/shell_bootimage_LABEL.sh` which uses the `xtclone` and `xtpackage` utilities on the SMW. Use these commands to set up boot images. You can boot CNL on compute nodes. For more information, see Preparing a Service Node and Compute Node Boot Image on page 64, the `xtclone`(8), `xtpackage`(8), and `xtnodestat`(8) man pages, and the *Installing and Configuring Cray Linux Environment (CLE) Software* (S–2444).

2.7 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 `cnselect` command to specify node-selection criteria. The `cnselect` script uses these selection criteria to query the table of node attributes in the SDB; then it 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 Chapter 8, Using the Application Level Placement Scheduler (ALPS) on page 247.

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

2.8 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 for you to change any installation configuration

parameters or issue any commands. To configure NHC tests and to optionally configure NHC to use the secure sockets layer (SSL) protocol, see Configuring Node Health Checker (NHC) on page 160.

2.9 Comprehensive System Accounting (CSA)

Comprehensive System Accounting (CSA) is open-source software that includes changes to the Linux kernel so that 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, is included with CLE and runs on login nodes and CNL compute nodes only. For more information, see Chapter 9, Using Comprehensive System Accounting on page 275.

2.10 Checkpoint/Restart (CPR)

Checkpoint/Restart (CPR) provides a way to stop applications at specified points and later restart them from that point. The CLE CPR feature is built upon the Berkeley Lab Checkpoint/Restart (BLCR) for Linux. Specific third-party batch system software releases are required for checkpoint/restart support (see Optional Workload-management (Batch) System Software Products on page 44). For detailed information about the CLE implementation of CPR, see Chapter 10, Using Checkpoint/Restart on Cray Systems (Deferred implementation) on page 289.

2.11 Optional Workload-management (Batch) System Software Products

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

PBS Professional:	Altair Engineering, Inc.	http://www.pbsworks.com
Moab and TORQUE:	Cluster Resources, Inc.	http://www.clusterresources.com
Platform LSF:	Platform Computing Corporation	http://www.platform.com

Note: Specific third-party batch system software releases are required for Checkpoint/Restart (CPR) support. For more information, access the **3rd Party Batch SW** link on the CrayPort website at http://www.crayport.cray.com.

2.12 Hardware Supervisory System (HSS)

The 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. The HSS communicates with nodes and with the management processors over an internal (private) Ethernet network that operates independently of the system interconnection network. The HSS data is stored on an internal hard drive of the SMW.

For a complete list of Cray developed HSS commands, see Appendix A, SMW and CLE System Administration Commands on page 321.

The HSS includes the following components:

- The HSS network (see HSS Network on page 45).
- The HSS interface (see HSS Interface on page 46).
- Blade and cabinet control processors (L0 and L1 controllers) (see Blade Control Processor (L0 Controller) and Cabinet Control Processors (L1 Controller) on page 46).
- Network Time Protocol (NTP) server (see NTP Server on page 47).
- Event router (see Event Router on page 47).
- HSS managers (see HSS Managers on page 47).
- `xtdiscover` command (see Automatically Discover and Configure Cray System Hardware on page 50).
- Various logs (see Event Logs on page 51, Boot Logs on page 51, Dump Logs on page 51).

2.12.1 HSS Network

The SMW, with its HSS Ethernet network, performs reliability, accessibility, and serviceability tasks. The HSS commands monitor and control the physical aspects of the system.

The SMW manages the HSS network. A series of Ethernet switches connects the SMW to all the cabinets in the system.

2.12.2 HSS Interface

The HSS has a command-line interface to manage and monitor your system. You can use the command-line interface to manage your Cray system from the SMW. For usage information, see Chapter 3, Managing the System on page 63 and Chapter 4, Monitoring System Activity on page 93. For a list of all HSS system administration commands, see Appendix A, SMW and CLE System Administration Commands on page 321.

2.12.3 Blade Control Processor (L0 Controller) and Cabinet Control Processors (L1 Controller)

A blade control processor *(L0 controller)* is hierarchically the lowest component of the monitoring system. One L0 controller resides on each compute blade and service blade, monitoring only the nodes and Cray Gemini chips. It provides access to status and control registers for the components of the blade. The L0 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 L0 controller.

Note: In some contexts, the L0 controller is referred to as a *slot*.

Each cabinet has a cabinet control processor (L1 controller) that monitors and controls the cabinet power and cooling equipment and communicates with all the L0 controllers in the cabinet. It sends a periodic heartbeat to the SMW to indicate cabinet health.

The L1 controller connects to the chassis controller and in turn the chassis controller connects to the L0s (via the backplane) on each blade by Ethernet cable and routes HSS data to and from the SMW. The L1 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 (see State Manager on page 48) sets an alert flag for the component (L0 or L1 controller) that spawned it.

The L1 and L0 controllers use `ntpclient` to keep accurate time with the SMW.

You can dynamically configure the L1 system daemon and the L0 system daemon with the `xtdaemonconfig --`*daemon_name* command (see the `xtdaemonconfig`(8) man page for detailed information).

Note: There is no NV write protection feature on the L1 and L0 controllers; you should not assume the write protection functionality on the L1 front panel display will protect the NV memory on the L1 and L0 controllers.

2.12.4 NTP Server

The SMW workstation is the primary NTP server for the Cray system. The L0 controllers use the HSS network to update themselves according to the NTP protocol. To change the NTP server, see Configuring the SMW to Synchronize to a Site NTP Server on page 217.

2.12.5 Event Router

HSS functions are event-driven. The event router daemon, `erd`, is the root of the HSS. It is a system daemon that runs on the SMW, L1 controllers, and L0 controllers. The SMW runs a separate thread for each L1. The L1 runs a separate thread for each L0. HSS managers subscribe to events and inject events into the HSS system by using the services of the `erd`. (For descriptions of HSS managers, see HSS Managers on page 47) The event router starts as each of the devices (SMW, L1, L0) 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 (see Monitoring Events on page 101) monitors the `erd`. The `xtconsole` command (see Monitoring Node Console Messages on page 101) operates a shell window that displays all node console messages.

2.12.6 HSS Managers

HSS managers are located in `/opt/cray/hss/default/etc`. They report to the event router and get information from it. HSS has the following managers:

- state manager
- boot manager
- system environmental data collections (SEDC) manager
- flash manager
- NID manager

The HSS managers are started by running the `/etc/init.d/rsms start` command.

You can configure HSS daemons dynamically by executing the `xtdaemonconfig` command. For further information, see the `xtdaemonconfig`(8) man page.

2.12.6.1 State Manager

Every component has a state at all times. The state manager, `state_manager`, runs on the SMW and maintains the state of the components in the HSS database. The state manager performs the following functions:

- Updates and maintains component state information (see Appendix B, System States on page 327)
- Monitors events to update component states
- Detects and handles state notification upon failure
- Provides state and configuration information to HSS applications so that they do not interfere 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.

2.12.6.2 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.

2.12.6.3 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, you can also create sets of different configurations as groups so that the L0/L1 daemons can scan components at different frequencies. The `sedc_manager` sends out the scanning configuration for specific groups to the L1s and L0s and records the incoming data by group. For information about configuring the SEDC manager, see *Using and Configuring System Environment Data Collections (SEDC)* and the `sedc_manager`(8) man page.

To view System Environment Data Collections (SEDC) scan data, use the `xtsedcviewer` command-line interface. This utility allows you to view the server configurations (groups) as well as the SEDC scan data from L0 and L1 controllers. For information about viewing SEDC server configuration and the SEDC scan data, see *Using and Configuring System Environment Data Collections (SEDC)* and the `xtsedcviewer`(8) man page.

2.12.6.4 Flash Manager

The flash manager, `fm`, runs on the SMW. The `fm` command is intended for use by Cray Service Personnel only; improper use of this restricted command can cause serious damage to your computer system. `fm` 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.

Note: 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.

2.12.6.5 NID Manager

The NID (node ID) 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 a NID assignment file, which is `/etc/opt/cray/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 could make the Cray system unrouteable.

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 Updating the System Configuration After A Hardware Change on page 211). This may require the NID mapping to change, depending on the nature of the configuration change. Adding additional cabinets to the ends of rows does not typically result in a new mapping. Adding additional rows most likely does result in a new mapping. If the configuration change is such that the topology class of the system is changed, this will require a new NID mapping. 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 L0s. The L0s, in turn, forward the mappings to the RCA on each node. Because the operating system always uses node IDs (NIDs), the HSS converts these to NIC IDs when sending them onto the HSS network and back to NIDs when forwarding events from the HSS network to a node.

For more information about node IDs, see Identifying Components on page 52 and Identifying Components on page 52.

2.12.7 Automatically Discover and Configure Cray System Hardware

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 can correctly identify 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. When it has finished, you 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, and you can modify the 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 allows you 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 on Cray XE systems.

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.

The state manager 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. In addition, the dynamic system state persists between boots.

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 (for example, a service blade that is 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.

2.12.8 Cray System Network Routing Utility

Use the `rtr` command to perform a variety of routing-related tasks. The `rtr` command is also invoked as part of the `xtbootsys` process. For more information, see the `rtr`(8) man page.

2.12.9 Event Logs

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

2.12.10 Boot Logs

The `/var/opt/cray/log/bootlogs` directory is a repository for files collected by commands such as `xtbootsys`, `xtconsole`, `xtconsumer`, and `xtnetwatch`.

For more information about examining log files, see Managing Log Files Using CLE and HSS Commands on page 94.

2.12.11 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 about examining log files, see Managing Log Files Using CLE and HSS Commands on page 94.

2.13 Cray Management Services (CMS)

The CMS is the administrative framework that integrates both hardware environments to provide monitoring of and administration functionality for Cray XE systems. CMS software provides tools for tasks such as collecting log information, managing system resources, and coordinating miscellaneous information from all parts of the system.

For detailed information, see *Using Cray Management Services (CMS)* (S–2484).

2.14 Storage

The Cray system RAID storage is a disk farm that supports high bandwidth and shared access to and backup of large volumes of data.

Every independent Fibre Channel host interface in each controller provides full-speed access to all the disk storage on its RAID device. Each tier is configured with a parity disk. There are redundant controllers for each RAID.

The RAID device is 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.

Boot RAID is partitioned for boot and system (database) functions.

Parallel storage contains user partitions and scratch partitions.

Common storage vendors for a Cray system are DDN devices from DataDirect Networks and LSI devices from LSI Logic Corporation.

CLE supports the capability to configure multiple I/O paths to the controllers on a disk array. One path is designated as the active primary path and the remaining paths are considered inactive or alternate paths. When the primary path to the array is lost due to a failure, disk-specific multi-pathing functionality automatically switches the data access to an alternate path. For Data Direct Networks (DDN) devices, multi-pathing functionality is provided using Device Mapper (DM) functionality that is included in the Linux kernel. With CLE, DM multi-pathing is only supported on DDN 9900 arrays. For LSI devices, multi-pathing functionality is provided using the LSI Redundant Disk Array Controller (RDAC). LSI RDAC is a self-contained module that operates as a device driver. This module has no external interfaces; it interacts directly with Linux kernel I/O functionality. For Cray systems, the LSI RDAC driver module must be integrated into the OS boot image so that the RDAC module is loaded before the Fibre Channel Driver is loaded. You must configure system boot scripts to recognize service nodes with LSI connections and load the RDAC and Qlogic driver modules in the correct order. For more information, contact your Cray service representative.

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 that you NEVER mount the same file system in more than one place at the same time. If you do so, the Linux operating system will corrupt the file system.

For more information about configuring RAID, see the *Installing and Configuring Cray Linux Environment (CLE) Software* (S–2444) and *Managing Lustre for the Cray Linux Environment (CLE)* (S–0010).

2.15 Other Administrative Information

This section contains additional information that is helpful for the administrator.

2.15.1 Identifying Components

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.

2.15.1.1 Physical ID

The physical ID identifies the cabinet's location on the floor and the component's location in the cabinet as seen by the HSS.

Table 1 shows the physical ID naming conventions. Descriptions assume that you are standing in front of the system cabinets.

Note: Gemini MMR space **must** use a "g" name, possibly with a NIC identifier; and processor memory **must** use an "n" name.

Table 1. Physical ID Naming Conventions

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	`c`*X*`-`*Y*	Position: row (X) and row (Y) of cabinet; also used as L1 controller host name. For example: `c12-3` is cabinet 12 in row 3.
chassis	`c`*X*`-`*Y*`c#`	Physical unit within cabinet: `c`*X*`-`*Y*; `c#` is chassis 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	`c`*X*`-`*Y*`c#s#`	Physical unit within a slot of a chassis `c`*X*`-`*Y*`c#`; `s#` is the slot of the blade and # is 0-7; also used as L0 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`.

Component	Format	Description
node	c*X*-*Y*c#s#n#	Node on a blade; n# is the location of the node and # is 0-3. For example: c0-0c2s4n0 is node 0 on blade 4 of chassis 2 of cabinet c0-0. For example: c0-0c2s4n* is all nodes on blade 4 of chassis 2 of cabinet c0-0.
GPU	c*X*-*Y*c#s#n#a#	Accelerator module (GPU); a# is the location of the GPU and # is 0-15. For example: c0-0c2s4n0a2 is GPU 2 on node 0 on blade 4 of chassis 2 of cabinet c0-0.
Gemini ASIC	c*X*-*Y*c#s#g#	Gemini ASIC within a module; g# is the location of the Gemini ASIC within a module and # is 0 or 1. For example: c0-1c2s3g0
LCB within a Gemini ASIC	c*X*-*Y*c#s#g#l*RC*	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 (*R*) and 0-7 for column (*C*). For example: c1-0c2s3g0l57 (row 5, column 7) **Note:** The number of LCBs per Gemini ASIC is 48. Of these, LCBs (octal) l23, l24, l33, l34, l43, l44 and l53, l54 are normally used as processor links and not as network links. For this reason a display of the status of LCBs will normally show these LCBs in a different state than the remaining LCBs.

Component	Format	Description
section	t*A*-*B*	Grouping of cabinets; *A* is the start cabinet number and *B* 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 defined when the `xtdiscover` command is executed (see *Installing Cray System Management Workstation (SMW) Software* (S–2480) and the `xtdiscover`(8) man page). 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 60). `p0`, however, is reserved to refer to the entire machine as a partition.
SerDes macro within a Gemini ASIC	c*X*-*Y*c#s#g#m#	SerDes macro within a Gemini ASIC. Each macro implements 4 LCBs. Valid values are 0-9. For example: `c0-1c2s3g0m1`
Node socket	c*X*-*Y*c#s#n#s#	Node socket within a physical node. Valid values are 0-7. For example: `c0-1c2s3n0s1`
Die within a node socket	c*X*-*Y*c#s#n#s#d#	Die within a node physical socket. Valid values are 0-3. For example: `c0-1c2s3n0s1d2`
Core within a die, within a socket, within a node	c*X*-*Y*c#s#n#s#d#c#	Core within a die, within a socket, within a node. Valid values are 0-15. For example: `c0-1c2s3n0s0d1c2`

Component	Format	Description
Memory controller within a die, within a socket, within a node	c*X*-*Y*c#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	c*X*-*Y*c#s#n#d#	DIMM within a node. Valid values are 0-31. For example: `c0-1c2s3n0d3`
Gemini NIC	c*X*-*Y*c#s#g#n#	NIC (Network Interface Controller) within a Gemini ASIC. Valid values are 0 and 1. For example: `c0-1c2s3g0n1`
VERTY	c*X*-*Y*c#s#v#	VERTY (voltage converter/regulator) on a blade. Valid values are 0-15. For example: `c0-1c2s3v0`
FPGA	c*X*-*Y*c#s#f#	FPGA. 0 is the L0E and 1 is the L0G on Gemini systems. For example: `c0-1c2s3f1` is the L0G on a Gemini system.
XDP cabinet	x*X*-*Y*	Power cooling control cabinet. For example: `x0-l`.

2.15.1.2 Node ID (NID)

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; so, cabinet 0 has NIDs 0-95, and cabinet 1 has NIDs 96 - 191, and so on.

A single Gemini ASIC connects to two nodes. 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; as you face 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 3 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.

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 *nid* mappings. For information about using the `nid2nic` command, see the `xtnid2str`(8) man page.

2.15.1.3 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 can be changed if you add or remove nodes (see Changing Nodes and Classes on page 194).

> **Note:** 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 *Installing and Configuring Cray Linux Environment (CLE) Software* (S–2444).

The only restriction about how you name the classes is that the class name must be valid (e.g. the name of a directory.) There is no restriction about how many classes you specify; however, you must use the same class names when you invoke the `xtspec` specialization command (see Specializing Files on page 131).

Change the class of a node (see Changing the Class of a Node on page 136) when you change its function, for example, when you have added an additional login node.

The `/etc/opt/cray/sdb/node_classes` file describes the nodes associated with each class.

Example 1. Sample `/etc/opt/cray/sdb/node_classes` file

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

2.15.2 Topology Class

Each Cray system is given a topology class based in the number of cabinets and their cabling. Some commands, such as `xtbounce`, let you specify topology class as an option.

You can see the class value of your system in a number of places, such as `xtcli status` output, `rca-helper -o` command output (`rca-helper` is run from a Cray node), or by using the `xtclass` command from the SMW:

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

2.15.3 Persistent `/var` Directory

You can set up 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.

Persistent `/var` retains the contents of `/var` directories between system boots. Because the Cray system root file system is read-only, some subdirectories of `/var` are mounted on `tmpfs` (memory) and not on disk. You must take this extra step to keep your files. Configure the values `VAR_SERVER`, `VAR_PATH`, and `VAR_MOUNT_OPTIONS` in the `/etc/sysconfig/xt` file so the service nodes NFS mount that path at boot time. Update `CLEinstall.conf` with these values before running `CLEinstall` to configure persistent `/var`.

Boot scripts and the `xtopview` utility (see Managing System Configuration With the `xtopview` Tool on page 129) respect these configuration values and mount the correct `/var` directory.

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

2.15.4 Default Network IP Addresses

The default IP addresses for network components are described in the *Installing Cray System Management Workstation (SMW) Software* (S–2480).

2.15.5 `/etc/hosts` Files

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 `xtcdr2proc` utility takes information from the Resiliency Communication Agent (RCA) to build the `/etc/hosts` file on the boot node. The `/etc/hosts` file on the boot node maps IP addresses to node IDs on the system interconnection network (see Identifying Components on page 52). The file can also contain aliases for the physical ID location of the system interconnection network components and class names. The following example shows part of the boot node `/etc/hosts` file. The file is updated or created at boot time and contains the default hostname mappings as well as service and HSS 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. NIDs start on 128-count boundaries per cabinet, so cabinet 0 has NIDs 0-95, cabinet 1 starts at 128, and so on. In this example, NID is node ID and component naming information is found in Identifying Components on page 52.

For CNL compute nodes, the `/etc/hosts` file on the boot node is generated at boot time to include CNL compute nodes. Also, the installation and upgrade process modifies the `/etc/hosts` file on the boot root to include CNL compute nodes if they are not included.

The `/etc/hosts` file on the SMW contains `physID`s (physical IDs that map to the physical location of HSS network components), such as the L0 and L1 controllers (see Physical ID on page 53).

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

The `xtdb2etchosts` command converts service information in the SDB to an `/etc/hosts` style file. The resulting `/etc/hosts` file has lines of the following form, where the first column is the IP address, the second column is the NID, and the third column is the service type and class ID of the node:

```
10.131.255.254 nid12345  boot001
10.131.255.253 nid67890  boot002
10.131.255.252 nid55512  login001
```

The service configuration table (`service_config`) in the SDB XTAdmin 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
```

Note: Each time you update or upgrade your CLE software, `CLEinstall` verifies the content of `/etc/opt/cray/sdb/node_classes` and modifies `/etc/hosts` to match the configuration specified in your `CLEinstall.conf` file. For additional detail about how `CLEinstall` modifies the `/etc/hosts` file, see *Installing and Configuring Cray Linux Environment (CLE) Software* (S–2444).

The `xtdb2etchosts` command is documented on the `xtdb2etchosts`(8) man page.

2.15.6 Realm-Specific IP Addressing (RSIP) for CNL Compute Nodes

Realm-Specific IP Addressing (RSIP) allows CNL 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, you may rely on your system's use of private address space and do not need to configure compute nodes with addresses within your site's IP address space. The external hosts see only the external IP addresses of the Cray system.

To configure RSIP for CNL compute nodes, see Configuring Realm-specific IP Addressing (RSIP) on page 206.

2.15.7 Security Auditing

Cray Audit is a set of Cray specific extensions to standard Linux security auditing. When the Cray Audit is configured, separate logs are generated for each audited node on a Cray system. Cray specific utilities simplify administration of auditing options and log files across a large number of nodes. For more information, see Security Auditing and Cray Audit Extensions on page 145.

2.15.8 Logging 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. For more information, see Using the `cray_pam` PAM to Log Failed Login Attempts on page 151.

2.15.9 Logical Machines

You 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 you do not configure more than one logical machine per cabinet. That way, if you power down a cabinet, you do not affect more than one logical machine. A logical machine can include more than one cabinet.
- A job is limited to running within a single logical machine.
- Although the theoretical maximum allowable logical machines per physical Cray system is 31 logical machines (as `p0` is the entire system), you must consider your hardware requirements to determine a practical number of logical machines to configure.
- You can run only a single instance of SMW software.
- Boot and routing commands affect only a single logical machine.

To create logical machines, see Creating Logical Machines on page 182.

Managing the System [3]

Important: 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, you must use persistent device names for file systems on your Cray system. This does **not** apply to SMW disks. For additional information, see Using Persistent SCSI Device Names on page 244.

3.1 Connecting the SMW to the Console of a Service Node

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

Example 2. Establishing a two-way connection between the SMW and console of service node `c0-0c0s0n1`

```
smw:~> xtcon c0-0c0s0n1
--- Console for node c0-0c0s0n1.  Use ^] to quit ---
```

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

3.2 Logging On to the Boot Node

The standard Cray configuration has a single GigE connection between the SMW and boot node. You can access other service nodes from the boot node.

Procedure 1. Logging on to the boot node

- From the SMW, log on to the boot node.

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

  Note: You can open an administrator window on the SMW to access the boot node:

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

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

3.3 Preparing a Service Node and Compute Node Boot Image

This section describes how to prepare a Cray service node and compute node boot image.

A *boot image* is an archive containing all the components necessary to boot Cray service nodes and CNL 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.

The `xtclone`, `xtpackage` and `xtbootimg` utilities run on the SMW. Use these utilities to set up CNL compute nodes or service node images.

Note: You must have root privileges to invoke the `xtclone` and `xtpackage` commands.

You can create a boot node image on the SMW using a four-step process:

1. Run the `xtclone` utility to create your work area, which is copied from the master work area.
2. In your work area, make necessary changes, for example, install RPMs, edit configuration files, or add or remove scripts.
3. Run the `xtpackage` utility to properly package the operating system components and prepare a load file for use by `xtbootimg`.
4. Run the `xtbootimg` utility to create a boot image (an archive or `cpio` file) from your 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`.

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

```
#Kernel source: /opt/xt-images/p1/service/boot/vmlinuz-2.6.27.42-0.1.1_1.0300.4999-cray_ss_s
SNL0/vmlinuz-2.6.27.42-0.1.1_1.0300.4999-cray_ss_s.bin 0x100000
#Parameters source: /opt/xt-images/p1/service/boot/parameters-snl
SNL0/parameters 0x90800
SNL0/initramfs.gz 0xFA00000
SNL0/size-initramfs 0x9021C
```

Cray system compute and service nodes use a RAM disk for booting. Service nodes and CNL compute nodes use the same `initramfs` format and work space environment. This space is created in `/opt/xt-images/`*machine*`-`*xtrelease*`-`*partition*`/`*nodetype*, where *machine* is the Cray hostname, *xtrelease* is the CLE release level, *partition* describes a system partition or is omitted for a full system, and *nodetype* is either `compute` or `service`.

To create load files for supporting, for example, different boot parameters or different RAM disk contents, use the `xtpackage` command with the `-L` option.

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.

Procedure 2. Creating a Cray boot image from existing file system images

1. Make copies of the compute-node-side and service-node-side of the master work area.

   Note: It is recommended that your work area be in a subdirectory of `/opt/xt-images`, as shown in the example.

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

2. Make any changes to your work area that are necessary for your site. For example, you can 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 you are finished making changes, wrap up (package) the compute-node-side and service-node-side of your work area.

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

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

3. Finally, make a boot image (a `cpio` file) from your work area.

```
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
```

Note: 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.

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

3.3.1 Using `shell_bootimage_`*label*`.sh` to Prepare Boot Images

The `CLEinstall` installation program creates a `/var/opt/cray/install/shell_bootimage_`*label*`.sh` script on the SMW. This script is unique to the system set label you installed, based on settings in the `CLEinstall.conf` and `/etc/sysset.conf` installation configuration files. You can re-use this script to automate some of the steps for creating boot images.

Procedure 3. Preparing a boot image for CNL compute nodes and service nodes

Invoke the `shell_bootimage_`*label*`.sh` script to prepare boot images for the system set with the specified *label*. This script uses `xtclone` and `xtpackage` to prepare the work space in `/opt/xt-images`.

`shell_bootimage_`*label*`.sh` accepts the following options:

`-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 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`.

`-h` Display help message.

`-v` Run in verbose mode.

Optionally, this script includes `CNL_*` parameters that you can use to modify the CNL boot image configuration you 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 required to build and set the boot image for the next boot. If you specified the `-c` option, the script invokes these commands automatically and you should skip the remaining steps in this procedure.

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 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.

   Type the following command:

   ```
   smw:~# xtbootimg \
   -L /opt/xt-images/xthostname-CLE_version/compute/CNL0.load \
   -L /opt/xt-images/xthostname-CLE_version/service/SNL0.load \

   -C /opt/hss-coldstart+gemini/default/xt \
   -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, mount the boot node root file system to `/bootroot0`, copy the boot image file from the SMW to the same directory on the boot root, and then unmount the boot node root file system. 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 these commands.

```
smw:~ # mount /dev/bootrootdevice /bootroot0
smw:~ # cp -p /bootimagedir/bootimage.new /bootroot0/bootimagedir/bootimage.new
smw:~ # umount /bootroot0
```

3. Set the boot image for the next system boot 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 that you are using. 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 you 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 this 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 this command to select the boot image to be used for the designated partition.

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

3.3.2 Changing 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. You can either edit the files in the file system image or specify a path to a file containing parameters. If editing the files, the default service and compute node parameters can be found in `boot/parameters-snl` and `boot/parameters-cnl`, respectively.

Example 3. Making 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
```

3.4 Booting Nodes

This section describes how to manually boot your boot node and service nodes and the CNL compute nodes. It also describes how to reboot a single compute node, and reboot login or network nodes.

For information about modifying boot automation files, see Modifying Boot Automation Files on page 187.

3.4.1 Booting the System

Use the `xtbootsys` command to manually boot your boot node, service nodes, and CNL compute nodes.

Note: You can also boot the system using both user-defined and built-in procedures in automation files, for example, `auto.generic.cnl`. Before you modify the `auto.generic.cnl` file, Cray recommends copying it first because it will be replaced by an SMW software upgrade. For related procedures, see *Installing and Configuring Cray Linux Environment (CLE) Software*.

Procedure 4. Manually booting the boot node and service nodes

Note: 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.

Note: If you run more than one boot image, you can check which image you are set up to boot with the `xtcli boot_cfg show` or `xtcli part_cfg show p`*N* commands. To change which image you are booting, see Updating Boot Configuration on page 186.

1. As `crayadm`, use the `xtbootsys` command to boot the boot node.

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

   Note: If you have a partitioned system, invoke `xtbootsys` with the `--partition p`*n* option.

   The `xtbootsys` command prompts you with a series of questions. Cray recommends that you answer yes by typing **Y** to each question.

   The session pauses at:

```
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.
```

   Choose option **10** to boot the boot node and wait.

   You are prompted to confirm your selection. Press the `Enter` key or type **Y** to each question to confirm your selection.

```
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
```

2. After the boot node has finished booting, the process returns to the boot choice menu. Choose option **11** to boot the SDB node and wait.

   You are prompted to confirm your selection. Press the `Enter` key or type **Y** to each question to confirm your selection.

```
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
```

3. Next, select option **13** to boot the service nodes and wait.

   You are prompted to enter a list of the service nodes to be booted. To display service node information, type one of the following commands. Use the `s0` option for the entire system or the `p`*n* option for a partition; for example:

   ```
   smw:~# xtcli status s0 | grep service
   smw:~# xtcli status p2 | grep service
   ```

4. Type a list of service nodes to be booted; for example:

   ```
   c0-0c0s1n0 c0-0c0s1n3 c0-0c0s2n0 c0-0c0s2n3
   ```

   You can also use this format for specifying the same service nodes:

   ```
   c0-0c0s1 c0-0c0s2
   ```

   Alternatively, type **all_serv** to boot all remaining service nodes.

5. You are prompted to confirm your selection. Press the Enter key or type **Y** to each question to confirm your selection.

```
Do you want to boot service c0-0c0s1n0,c0-0c0s1n3,c0-0c0s2n0,c0-0c0s2n3 ? [Yn] Y
Do you want to send the ec_boot event ('no' means to only load memory) ? [Yn] Y
```

   After the specified service nodes are booted, you are prompted to `Enter your boot choice` again. **Do not close** the `xtbootsys` window. You will use this terminal session to boot the compute nodes.

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

Procedure 5. Booting CNL compute nodes

1. After all service and login nodes are booted and Lustre has started (if configured at this time), return to the `xtbootsys` menu.

2. Select **17** from the `xtbootsys` menu. 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 **p***N* (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. After all the compute nodes are booted, return to the `xtbootsys` menu. Type **q** to exit the `xtbootsys` program.

Note: If the system was shut down using `xtshutdown` or `xtbootsys -s last -a auto.xtshutdown`, 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
```

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

To boot a specific image or load file on a given node or set of nodes, you can execute the HSS `xtcli boot` *boot_type* command, as shown in the following examples.

Note: When using a file for the boot image, the same file must be on both the SMW and the `bootroot` at the same path.

Example 4. Booting all service nodes with a specific image

The following example boots all service nodes with the specific image located at `/raw0`:

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

Example 5. Booting all compute nodes with a specific image

The following example boots all compute nodes with the specific image located at */bootimagedir/bootimage*:

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

Example 6. Booting 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
```

3.4.3 Rebooting a Single Compute Node

You can initiate a warm boot with the `xtbootsys` 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, so 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.

Example 7. Rebooting a single compute node

```
crayadm@smw:~> xtbootsys --reboot c1-0c2s1n2
```

3.4.4 Rebooting Login or Network Nodes

Login or network nodes cannot be rebooted via 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.

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

Example 8. Rebooting login or network nodes

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

For additional information, see the `xtbootsys` man page.

3.5 Requesting and Displaying System Routing

Use the HSS `rtr` command to request routing for the system interconnection network, to verify your 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.

Example 9. Displaying 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
NID    NIC-Addr  Node          Gemini         X  Y  Z
----   --------  ------------  ------------   -- -- --
128      256     c1-0c1s0n0    c1-0c1s0g0     0  0  0
129      257     c1-0c1s0n1    c1-0c1s0g0     0  0  0
130      260     c1-0c1s1n0    c1-0c1s1g0     0  0  1
131      261     c1-0c1s1n1    c1-0c1s1g0     0  0  1
...
```

Example 10. Routing the entire system

The `rtr -R|--route-system` command sends a request to the router manager 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 to the router 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
```

For more information about displaying system routing information, see the `rtr`(8) man page.

3.6 Shutting Down the System Using the auto.xtshutdown File

The preferred method to shut down the system is to use the xtbootsys -s last -a auto.xtshutdown command. This method shuts down the compute nodes (which are commonly also Lustre clients), then runs xtshutdown on service nodes, halting the nodes and then stopping processes on the SMW. You 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.

Example 11. Shutting down the system using the auto.xtshutdown file

To shut down the system using the auto.xtshutdown file, execute the following command from the SMW:

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

Or

for a partitioned system with partition p*N*:

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

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

3.7 Shutting Down Service Nodes Using the xtshutdown Command

The xtshutdown command runs from the boot node to shut down the services on service nodes and then shut down the service nodes of the Cray system. It executes a series of commands locally on the boot node and on the service nodes to shut down the system in an orderly fashion.

Procedure 6. Shutting down service nodes

- Modify the /etc/opt/cray/init-service/xtshutdown.conf file or in 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.)

Caution: The xtshutdown command does not shut down compute nodes. To shut down CNL compute nodes and service nodes, see Shutting Down the System or Part of the System Using the xtcli shutdown Command on page 75.

The xtshutdown command uses pdsh to invoke commands on the service nodes you select. You can choose the boot node, SDB node, a class of nodes, or a single host. You 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.

Note: You must be `root` user to use the `xtshutdown` command. Passwordless `ssh` must be enabled for the root user from the boot node to all service nodes.

```
boot:~ # xtshutdown
```

After you have shut down the software on the nodes, you can halt the hardware, reboot, or power down.

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

3.8 Shutting Down the System or Part of the System Using the `xtcli shutdown` Command

The HSS `xtcli shutdown` command allows you to shut 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.

Example 12. Shutting down all compute nodes

To gracefully shut down all compute nodes, execute the following command:

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

Example 13. Shutting down specified compute nodes

To gracefully shut down only compute nodes in cabinet `c13-2`:

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

Example 14. Shutting down all nodes of a system

The `xtcli shutdown` command allows you to shut down the system; to shut down a partition, use the `p`*n* command, where *n* is the partition you want to shut down.

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

Example 15. Forcing nodes to shut down

To force nodes to shut down, for example when all nodes of a system must be halted immediately, use the `-f` argument; you can force a shutdown by using the `-f` argument, even if the nodes have an alert status present. For example:

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

After you have shut down the software on the nodes, you can halt the hardware, reboot, or power down.

For information about shutting down nodes using the `xtcli shutdown` command, see the `xtcli`(8) man page.

3.9 Stopping System Components

When you remove, stop, or power down components, any applications and compute processes that are running on those components are lost.

3.9.1 Reserving a Component

If you want the applications and compute processes to complete before you stop components, use the HSS `xtcli set_reserve` *idlist* command to select the nodes you want to remove. This prevents them from accepting new jobs.

Note: If you are running CNL and using ALPS, after a node is reserved it is considered to be `down` by ALPS. 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.

Procedure 7. Reserving a component

- Type:

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

3.9.2 Powering Down Blades or Cabinets

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

Warning: Ensure the operating system is not running before you power down a blade or a cabinet.

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 (see Appendix B, System States on page 327) to receive power commands.

The `xtcli power force_down` and `xtcli power down_slot` commands are aliases for the `xtcli power down` command.

Procedure 8. Powering down a specified blade

The `xtcli power down` command has the form, where *physIDlist* is a comma-separated list of cabinets or blades present on the system (see Physical ID on page 53).

```
xtcli power down physIDlist
```

- To power down a blade with the ID `c0-0c0s7`, type:

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

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

For information about disabling and enabling components, see Disabling Hardware Components on page 82, and Enabling Hardware Components on page 83, respectively. For information about powering down a component, see the `xtcli_power`(8) man page.

3.9.3 Halting Selected Nodes

You can halt selected nodes with the HSS `xtcli halt` command.

Procedure 9. Halting a node

The command has the form:

```
xtcli halt node
```

- Type:

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

For more information about halting a node, see the `xtcli`(8) man page.

3.10 Restarting a Blade or Cabinet

Note: 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 Appendix B, System States on page 327) to receive power commands.

The `xtcli power up` command does not attempt to power up network mezzanine cards or nodes, which are handled by the `xtbounce` command during system boot. The `xtcli power up_slot` command is an alias for the `xtcli power up` command.

Use the HSS `xtcli power up` command to restart a component.

Procedure 10. Power up blades in a cabinet

The `xtcli power up` command has the form, where *physIDlist* is a comma-separated list of cabinets or blades present on the system (see Physical ID on page 53).

```
xtcli power up physIDlist
```

- Power up the selected blade:

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

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

3.11 Aborting 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 running a specific command such as `xtcli power up` or `xtcli boot`.

Example 16. Aborting a session running on the boot manager

To display all running sessions in the boot manager, execute the following command.

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

Execute the following HSS `xtcli session abort` command to abort session 1 running on the boot manager:

```
crayadm@smw:~> xtcli session abort BM 1
```

Use this command if you have started an `xtcli power` or `xtcli boot` command but want to stop it before the command has completed.

Note: Only the boot manager supports multiple simultaneous sessions.

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

3.12 Displaying and Changing Software System Status

There are a number of tools that enable you to inspect and change the status of compute nodes on a running system.

3.12.1 Displaying the Status of Nodes from the Operating System

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. You must be running ALPS in order for `xtnodestat` to report job information.

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

3.12.2 Viewing and Changing 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 you to retrieve or set the processing mode (`interactive` or `batch`) of specified nodes. You can display the state (`up`, `down`, `admindown`, `route`, or `unavailable`) of the selected components, if needed. You 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`.

Example 17. Looking at node characteristics

```
$ 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       batch
     3      0x3  c0-0c0s1n1  compute        up       batch
     4      0x4  c0-0c0s2n0  compute        up       batch
     5      0x5  c0-0c0s2n1  compute        up       batch
     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
```

Example 18. Viewing all node attributes

Use the `xtprocadmin` command to view current node attributes. The `xtprocadmin -A` option lists all attributes of selected nodes. For example:

```
$ xtprocadmin -A
Connected
    NID   (HEX)    NODENAME     TYPE ARCH       OS  CORES AVAILMEM PAGESZ CLOCKMHZ LABEL0 LABEL1 LABEL2 LABEL3
     0      0x0  c0-0c0s0n0  service   xt (service)     6    16000   4096     2200
     1      0x1  c0-0c0s0n1  service   xt (service)     6    16000   4096     2200
     2      0x2  c0-0c0s1n0  compute   xt       CNL    16    32000   4096     2400
     3      0x3  c0-0c0s1n1  compute   xt       CNL    16    32000   4096     2400
     4      0x4  c0-0c0s2n0  compute   xt       CNL    24    32000   4096     2200
     5      0x5  c0-0c0s2n1  compute   xt       CNL    24    32000   4096     2200
     6      0x6  c0-0c0s3n0  compute   xt       CNL    24    32000   4096     2200
     7      0x7  c0-0c0s3n1  compute   xt       CNL    24    32000   4096     2200
     8      0x8  c0-0c0s4n0  service   xt (service)     6    16000   4096     2200
     9      0x9  c0-0c0s4n1  service   xt (service)     6    16000   4096     2200
```

Example 19. Viewing selected node attributes of selected nodes

The `xtprocadmin -a` *attr1,attr2* option lists selected attributes of selected nodes. For example:

```
$ xtprocadmin -n 7 -a arch,clockmhz,os,cores
Connected
NID (HEX)   NODENAME   TYPE     ARCH   CLOCKMHZ OS  CORES
7   0x7     c0-0c0s1n3 service  xt     2000     CNL 1
```

Example 20. Disabling a node

To mark a node as `admindown` and not allow it to be allocated, type the following command:

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

Example 21. Disabling all processors

To mark all processors as `admindown` and to disable the system's ability to change their state, type the following command:

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

Note: 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.

3.12.3 Marking 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 you 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.

3.12.4 Finding Node Information

3.12.4.1 Translating Between Physical ID Names and Integer NIDs

To translate between physical ID names (cnames) and integer NIDs, generate a system map by using the `rtr` command with the `--system-map` option on the SMW.

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

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

3.12.4.2 Finding 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, Gemini ASIC NIC address values, or Gemini ID values.

Example 22. Finding the physical ID for node 38

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

Example 23. Finding 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'
```

Example 24. Finding 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'
```

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

3.12.4.3 Finding Node Information Using the `nid2nic` Command

Use the `nid2nic` command to print 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.

Example 25. Printing the *nid*-to-*nic* address mappings for the node with NID 31.

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

Example 26. Printing the *nid*-to-*nic* address mappings for the same node as shown in Example 25, but specifying the NIC value in the command line

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

3.13 Displaying and Changing Hardware System Status

You can run commands that look at and change the status of the hardware.

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

3.13.1 Generating HSS Physical IDs

Run the HSS `xtgenid` command to generate HSS physical IDs, for example, to create a list of L0 identifiers for input to the flash manager. You can restrict your selections to components that are of a particular type.

Note: Only user `root` can execute the `xtgenid` command.

Example 27. Creating a list of node identifiers that are not in the `DISABLE`, `EMPTY`, or `OFF` state

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

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

3.13.2 Disabling Hardware Components

If links, nodes, or Cray ASICs have hardware problems, you 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.

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

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

Procedure 11. Disabling a Gemini ASIC

- The `xtcli disable` command has the form:

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

  where *idlist* is a comma-separated list of components (in cname format) that you want the system to ignore. The system disregards these links or nodes.

  Example 28. Disabling 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 `OFF` state, power down the blade containing the ASIC.

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

  3. Disable the ASIC.

     ```
     crayadm@smw:~> xtcli disable c0-0c1s3g
     ```

4. Power up the blade containing the disabled ASIC.

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

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

3.13.3 Enabling Hardware Components

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

By default, when enabling 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 enable command must be used carefully because this may create invalid system state configurations.

Procedure 12. Enabling a Gemini ASIC

- The `xtcli enable` command has the form:

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

  where *idlist* is a comma-separated list of components (in cname format) that you want the system to recognize.

  The state of `off` means that a component is present on the system. If the component is an L0, node, or ASIC, then this will also mean that the component is powered off. If you disable a component, the state shown becomes `disabled`. When you use the `xtcli enable` command 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`.

  Example 29. Enabling Gemini ASIC `c0-0c1s3g0`

  ```
  crayadm@smw:~> xtshow_disabled s0 | grep c0-0c1s3
     c0-0c1s3g0:       -         OP|    disabled      [noflags|]
  c0-0c1s3g0l00:       -         OP|    disabled      [noflags|]
  c0-0c1s3g0l01:       -         OP|    disabled      [noflags|]
  c0-0c1s3g0l02:       -         OP|    disabled      [noflags|]
  c0-0c1s3g0l03:       -         OP|    disabled      [noflags|]
  c0-0c1s3g0l04:       -         OP|    disabled      [noflags|]
  .
  .
  .

  smw:~> xtcli enable c0-0c1s3g0
  Network topology: class 0

  All components returned success.
  ```

For more information about stopping components, see Stopping System Components on page 76 and the `xtcli`(8) man page.

3.13.4 Setting 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 setting a component to an `empty` state, 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.

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

Example 30. Setting a blade to the `empty` state

Set the blade and all its components to `empty`:

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

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

3.13.5 Locking 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 runs. When the manager is finished with the command, it unlocks the components.

Use the HSS `xtcli lock` command to lock components.

Example 31. Locking cabinet `c0-0`

The lock command identifies the session ID. Locking a component prints out the state manager session ID.

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

Example 32. Show all session (lock) data

You can use the `xtcli lock show` command to show session (lock) information.

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

3.13.6 Unlocking Components

Use the HSS `xtcli lock` command to unlock components.

Example 33. Unlocking cabinet `c0-0`

The `xtcli lock` command is useful when a manager fails to unlock some set of components. You can manually check for locks with the `xtcli lock show` command and unlock them. Unlocking a component does not print out the state manager session ID. The `-u` option must be used to unlock a component.

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

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 managers cannot affect components that are locked by a different session.

3.14 Performing Parallel Operations on Nodes

Use the `pdsh` command, which is the CLE parallel remote shell utility, on a service node to issue commands to groups of nodes in parallel. You 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. You must be user `root` to execute the `pdsh` command. The command has the form:

```
pdsh [options] command
```

Example 34. Restarting the NTP service

To restart the network time protocol (NTP) service on the first 9 login nodes, type:

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

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

3.15 xtbounce Error Message Indicating L1 and L0s Not in Sync

During the gather_cab_pwr_states phase of xtbounce, if the HSS software on an L1 and any of its L0s 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
ERROR: rs_phys_node2ascii() failed for node struct 0xb7e70150080700f8
ERROR: rs_phys_node2ascii() failed for node struct 0xb7e70150080700f8
ERROR: rs_phys_node2ascii() failed for node struct 0xb7e70150080700f8
```

If this occurs, it indicates that the L0 software is at a different revision than the L1 software. xtbounce will print a list of cabinets for which this error has occurred. The message will be like:

```
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 L1s and the L0s 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 L0s to finish, then reboot the L1(s) and their associated L0s to get the HSS software synchronized. Following this, the xtbounce may be run once again.

3.16 Handling Bus Errors

Bus errors are caused by machine-check exceptions. If you have received a bus error, try the following procedure:

Procedure 13. Power-cycling a component

Power down then power up components. The *physIDlist* is a comma-separated list of components present on the system (see Physical ID on page 53).

1. Power down the components.

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

2. Power up the components.

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

3.17 Handling Component Failures

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 your Customer Service Representative to schedule a repair.

3.18 Capturing and Analyzing System-level and Node-level Dumps

3.18.1 Dumping Information Using the `xtdumpsys` Command

The `xtdumpsys` 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 `xtdumpsys` command with the `--add` option and name the components from which to collect data. The HSS `xtdumpsys` command saves dump information in `/var/opt/cray/dump/`*timestamp* by default.

Choose the `--snapshot` option to perform a quick analysis of the running system or the `--summary` option to perform an analysis of a previous dump. Neither option creates new files.

Note: Gemini HyperTransport MMRs are read and dumped to a file on the L0 in `/var/log/debug.[0-3]` for the node being dumped.

Example 35. Dumping information about a working component

To dump the entire system and collect detailed information from all L0s in chassis `0` of cabinet `0`, type:

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

Note: An example file, `example.xtdumpsys-plugin`, is included in the `/opt/cray/hss/default/etc` directory and provides techniques to help you customize your own `xtdumpsys` plugin so it can collect additional data during an `xtdumpsys` session.

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

3.18.2 ldump and lcrash Utilities for Node Memory Dump and Analysis

The ldump and lcrash utilities may be used to analyze the memory on any Cray service node or CNL compute node. The ldump command is used to dump node memory to a file. After ldump completes, you may then use the lcrash utility on the dump file generated by ldump.

Cray recommends running the ldump 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 ldump -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 via 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 Gemini 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.

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

method[*@host*]

For additional information, see the ldump(8) and lcrash(8) man pages.

3.18.3 Using dumpd to Automatically Dump and Reboot Nodes

The SMW daemon dumpd initiates automatic dump and reboot of nodes when requested by 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.

You 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 Configuring Node Health Checker (NHC) on page 160.

You can also set variables in the /etc/opt/cray-xt-dumpd/dumpd.conf configuration file on the SMW to control how dumpd behaves on your 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 your site-specific modifications previously made to the file. However, you should compare your /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 your /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 you must explicitly enable it.

3.18.3.1 Enabling dumpd

Procedure 14. Enabling 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 if you want dumps to be taken.

3. Specify an action of dump, reboot, or dumpreboot for any tests for which you want NHC to make a request of dumpd when that test fails.

4. In 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.

3.18.3.2 `/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 for you to change any installation configuration parameters; however, you may edit the `/etc/opt/cray-xt-dumpd/dumpd.conf` file to customize how `dumpd` behaves on your system using the following configuration variables.

`enable: yes|no`

Provides a quick on/off switch for all `dumpd` functionality (the default value in the file provided from Cray Inc. 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 you specify `ignore`, requests are removed from the database and not acted upon. If you specify `queue`, 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 run. 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 run if the directory specified by the variable `dump_dir` does not have enough space free, as specified by `max_disk`.

Examples of possible actions:

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 email to an administrator at admin@fictionalcraysite.com:

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

3.18.3.3 Using 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.

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

3.18.3.4 Using the dumpd-request Tool

You can 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
```

You can define additional actions in the dumpd.conf configuration file; to use, you must 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, you must define a myaction1 and myaction2 in the dumpd.conf file. See the examples in the configuration file for more detail.

3.19 Using xtnmi Command to Collect Debug Information from Hung Nodes

Caution: This is not a harmless tool to use to repeatedly get information from a node at various times; only use this command when you need debugging data from nodes that are in trouble. 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.

Monitoring System Activity [4]

4.1 Monitoring the System with the System Environmental Data Collector (SEDC)

To use the System Environmental Data Collector (SEDC) to collect data about internal cabinet temperatures, cooling system air pressures, critical voltages, etc., see *Using and Configuring System Environment Data Collections (SEDC)*.

4.2 Displaying 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.

Example 36. Displaying installed SMW release level

```
% cat /opt/cray/hss/default/etc/smw-release
6.0.UP01
```

4.3 Displaying Installed CLE Release Level

Following a successful installation, the file `/etc/opt/cray/release/clerelease` is populated with the installed CLE release level.

Example 37. Displaying installed CLE release level

```
% cat /etc/opt/cray/release/clerelease
4.0.UP01
```

Note: This file only indicates the CLE version that was last installed, but it may not be the CLE version that is currently running on your system.

4.4 Displaying 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.

Procedure 15. Showing boot configuration information for the entire system

- To display boot configuration information for the entire system, execute the `xtcli boot_cfg show` command:

```
crayadm@smw:~> xtcli boot_cfg show
Network topology: class 0
[boot]: c0-0c0s0n1:halt
[sdb]: c0-0c0s2n1:halt
[cpio_path]: /tmp/boot/cray_system-p1-CLE40-4.0.18c02.cpio_0705hss
```

Procedure 16. Showing boot configuration information for a partition of a system

- To display boot configuration information for one partition in a system, specify the partition number, p*N*. `p0` is always the whole system:

```
crayadm@smw:~> xtcli part_cfg show p1
Network topology: class 0
=== part_cfg ===
------------------
[partition]: p1: enable (noflags|)
[members]: c0-0c0
[boot]: c0-0c0s0n1:halt
[sdb]: c0-0c0s2n1:halt
[cpio_path]: /tmp/boot/cray_system-p1-CLE40-4.0.18c02.cpio_0705hss
================
```

4.5 Managing 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.

CLE log files are saved on the `syslog` node in the `message` file located by default in the `/syslog/var/log` directory. (The directory path is set in the `sysset.conf` and `CLEinstall.conf` files.)

Linux system event log files are stored on the service partition.

4.5.1 Filtering the Event Log

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

Example 38. Finding information in the event log

To search for all console messages from node `c9-2c0s3n2`, type:

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

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

4.5.2 Adding Entries to Log Files

You can add entries to the `syslog` with the `logger` command. For example, to identify the start or finish of system activities, use the `/bin/logger` command to log events into the system log, `/syslog/var/log/messages`. The message is then available to anyone who reads the log.

Example 39. Adding entries to `syslog` file

To mark the start of a new system test, type:

```
node/3/:/ # 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 2011
```

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

4.5.3 Examining 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/bootlogs` 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/bootlogs/console.1107132305`.

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.

4.5.4 Removing Old Log Files

The HSS command `/opt/cray/hss/default/bin/xtclean_logs` removes outdated HSS log files based on a user-specified cutoff date. This command is a bash script that checks the SMW directories `/var/opt/cray/log`, `/var/opt/cray/log/bootlogs`, `/var/opt/cray/log/diaglogs`, and `/var/opt/cray/dump`.

Attach the `xtclean_logs` script to the desired `cron` job by copying it into the appropriate `cron` directory. The owner and group must be `root`, and permissions must be `755`. The script is triggered by `/usr/lib/cron/run-crons`.

To choose the log files to delete, modify the parameters to specify the number of days (age) that files are to be kept in the log directories. To turn off file deletion, set the parameter value(s) to `0`. Parameters include:

`BOOT_DAYS_TO_KEEP`

Number of days after which files in `/var/opt/cray/log/bootlogs` are deleted, which is set by the `SMWinstall.conf` file variable `BootDtK` when the `SMWinstall` program is executed. Boot log file names indicate the start of the boot session, and this value is used to determine age. The default parameter setting provided by Cray is `30` days.

Note: Boot session files for the current boot session are not deleted, regardless of age.

`EVENT_DAYS_TO_KEEP`

Number of days after which files in `/var/opt/cray/log/eventlog` are deleted, which is set by the `SMWinstall.conf` file variable `EventDtK` when the `SMWinstall` program is executed. The age of each `eventlog` file is determined by its last modification time. The default parameter setting provided by Cray is `30` days.

`DIAG_DAYS_TO_KEEP`

Number of days after which diagnostic directories in `/var/opt/cray/log/diaglogs` are deleted, which is set by the `SMWinstall.conf` file variable `DiagDtK` when the `SMWinstall` program is executed. Diagnostic subdirectory names include last modification time, and this value is used to determine age. The default parameter setting provided by Cray is `60` days.

`DUMP_DAYS_TO_KEEP`

Number of days after which dump directories in `/var/opt/cray/dump` are deleted, which is set by the `SMWinstall.conf` file variable `DumpDtK` when the `SMWinstall` program is executed. Dump subdirectory names include the last modification time, and this value is used to determine age. The default parameter setting provided by Cray is `30` days.

The `SMWinstall` program automatically creates `/etc/cron.daily/xtclean_logs` and sets it to execute. If your log rotation policy requires that you clean the HSS log files on a schedule that differs from the default, set the `SMWinstall.conf` file variable `ENABLE_XTCLEAN_LOGS` to `no`; this creates `/etc/cron.daily/xtclean_logs` but does not set it to execute.

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

4.6 Managing Log Files Using the Cray Management Services (CMS) Log Manager

The CMS log manager provides the capability to collect, analyze, and display messages from the system. The data is collected from a variety of sources and loaded into the CMS database on the SMW. You can view and search the database for events of interest. Notification of events on the event router uses the `mzlogmanagerd` daemon; notification of events in the `syslog` uses the `mzsyslogd`.

For additional information, see *Using Cray Management Services (CMS)*.

4.7 Checking the Status of System Components

To check the status of the system or a component, use the `xtcli status` command on the SMW. By default, the `xtcli status` command returns the status of nodes.

Procedure 17. Showing the status of a component

- The `xtcli status` command has the form:

  `xtcli status [-n] [{-t` *type*`] [-a]}` *node_list*

  Note: The list should have component IDs only (no wild cards).

type may be: `node`, `l0`, `cage`, `l1`, `xdp`, `verty`, `dimm`, `socket`, `die`, `core`, `memctrl`, `gemini`, `nic`, `lcb`, `serdes_macro`, `fpga`, or `accel`.

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

4.8 Checking the Status of Compute Processors

To check that compute nodes are available after the system is booted, use the `xtprocadmin` command on a service node.

Example 40. Identifying nodes in down or admindown state

To identify if there are any nodes that are in a `down` or `admindown` state, execute the following command from a node:

```
nid00007:~> xtprocadmin | grep down
```

Example 41. Display current allocation and status of each compute processing element and the application that it is running

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. For example:

```
nid00007:~> xtnodestat
Current Allocation Status at Wed Jul 06 13:53:26 2011

     C0-0
  n3 AAaaaaaa
  n2 AAaaaaaa
  n1 Aeeaaaa-
c2n0 Aeeaaaaa
  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.

4.9 Checking CNL Compute Node Connection

Use the Linux `ping` command to verify that a compute node is connected to the network. The Linux `ping` command must be run from a node, not run on the SMW.

Example 42. Verifying 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
```

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

4.10 Checking 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 L0-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.

You 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 that you run `xtnetwatch` each time you boot the system with the `xtbootsys` command (the default). The output is a time-stamped log file such as:

`/var/opt/cray/log/bootlogs/netwatch.1107131250`

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.

Note: To turn off L0 high-speed interconnect link monitoring, use the `xtnetwatch -d` option.

Example 43. Running `xtnetwatch` to monitor the system interconnection network

Sample the network once every 10 seconds using the least verbose display format:

```
crayadm@smw:~> xtnetwatch -i 10
100407 23:01:58  ################  ################  ##################    ##################
100407 23:01:58  LCB ID            Peer LCB          Soft Errors           Fatal Errors
100407 23:01:58  ################  ################  ##################    ##################
100407 23:02:27  c1-0c0s3g0l30     c0-0c0s3g0l30       1 TX lanemask=5
100407 23:02:27  c1-0c0s4g1l03     c1-0c0s3g1l03       1 Mode Exchanges
100407 23:02:27  c1-3c1s2g1l03     c2-2c0s7g0l53                             1 Link Inactive
```

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

4.11 Monitoring the Status of Jobs Started Under a Third-party Batch System

To monitor the status of jobs that were started under a third-party batch system, use the command appropriate to your batch system. For more information, see the documentation provided by your batch system vendor.

4.12 Using the `cray_pam` Module to Monitor Failed Login Attempts

The `cray_pam` module is a Pluggable Authentication Module (PAM). When configured, the `cray_pam` module provides information to the user at login time about any failed login attempts since their last successful login. See Using the `cray_pam` PAM to Log Failed Login Attempts on page 151 and the procedure to configure the `cray_pam` module, Procedure 32 on page 152.

4.13 Monitoring DDN RAID

Use Data Direct Networks tools to monitor DDN RAID. These can be accessed by telnetting to the RAID device from the SMW. To configure remote logging of DDN messages, see the *Cray System Management Workstation (SMW) Software Installation Guide*. For additional information, see your DDN documentation.

4.14 Monitoring LSI Engenio RAID

Use Engenio tools to monitor Engenio RAID. The LSI Engenio storage system uses SNMP to provide boot RAID messages. For additional information, see your LSI Engenio Storage System documentation.

4.15 Monitoring HSS Managers

This section provides procedures to view active sessions and to check whether the boot manager or the L0 or L1 controller daemons are running.

4.15.1 Examining 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.

Example 44. Looking at a session running on the boot manager

Execute the HSS `xtcli session show` command to view the session running on the boot manager:

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

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

4.15.2 Polling 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.

Example 45. Checking the boot manager

```
crayadm@smw:~> xtalive -l smw -a bm s0
```

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

4.16 Monitoring Events

The HSS `xtconsumer` command enables you to monitor events mediated by the event router daemon `erd`, which runs passively.

Example 46. Monitoring for specific events

This command shows watching 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 L0 report a bad health indication.

```
crayadm@smw:~> xtconsumer -b ec_heartbeat_stop ec_l0_health
```

Example 47. Checking events except heartbeat:

To display all events except heartbeats:

```
crayadm@smw:~> xtconsumer -x ec_l1_heartbeat
```

Use the `xthb` command to confirm the stopped heartbeat. Use the `xthb` command only when you are actively looking into a known problem because it is intrusive and degrades system performance.

For more information, see the `xtconsumer`(8) and `xthb`(8) man pages.

4.17 Monitoring Node Console Messages

Use the HSS `xtconsole` command to monitor and display console messages of a specific node. The command can monitor a single node or multiple nodes.

Example 48. Obtaining node console messages

To view the console output of node `c30-0c0s0n0`, type:

```
crayadm@smw:~> xtconsole c30-0c0s0n0
```

To view the console output from all nodes, use the `xtconsole -a` command. Using the `xtconsole -at` command adds a timestamp as a prefix to the line (use `-t` to show the current time as the prefix. Use `-t -t` to show the current date and current time as the prefix):

```
crayadm@smw:~> xtconsole -at
[11:25:47][c0-0c0s2n0]LDISKFS-fs warning: maximal mount count reached, running e2fsck is recommended
[11:25:47][c0-0c0s2n3]LDISKFS-fs warning: maximal mount count reached, running e2fsck is recommended
[11:25:47][c0-0c0s2n3]LDISKFS-fs warning: maximal mount count reached, running e2fsck is recommended
[11:25:47][c0-0c0s2n0]LDISKFS-fs warning: maximal mount count reached, running e2fsck is recommended
[11:25:47][c0-0c0s2n0]LDISKFS-fs: file extents enabled
[11:25:47][c0-0c0s2n0]LDISKFS-fs: mballoc enabled
[11:25:47][c0-0c0s2n3]LDISKFS-fs warning: maximal mount count reached, running e2fsck is recommended
[11:25:47][c0-0c0s2n3]LDISKFS-fs: file extents enabled
[11:25:47][c0-0c0s2n3]LDISKFS-fs: mballoc enabled
[11:25:47][c0-0c0s2n3]LDISKFS-fs warning: maximal mount count reached, running e2fsck is recommended
[11:25:47][c0-0c0s2n3]LDISKFS-fs: file extents enabled
[11:25:47][c0-0c0s2n3]LDISKFS-fs: mballoc enabled
[11:25:47][c0-0c0s2n0]LDISKFS-fs: file extents enabled
[11:25:48][c0-0c0s2n0]LDISKFS-fs: mballoc enabled
[11:25:48][c0-0c0s2n0]LDISKFS-fs warning: maximal mount count reached, running e2fsck is recommended
[11:25:48][c0-0c0s2n0]LDISKFS-fs warning: maximal mount count reached, running e2fsck is recommended
[11:25:48][c0-0c0s2n0]LDISKFS-fs: file extents enabled
...
```

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

4.18 Showing the Component Alert, Warning, and Location History

Use the `xtcli comp_hist` command to display the 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.

Procedure 18. Displaying the location history for component `c0-0c0s0n1`

- Type:

  ```
  crayadm@smw:~> xtcli comp_hist -o loc c0-0c0s0n1
  ```

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

4.19 Displaying Component Information

Use the HSS `xtshow` command to identify compute and service components. Commands are typed as `xtshow --`*option_name*. You can also combine the `--service` or `--compute` option with other `xtshow` options to limit your selection to the specified type of node.

Example 49. Identifying all service nodes

```
crayadm@smw:~> xtshow --service
XDPs ...
L1s ...
Cages ...
L0s ...
         c0-0c0s0: service         OP|       ready [noflags|]
         c0-0c0s1: service         OP|       ready [noflags|]
         c0-0c0s2: service         OP|       ready [noflags|]
         c0-0c1s0: service         OP|       ready [noflags|]
         c0-0c1s1: service         OP|       ready [noflags|]
         c0-0c1s2: service         OP|       ready [noflags|]
         c0-0c2s0: service         OP|       ready [noflags|]
         c0-0c2s1: service         OP|       ready [noflags|]
         c0-0c2s6: service         OP|       ready [noflags|]
Nodes ...
       c0-0c0s0n0: service  IB06   OP|        halt [noflags|]
       c0-0c0s0n1: service  IB06   OP|        halt [noflags|]
       c0-0c0s0n2: service  IB06   OP|        halt [noflags|]
       c0-0c0s0n3: service  IB06   OP|        halt [noflags|]
       c0-0c0s1n0: service  IB06   OP|        halt [noflags|]
       c0-0c0s1n1: service  IB06   OP|        halt [noflags|]
       c0-0c0s1n2: service  IB06   OP|        halt [noflags|]
       c0-0c0s1n3: service  IB06   OP|        halt [noflags|]
       c0-0c0s2n0: service  OPTD   OP|        halt [noflags|]
       c0-0c0s2n1: service         OP|       empty [noflags|]
       c0-0c0s2n2: service         OP|       empty [noflags|]
       c0-0c0s2n3: service  OPTD   OP|        halt [noflags|]
       c0-0c1s0n0: service         OP|       ready [noflags|]
       c0-0c1s0n1: service         OP|       empty [noflags|]
       c0-0c1s0n2: service         OP|       empty [noflags|]
       c0-0c1s0n3: service         OP|       ready [noflags|]
       c0-0c1s1n0: service  OPTD   OP|       ready [noflags|]
       c0-0c1s1n1: service         OP|       empty [noflags|]
       c0-0c1s1n2: service         OP|       empty [noflags|]
       c0-0c1s1n3: service  OPTD   OP|       ready [noflags|]
       c0-0c1s2n0: service  OPTD   OP|       ready [noflags|]
       c0-0c1s2n1: service         OP|       empty [noflags|]
       c0-0c1s2n2: service         OP|       empty [noflags|]
       c0-0c1s2n3: service  OPTD   OP|       ready [noflags|]
       c0-0c2s0n0: service  OPTD   OP|       ready [noflags|]
       c0-0c2s0n1: service         OP|       empty [noflags|]
       c0-0c2s0n2: service         OP|       empty [noflags|]
       c0-0c2s0n3: service  OPTD   OP|       ready [noflags|]
       c0-0c2s1n0: service  OPTD   OP|       ready [noflags|]
       c0-0c2s1n1: service         OP|       empty [noflags|]
       c0-0c2s1n2: service         OP|       empty [noflags|]
       c0-0c2s1n3: service  OPTD   OP|       ready [noflags|]
       c0-0c2s6n0: service  IB06   OP|       ready [noflags|]
       c0-0c2s6n1: service  IB06   OP|       ready [noflags|]
```

```
       c0-0c2s6n2: service  IB06   OP|       ready [noflags|]
       c0-0c2s6n3: service  IB06   OP|       ready [noflags|]
GPUs ...
Geminis ...
       c0-0c0s0g0: service         OP|          on [noflags|]
       c0-0c0s0g1: service         OP|          on [noflags|]
       c0-0c0s1g0: service         OP|          on [noflags|]
       c0-0c0s1g1: service         OP|          on [noflags|]
       c0-0c0s2g0: service         OP|          on [noflags|]
       c0-0c0s2g1: service         OP|          on [noflags|]
       c0-0c1s0g0: service         OP|          on [noflags|]
       c0-0c1s0g1: service         OP|          on [noflags|]
       c0-0c1s1g0: service         OP|          on [noflags|]
       c0-0c1s1g1: service         OP|          on [noflags|]
       c0-0c1s2g0: service         OP|          on [noflags|]
       c0-0c1s2g1: service         OP|          on [noflags|]
       c0-0c2s0g0: service         OP|          on [noflags|]
       c0-0c2s0g1: service         OP|          on [noflags|]
       c0-0c2s1g0: service         OP|          on [noflags|]
       c0-0c2s1g1: service         OP|          on [noflags|]
       c0-0c2s6g0: service         OP|          on [noflags|]
       c0-0c2s6g1: service         OP|          on [noflags|]
Nics ...
(snip)
crayadm@smw:~>
```

Example 50. Showing compute nodes in the `disabled` state

```
crayadm@smw:~> xtshow --compute --disabled
XDPs ...
L1s ...
Cages ...
L0s ...
Nodes ...
       c2-0c1s2n1:       -         OP|    disabled      [noflags|]
       c3-0c0s5n2:       -         OP|    disabled      [noflags|]
       c3-0c1s5n0:       -         OP|    disabled      [noflags|]
       c3-0c2s2n0:       -         OP|    disabled      [noflags|]
GPUs ...
Geminis ...
Nics ...
Serdes_macros ...
Sockets ...
     c2-0c1s2n1s0:       -         OP|    disabled      [noflags|]
     c3-0c0s5n2s0:       -         OP|    disabled      [noflags|]
     c3-0c1s5n0s0:       -         OP|    disabled      [noflags|]
     c3-0c2s2n0s0:       -         OP|    disabled      [noflags|]
Dies ...
   c2-0c1s2n1s0d0:       -         OP|    disabled      [noflags|]
   c3-0c0s5n2s0d0:       -         OP|    disabled      [noflags|]
   c3-0c1s5n0s0d0:       -         OP|    disabled      [noflags|]
   c3-0c2s2n0s0d0:       -         OP|    disabled      [noflags|]
Cores ...
 c2-0c1s2n1s0d0c0:       -         OP|    disabled      [noflags|]
 c3-0c0s5n2s0d0c0:       -         OP|    disabled      [noflags|]
 c3-0c1s5n0s0d0c0:       -         OP|    disabled      [noflags|]
 c3-0c2s2n0s0d0c0:       -         OP|    disabled      [noflags|]
Memctrls ...
 c2-0c1s2n1s0d0m0:       -         OP|    disabled      [noflags|]
 c3-0c0s5n2s0d0m0:       -         OP|    disabled      [noflags|]
 c3-0c1s5n0s0d0m0:       -         OP|    disabled      [noflags|]
 c3-0c2s2n0s0d0m0:       -         OP|    disabled      [noflags|]
Dimms ...
     c2-0c1s2n1d0:       -         OP|    disabled      [noflags|]
     c2-0c1s2n1d1:       -         OP|    disabled      [noflags|]
     c2-0c1s2n1d2:       -         OP|    disabled      [noflags|]
.
.
.
Vertys ...
FPGAs ...
Lcbs ...
```

4.20 Displaying 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`.

Note: Alerts are not propagated through the system hierarchy, so you only receive information for the component you are examining. A node can have an alert, but you would not see it if you ran the `xtshow --alert` command for a cabinet. In a similar fashion, you would not detect an alert on a cabinet if you checked the status of a node.

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 you manually clear them. For example, the temporary loss of a heartbeat by the L0 controller may set a warning state on a chip.

4.21 Clearing Flags

Use the `xtclear` command to clear system information for components you select. Type commands as `xtclear --`*option_name*, where *option_name* is `alert`, `reserve`, or `warn`.

You must clear alerts, reserves, and warnings before a component can operate. Clearing an alert on a component frees its state so that subsequent commands can execute (see Appendix B, System States on page 327).

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

Managing User Access [5]

For a description of administrator accounts that enable you to access the functions described in this chapter, see Administering Accounts on page 113.

5.1 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 on page 38.) 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. You can use the `lbnamed` load-balancing software to distribute logins to different login nodes. The `lbnamed` 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 `lbnamed` runs on the SMW, `eth0` on the SMW must be connected to the same network from which users log on the login nodes.

> **Note:** If security considerations do not allow you to put the SMW on the public network, `lbnamed` 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.

The behavior of the `lbnamed` daemon is site-configurable and determined by the contents of the `/etc/opt/cray-xt-lbnamed/lbnamed.conf` and `/etc/opt/cray-xt-lbnamed/poller.conf` configuration files. For details about configuring the load balancer, see Configuring the Load Balancer on page 158, and the `lbcd`(8), `lbnamed`(8), and `lbnamed.conf`(5) man pages.

5.2 Passwords

The default passwords for the `root` and `crayadm` accounts are the same for the System Management Workstation (SMW), the boot node, and the shared root.

Default passwords for the root and crayadm accounts are provided in the *Installing and Configuring Cray Linux Environment (CLE) Software*. Also, default MySQL passwords and an example of how to change them are provided in the *Installing and Configuring Cray Linux Environment (CLE) Software*. Cray recommends changing these default passwords as part of the software installation process.

5.2.1 Changing Default SMW Passwords After Completing Installation

After completing the installation, change the default SMW passwords. 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
```

5.2.2 Changing 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.

Procedure 19. Changing the root and crayadm passwords on boot and service nodes

1. The boot node 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 boot node, use these commands. You will be prompted to type and confirm new root and administrative passwords.

   ```
   boot:~ # passwd root
   boot:~ # passwd crayadm
   ```

2. To change the passwords on the other service nodes, you must run these commands on the shared root. Again, you will be prompted to type and confirm new passwords for the `root` and `crayadm` accounts.

   Note: If the SDB node is not started, you must add the `` `-x /etc/opt/cray/sdb/node_classes` `` option to the `xtopview` command in this procedure.

```
boot:~ # xtopview
default/:/ # passwd root
default/:/ # passwd crayadm
default/:/ # exit
```

For more information about using the `xtopview` command, see Managing System Configuration With the `xtopview` Tool on page 129, and the `xtopview`(8) man page.

5.2.3 Changing the `root` Password on CNL Compute Nodes

Procedure 20. Changing the `root` password on CNL compute nodes

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

Note: To make these changes for a system partition, rather than for the entire system, replace `/opt/xt-images/templates` with `/opt/xt-images/templates-p`*N*, 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. After making these changes, update the boot image by following the steps in Procedure 3 on page 66.

5.2.4 Changing 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 you for the current HSS data store (MySQL) `root` password. When you type the current and new passwords, they are not echoed.

Note: The `hssds_init` utility is run by the `SMWinstall` command, so you do not have to run it during installation of an SMW release package.

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

5.2.5 Changing Default MySQL Passwords on the SDB

Procedure 21. Changing default MySQL passwords on the SDB

For security, you should change the default passwords for MySQL database accounts.

1. If you have not set a site-specific MySQL password for root, type the following commands. Press the `Enter` key when prompted for a password.

```
boot:~ # ssh root@sdb
sdb:~ # mysql -h sdb -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.0.64-enterprise MySQL Enterprise Server (Commercial)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
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. To change the password for the `sys_mgmt` account, type the following MySQL command. You must also update `.my.cnf` in step 4.

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

   b. To change the password for the `basic` account, type the following MySQL command. You must also update `/etc/opt/cray/MySQL/my.cnf` in step 5.

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

   c. To change the password for the `mazama` account, type the following MySQL commands. You must also update `/etc/sysconfig/mazama` in step 6.

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

Note: When making changes to the MySQL database, your connection may time out; however, it is automatically reconnected. If this happens, you will see messages similar to the following. 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
boot:~ #
```

4. (Optional) If you set a site-specific password for `sys_mgmt` in step 2, update the `.my.cnf` file for root with the new password.

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

```
boot:~ # cd ~root
boot:~ # vi .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
default/:/ # exit
boot:~ #
```

5. (Optional) If you set a site-specific password for `basic` in step 2, update the `/etc/opt/cray/MySQL/my.cnf` file with the new password.

   a. Edit `/etc/opt/cray/MySQL/my.cnf` on the boot node.

```
boot:~ # vi /etc/opt/cray/MySQL/my.cnf
# The following options will be passed to all MySQL clients
[client]
user=basic
password=newpassword
```

   b. Edit `/etc/opt/cray/MySQL/my.cnf` in the shared root.

```
boot:~ # xtopview
default/:/ # vi /etc/opt/cray/MySQL/my.cnf
# The following options will be passed to all MySQL clients
[client]
user=basic
password=newpassword
default/:/ # exit
boot:~ #
```

6. (Optional) If you set a site-specific password for `mazama` in step 2, update the `/etc/sysconfig/mazama` file with the new password. In addition, update the `mazama` MySQL account on the SMW to match.

a. Edit `/etc/sysconfig/mazama` on the boot node.

```
boot:~ # vi /etc/sysconfig/mazama
## Type:                string
## Default:             mazama
## Config:              ""
#
# Default password for mazama user in the mazama database
#
passwd=newpassword
```

b. Edit `/etc/sysconfig/mazama` in the shared root.

```
boot:~ # xtopview
default/:/ # vi /etc/sysconfig/mazama
## Type:                string
## Default:             mazama
## Config:              ""
#
# Default password for mazama user in the mazama database
#
passwd=newpassword
default/:/ # exit
boot:~ #
```

c. To change the password for the MySQL accounts on the SMW, type the following MySQL commands.

```
boot:~ # exit
smw:~# mysql -u root -p
mysql> set password for 'mazama'@'%' = password('newpassword');
mysql> set password for 'mazama'@'localhost' = password('newpassword');
mysql> set password for 'mazama'@'smw' = password('newpassword');
mysql> exit
```

d. Update `/etc/sysconfig/mazama` on the SMW.

```
smw:~# vi /etc/sysconfig/mazama
## Type:                string
## Default:             mazama
## Config:              ""
#
# Default password for mazama user in the mazama database
#
passwd=newpassword
```

Make the following additional change, unless you are using a remote MySQL server for CMS logs.

```
## Type:                string
## Default:             mazama
# Default password for mazama user in the mazama Log database
#
log_passwd=newpassword
```

5.2.6 Assigning and Changing User Passwords

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

5.2.7 Logins That Do Not Require Passwords

All logins must have passwords; however, you can set up 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 you wish to use passphrase-less authentication, Cray encourages you to consider using `ssh-agent` if available, or take other steps to mitigate risk.

5.3 Administering Accounts

Your Cray system supports several types of accounts:

- Boot node accounts allow only system administrator (`crayadm`) and superuser (`root`) access. To modify configuration files, the administrator must become superuser by supplying the `root` account password.
- SMW user account access is managed using local files (that is, `/etc/passwd`, `/etc/shadow`, etc.). In addition to the standard Linux system accounts, Cray includes an account named `crayadm`; `crayadm` is used for many of the Cray system management functions, such as booting the system.
- Cray provides a Virtual Network Computing (VNC) account on the SMW (for details, see Appendix D, Remote Access to the SMW on page 341).

5.3.1 Managing Boot Node Accounts

The only accounts that are supported on the boot node are `root` (superuser), `crayadm` (administrator), and those accounts for various services such as network time protocol (NTP). The boot node does not support user accounts.

The boot node has an `/etc/passwd` file that is separate from the password file for the rest of the system. For a list of default passwords, see *Installing and Configuring Cray Linux Environment (CLE) Software*.

5.3.2 Managing User Accounts on Service Nodes

User accounts are set up on the shared-root file system by using the `xtopview` command. Your Cray system supports 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 `root`, `crayadm`, and `mysql`.

For more information about using the `xtopview` command in the default view, see Managing System Configuration With the `xtopview` Tool on page 129, and the `xtopview`(8) man page. For more information about `mysql` accounts, see Database Security on page 192.

5.3.2.1 Adding a User or Group

To add additional accounts to the shared root for login nodes, use the `groupadd` and `useradd` commands using the `xtopview` command in the default view.

Example 51. Adding a group

To add the group `xtusers` with a gid of `5605`, type:

```
boot:~ # xtopview
default/:/ # groupadd -g 5605 xtusers
default/:/ # exit
```

The above `groupadd` command adds group `xtusers` to `/etc/group`.

Example 52. Adding a user account

This example creates a new user `bobp` from `xtopview` in the default view. The new user account, `bobp`, has a user ID of 12645, a home directory `bobp`, and runs a `/bin/bash` login shell. Then, as `root`, create the user's home directory and `chown` the directory to the new user.

```
boot:~ # xtopview
default/:/ # useradd -d /home/users/bobp -g 5605 -s /bin/bash -u 12645 bobp
default/:/ # exit
boot:~ # ssh root@login
login:~ # mkdir -p /home/users/bobp
login:~ # chown -R bobp:xtusers /home/users/bobp
```

After the account is created, use the `passwd` command to set a password in either `/etc/passwd` or `/etc/shadow`.

For more information, see the `useradd`(8), `passwd`(1), and `groupadd`(8) man pages.

5.3.2.2 Removing a User or Group

To remove a user account, first remove all files, jobs, and other references to the user. Then using the `xtopview` command in the default view, remove users or groups by using Linux commands `/usr/sbin/userdel` and `/usr/sbin/groupdel`, respectively; and, as `root`, remove the user's home directory.

Example 53. Removing a user account

To remove the user `bobp` and the user's home directory, type:

```
boot:~ # xtopview
default/:/ # userdel -r bobp
default/:/ # exit
boot:~ # ssh root@login
login:~ # rm -rf /home/users/bobp
login:~ # exit
```

For more information, see the `userdel`(8) and `groupdel`(8) man pages.

5.3.2.3 Changing User or Group Information

To change user and group information, use Linux commands. For more information, see the `usermod`(8) and `groupmod`(8) man pages.

5.3.2.4 Assigning Groups of CNL Compute Nodes to a User Group

Use the `/etc/opt/cray/sdb/attr.defaults` file `label` attribute to assign groups of CNL compute nodes to specific user groups without the need to partition the system. For more information, see Setting Node Attributes Using the `/etc/opt/cray/sdb/attr.xthwinv.xml` and `/etc/opt/cray/sdb/attr.defaults` Files on page 198.

5.3.2.5 Associating Users with Projects

You 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.

If your users run batch jobs, they can set a project code; 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 your batch system vendor.

5.3.2.6 Enabling LDAP Support for User Authentication

To enable LDAP support for user authentication, you must edit files as shown, in addition to making any other standard LDAP configuration setting changes necessary for your site.

Note: The following changes should be made using `xtopview` in the default view.

```
boot:~ # xtopview
default/:/ # vi /etc/pam.d/common-account-pc
```

In file `/etc/pam.d/common-account`, replace:

```
account required        pam_unix2.so
```

with

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

```
default/:/ # vi /etc/pam.d/common-auth-pc
```

In file `/etc/pam.d/common-auth`, replace:

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

with

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

```
default/:/ # vi /etc/pam.d/common-password-pc
```

In file `/etc/pam.d/common-password`, replace:

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

with

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

```
default/:/ # vi /etc/pam.d/common-session-pc
```

In file `/etc/pam.d/common-session`, 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
```

On the boot root, in the `xtopview` default view, make your site-specific changes to the `/etc/openldap/ldap.conf` or `/etc/ldap.conf`, `/etc/nsswitch.conf`, `/etc/sysconfig/ldap`, `/etc/passwd`, and `/etc/group` files.

Note: Adding the LDAP servers to the local host file allows you to **not** run DNS on the SDB and MDS. The SDB and MDS need access to the LDAP server. You can set up this access through RSIP or NAT; see Configuring Realm-specific IP Addressing (RSIP) on page 206.

```
default/:/ # exit
```

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

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

Note: When following the procedure in the `README.SUSE` file, use the `chkconfig` command instead of the `yast2` run level editor to turn on `quota` and `quotad` services; execute these `chkconfig` commands from `xtopview` in the default view:

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

Then start the services on all service nodes; either reboot the system or execute `/etc/init.d/boot.quota start; /etc/init.d/quotad start` on each service node.

After the quota services have been enabled, for each user you can use standard Linux quota commands to do the following:

- Enable quotas (`quotaon` command)
- Check quotas (`quotacheck` command)
- Set quotas (`edquota` command)

When a quota is exceeded, the quotas subsystem warns users when they exceed their allotted limit, but it allows some extra space 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 commands.

5.4 System-wide Default Modulefiles

The Base-opts modulefile loads two lists of module files: 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 CLE module files to load by default. On the Cray system, the file /etc/opt/cray/modules/Base-opts.default contains the list of CLE module files to load by default.

Additionally, all the module files listed in the file /etc/opt/cray/modules/Base-opts.default.local are loaded. Edit this file to make your site-specific changes.

The /etc/opt/cray/modules/Base-opts.default.local file initially includes the admin-modules module file, which loads a full set of module files. You do not need to manually load the admin-modules module file, unless the you have removed it from the default list. The CLE installation process removes admin-modules module file 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.

5.5 User Access to a Compiler Environment Using Modulefiles

The Modules software utility enables your users to modify their environment dynamically by using *modulefiles*; modulefiles are metafiles containing Tool Command Language (Tcl) code that is interpreted by the Modules software utility.

Each modulefile contains the information needed to configure the shell for an application. After the Modules software utility is initialized, users can modify the environment on a per-module basis using the module command, which interprets modulefiles. Typically, modulefiles instruct the module command to alter or set shell environment variables such as PATH, MANPATH, and others. The modulefile can be shared by many users on a system, and users can have their own collection to supplement or replace the shared modulefiles.

The Cray, PGI, GCC, PathScale, and Intel compilers are available to users through the `PrgEnv-cray`, `PrgEnv-pgi`, `PrgEnv-gnu`, `PrgEnv-pathscale`, and `PrgEnv-intel` modulefiles, respectively.

The `Base-opts`modulefile loads the OS modules in a versioned set that is provided with the CLE release. A user can have only one environment loaded at a time; however, users can add and remove modulefiles from their current environments.

Before beginning to compile programs, the user must verify that the target architecture is set correctly. The target architecture is used by the compilers and linker in creating executables to run on compute nodes. (The target architecture is not necessarily the kernel currently running on compute nodes; as the system administrator, at boot time, you determine the compute node kernel that will run.)

The CNL target is defined automatically at login. If any compute node is running CNL, the target module `xtpe-target-cnl` is loaded and the `XTPE_COMPILE_TARGET` environment is set to `linux`.

To support customer-specific needs, you can create your own modulefiles for a product set for your users; for details, see Appendix F, Creating Modulefiles on page 351.

For more information about the Modules software package, see the `module`(1) and `modulefile`(4) man pages.

5.6 Maintaining `*rc.local` Scripts

The `prgenv` RPM adds a section to the `/etc/bash.bashrc.local` and `/etc/csh.cshrc.local` scripts, which set default modulefiles to be loaded. `##BEGIN` and `##END` tags delimit the contents of this section. These scripts have clearly delimited sections for operating system changes. A CLE upgrade modifies these sections in place, maintaining any local changes you have made outside of the delimited block and, more importantly, the order of the blocks within the file.

The default programming environment (PE) is set to `PrgEnv-pgi` in the `PE-set-up` block, which should not be edited by users.

To change the default PE, and add more PE user defaults, add appropriate instructions to the `SITE-set-up` block. For example, if you want to change the default PE to `PrgEnv-cray`, the default link to `dynamic`, and the CPU target to `xyz`, add this to the `SITE-set-up` block:

In `bash.bashrc.local`:

```
##BEGIN SITE-set-up ADD SITE DEFAULTS HERE
# Site specific set up in this section.
module swap PrgEnv-pgi PrgEnv-cray
module add xtpe-xyz
export XTPE_LINK_TYPE=dynamic
##END SITE-set-up
```

In csh.cshrc.local:

```
##BEGIN SITE-set-up ADD SITE DEFAULTS HERE
# Site specific set up in this section.
module swap PrgEnv-pgi PrgEnv-cray
module add xtpe-xyz
setenv XTPE_LINK_TYPE dynamic
##END SITE-set-up
```

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, so make sure new instructions do not conflict with the ones in the PE-set-up block.

5.7 Using the pam_listfile Module in the Shared Root Environment

The Linux pam_listfile Pluggable Authentication Module (PAM) may 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 (see Caution in Login Nodes on page 38).

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

Example 54. Creating a pam_listfile list file

This example assumes you have created an empty pam_listfile called /var/../pam_listfile_authorized_users_list. It adds authorized users to it.

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

user1
user2
...
```

Example 55. Adding a line to /etc/pam.d/sshd to enable pam_listfile

Edit the pam.d/sshd file to include an alternative path for file=.

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

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

If you need nodes to have different pam_listfile list files, create the list files and specialize the PAM configuration files (such as pam.d/sshd) to point to them.

5.8 `ulimit` Stack Size Limit

The login environment defaults to the kernel default stack size limit. To set up the default user environment to have an unlimited stack size resource limit, add the following to `/etc/profile.local` in the shared root:

```
ulimit -Ss unlimited
```

5.9 Stopping a User's Job

This section describes how to stop a user's job.

5.9.1 Stopping a CNL Job Running in Interactive Mode

If the job is running on a CNL compute node in interactive mode (through `aprun`), perform the following procedure.

Procedure 22. Stopping a CNL job running in interactive mode

- Use the `apkill -`*signal apid* command to send 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.

5.9.2 Stopping a Job Running Under a Batch System

To stop a job that is running under a batch system, see the documentation provided by your batch system vendor.

Example 56. Stopping a job running under PBS Professional

If the job is running under PBS Professional, use the `qdel` command and name the job.

To terminate job 104, type:

```
% qdel 104
```

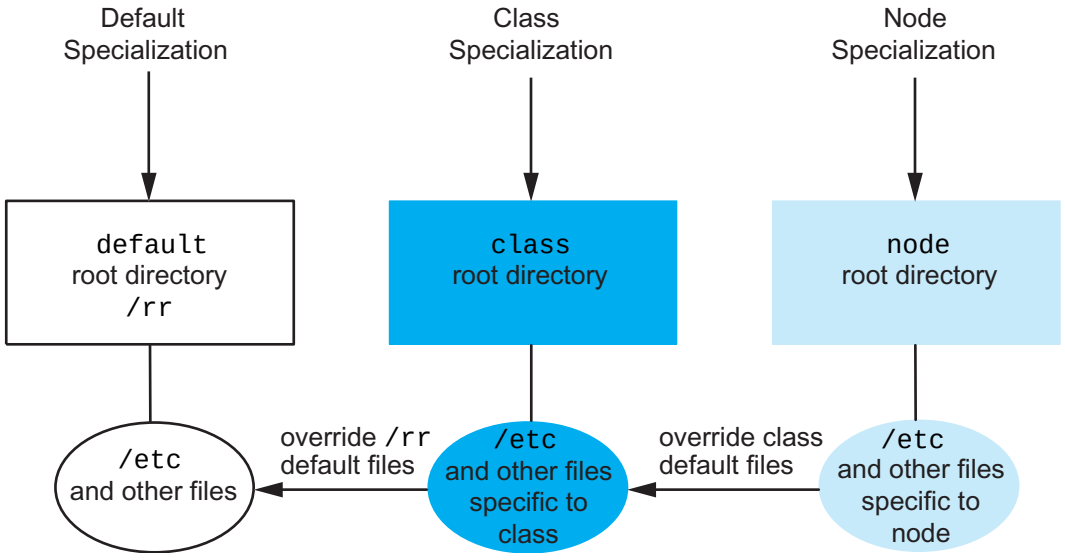
6.1 Configuring the Shared-root File System on Service Nodes

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. You can specialize one or more files for an individual node or for a class (type) of nodes, such as login. You must be root user to configure the shared-root file system in this manner. You can specialize files when you install the system or at a later time.

The hierarchical structure of the specialized files is shown in Figure 2. Node specialization is more specific than class specialization. Class specialization is more specific than default specialization. Generally, about 98% of the service nodes use the default version of the shared root.

Figure 2. Types of Specialization



6.1.1 Specialization

You specialize files when you 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, you might specialize files when differences exist in hardware, network configuration, or boot scripts or when there are services that run on a single node. You can also specialize files 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 (see Managing System Configuration With the `xtopview` Tool on page 129) and then specialize your files (see Specializing Files on page 131).

Table 2 lists files and directories that you can specialize by class and the reasons to do so. Table 3 lists files and directories that you can specialize by node and the reasons to do so. In these tables, * refers to "wildcard" characters that represent no characters or any number of characters.

Table 2. File Specialization by Class

File or Directory	Reason for Specialization
`/etc/auditd.conf`	Cray Audit configured on login nodes.
`/etc/audit.rules`	Cray Audit configured on login nodes.
`/etc/cron*`	Different classes need custom crontabs.
`/etc/fstab`	I/O nodes need to mount other file systems.
`/etc/hosts.{allow,deny}`	Must restrict logins on login nodes.
`/etc/init.d/boot.d/*`	Different classes have different start-up scripts enabled.
`/etc/init.d/rc*/`	Different classes have different start-up scripts enabled.
`/etc/issue`	Different classes have different messages.
`/etc/modprobe.conf`	I/O and login nodes have different hardware.
`/etc/motd`	Different classes have different messages.
`/etc/pam*`	Authentication is class-specific.
`/etc/profile.d/*`	Login nodes have custom environments.
`/etc/resolv.conf`	Hosts that interact with external servers need special resolver configurations.
`/etc/security/*`	Authorization and system limits are class-specific.
`/etc/sysconfig/network/*`	I/O and login nodes need custom network configuration.

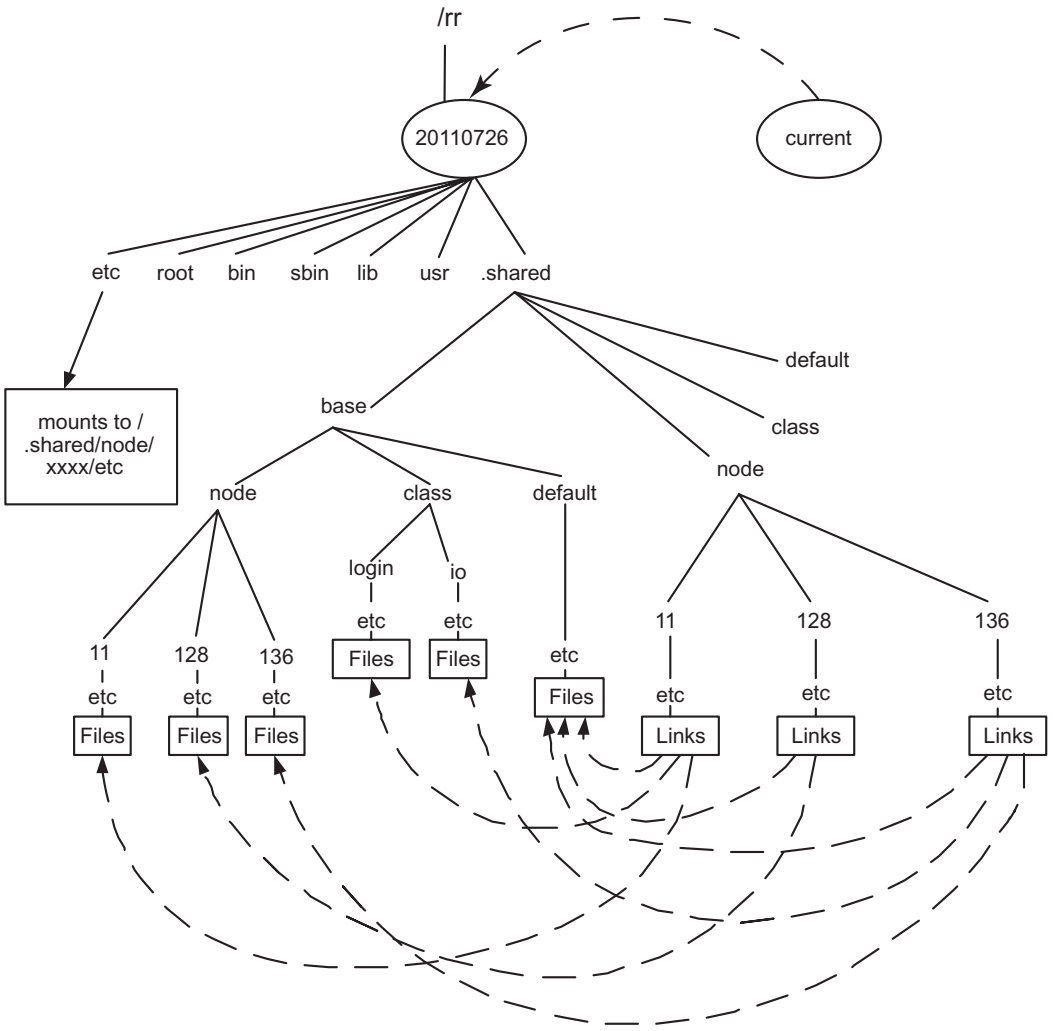
Table 3. File Specialization by Node

File or Directory	Reason for Specialization
`/etc/cron*`	Certain service nodes, such as `sdb` and `syslog`, need custom crontabs.
`/etc/ntp.conf`	A node that runs an NTP server needs a different configuration than NTP clients.
`/etc/sysconfig/network/*`	Each network node should have a different IP address.
`/etc/syslog-ng/syslog-ng.conf.in`	A node that runs a syslog server needs a different configuration than syslog clients.
`/etc/ssh/*key*`	Use when sharing keys across systems is unacceptable.

6.1.2 Visible Shared-root File System Layout

Figure 3 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 20110621). 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 you are viewing it from, 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 `/ufs/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 you do not 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

6.1.3 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 you can look at the file system (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.

Example 57. Shared-root links

The path of the link shows the type of specialization for the file.

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
```

6.1.4 Working with the Shared-root File System

CLE commands shown in Table 4 control and monitor the shared-root file system. For more information, refer to the sections noted and the related man pages.

Table 4. Shared-root Commands

Command	Function
`xtopview`	View file layout from the specified node (see Managing System Configuration With the `xtopview` Tool on page 129).
`xtopcommit`	Record file specialization before leaving `xtopview` shell (see Updating Specialized Files From Within the `xtopview` Shell on page 131).
`xtspec`	Specialize; create a directory structure that links files to non-default files (see Specializing Files on page 131).
`xthowspec`	Determine the type of specialization (see Determining which Files are Specialized on page 133).
`xtverifyshroot`	Verify that node-specialized and class-specialized files are linked correctly (see Checking Shared-root Configuration on page 135).
`xtverifyconfig`	Verify that start/stop links generated by tools such as `chkconfig` are consistent across all views of the shared root. You can configure `xtopview` to invoke `xtverifyconfig` automatically; this is the preferred usage. `xtverifyconfig` is not intended for direct use. (See Verifying the Coherency of `/etc/init.d` Files Across All Shared Root Views on page 135.)
`xtverifydefaults`	Verify and fix inconsistent system default links within the shared root. You can configure `xtopview` to invoke `xtverifydefaults` automatically; this is the preferred usage. `xtverifydefaults` is not intended for direct use. (See Verifying the Coherency of `/etc/init.d` Files Across All Shared Root Views on page 135.)
`xtcloneshared`	Create a directory structure for a new node or class based on an existing node or class (see Cloning a Shared-root Hierarchy on page 136).
`xtnce`	Modify the class of a node or display the current class of a node (see Changing the Class of a Node on page 136).

Command	Function
`xtunspec`	Remove specialization (see Removing Specialization on page 137).
`xtoprlog`	Display RCS log information for shared root files (see Displaying RCS Log Information for Shared Root Files on page 137).
`xtopco`	Check out (restore) RCS versioned shared root files (see Checking Out an RCS Version of Shared Root Files on page 138).
`xtoprdump`	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 (see Listing Shared Root File Specification and Version Information on page 139).
`xtoparchive`	Perform operations on an archive of shared root configuration files (see Performing Archive Operations on Shared Root Files on page 139).

6.1.4.1 Managing System Configuration With the `xtopview` Tool

The `xtopview` tool manages the files in the shared-root file system. The command runs on the boot node. You specify the view of the system you want, such as from a particular node, when you invoke the command. The system appears as if you were logged in directly to the location you specify; that is, the files that are specialized for that node appear in the `/etc` directory. You can specify location by node ID or hostname.

Changes you make within `xtopview` are logged to a revision control system (RCS) file. When you exit the shell, you are prompted to type a message about each change you have made. Use the `c` command to comment the work you have done in `xtopview`. This information is saved in the Revision Control System (RCS) files.

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: If you do not want the changes you have made in your `xtopview` session, you must invoke any necessary commands to undo them. There is no automatic way to back out.

Cray recommends that you configure the shared root from within the `xtopview` shell. Only operations that take place within the `xtopview` shell are logged. If you choose to use specialization commands outside of `xtopview`, they are not logged. Logs reside in the `/rr/current/.shared/log` path relative to the boot node.

New files that are created from within the `xtopview` shell automatically have the specialization that is associated with the view under which you are operating. You do not have to specialize them. If you want a file to be used by all service nodes, create the file in the default view.

Example 58. Starting the `xtopview` shell for a node

To start the `xtopview` shell for node 131, type:

```
boot:~ # xtopview -n 131
node/131/: #
```

Example 59. Starting the `xtopview` shell for a class of nodes

To start the `xtopview` shell for the login nodes, type:

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

Note: If you are using the `emacs` editor within the `xtopview` shell, you may see the following message:

```
Symbolic link to RCS-controlled source file; follow link [yes or no]?
```

The symbolic link points to a real file in the `/.shared` directory. If you choose `yes`, you edit the file directly. If you choose `no`, you replace the symbolic link with a real file, but when you exit the `xtopview` shell, the file is moved to the correct location and the link is recreated. The difference is that if you are editing the real file, modifications appear immediately in other views.

Example 60. Starting the `xtopview` shell for a directory other than `/rr/current`

To start the `xtopview` shell in a directory other than `/rr/current`, which is a link to the most current directory, type:

```
boot:~ # xtopview -r /rr/20050901
default/:/ #
```

Example 61. 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, you can direct `xtopview` to access the `/etc/opt/cray/sdb/node_classes` file by selecting the `-x` option.

Example 62. Starting `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
```

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

6.1.4.2 Updating Specialized Files From Within the `xtopview` Shell

When you exit the `xtopview` shell (see Managing System Configuration With the `xtopview` Tool on page 129), changes you make are propagated to the shared-root file system. Use the `xtopcommit` command to immediately update the shared root with modifications you have made. You do not need to leave the `xtopview` shell.

Example 63. Updating 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
```

6.1.4.3 Specializing Files

Specifying a view with the `xtopview` command does not automatically specialize existing files. To specialize existing files, you must 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 (see Figure 2), which is a more restrictive class.

If you are not within `xtopview` (see Managing System Configuration With the `xtopview` Tool on page 129) when you specialize a file, you must 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.

Note: The `xtspec` command can be used only to specify files or directories residing in or under the `/etc` directory. If you attempt 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 you do not specialize a file, the default specialization level is based on the current view if you are running in the `xtopview` shell (see Managing System Configuration With the `xtopview` Tool on page 129) or on the default view if you are operating outside the `xtopview` shell.

Classes are defined in the `node_classes` file (see Class Name on page 57).

Procedure 23. Specializing a file by class login

1. To specialize the file `/etc/dhcpd.conf` by the class of login nodes, enter the login shell.

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

2. Specialize the selected file.

   ```
   class/login:~ # xtspec /etc/dhcpd.conf
   ```

3. Edit `/etc/dhcpd.conf` if it is the default copy of the file. If you have pointed to a unique copy of the file 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.

Procedure 24. Specializing a file by node

1. To specialize the file `/etc/motd` for node `11`, enter the login shell.

   ```
   boot:~ # xtopview -n 4
   ```

2. Specialize the selected file.

   ```
   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.

Procedure 25. Specializing a file by node without entering `xtopview`

- Specify the root path and view mode.

```
boot:~ # xtspec -r /rr/current -V -n 4 /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.

After you have specialized nodes, you can unspecialize them (see `xtunspec` command, Removing Specialization on page 137) or determine how they are specialized (see `xthowspec` command Determining which Files are Specialized on page 133). You can also view or change the class type of a particular node (see `xtnce` command, Changing the Class of a Node on page 136).

You can use specialization commands only from the boot node. You must be root user to use them. For more information, see the `shared_root`(5) and `xtspec`(8) man pages.

6.1.4.4 Determining which Files are Specialized

The CLE `xthowspec` command displays how the files in a specified path are specialized. For example, you might use this command to examine restrictions on login nodes.

The `xthowspec` command has the form:

`xthowspec` [*options*] *path*

You 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; for example, for default view this is default; for class view options are class or default.

Procedure 26. Finding files in `/etc` that are specialized by a node

1. Enter the `xtopview` shell for the node.

```
boot:~ # xtopview -n 4
```

2. Use the `xthowspec` command for the node.

```
node/4/: # xthowspec -t node /etc
node:4:/etc/fstab.h:node
node:4:/etc/hostname:node
```

Or, outside the `xtopview` shell:

```
boot:~ # xthowspec Â–r /rr/current -t node Â–n 4 /etc
node:4:/etc/fstab.h:node
node:4:/etc/hostname:node
```

Example 64. Finding files in `/etc` that are specialized by class

To find all files specialized by class, type:

```
class/login:~ # xthowspec -r /rr/current Â–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
...
```

Example 65. Finding specialization of a file on a node

To find the specialization of `/etc/dhcpd.conf` on node 4, type:

```
boot:~ # xtopview -n 4
node/4/: # xthowspec /etc/dhcpd.conf
node:4:/etc/dhcpd.conf:default
```

Example 66. Finding nodes on which a file is specialized

To find the nodes that the `/etc/fstab` is specialized on, type:

```
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
```

To examine specialization outside the `xtopview` shell, you must type the full path name.

Example 67. Finding specialization of a file on a node without invoking the `xtopview` shell

To find the specialization of `/etc/fstab` on node `102`, type:

```
boot:~ # xthowspec -r /rr/current -n 102 /etc/fstab
node:102:/etc/fstab:node
```

Example 68. Finding specialization of files by class without invoking the `xtopview` shell

To find all files that are specialized by class in `/etc` for all nodes, type:

```
boot:~ # xthowspec -r /rr/current -N -t class /etc
node:11:/etc/crontab:class
node:1:/etc/crontab:class
```

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

6.1.4.5 Checking Shared-root Configuration

You can check the configuration of the shared-root file system with the `xtverifyshroot` command:

```
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.

Note: You must be in the `xtopview` shell to use the `xtverifyshroot` command.

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

6.1.4.6 Verifying 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.

Note: This is the preferred usage; the `xtverifyconfig` utility is not intended for direct use.

When you use the `chkconfig` utility 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, you 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.

6.1.4.7 Cloning a Shared-root Hierarchy

You can create a directory structure for a new node or class name in the shared-root hierarchy based on an existing node or class with the xtcloneshared command. For more information, see the xtcloneshared(8) man page.

6.1.4.8 Changing the Class of a Node

If you remove nodes, you may need to change the class of the remaining nodes. If you add a login node, you must add it 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
```

Example 69. Finding the class of a node

To identify the class of node 132, type:

```
crayadm@boot:~> xtnce 132
132:login
```

Example 70. Adding a node to a class

Use the xtnce command for the node and specify the class it should be:

```
crayadm@boot:~> xtnce -c login 104
```

You also need to change /etc/opt/cray/sdb/node_classes on the boot node 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.

Note: If you make changes to /etc/opt/cray/sdb/node_classes, you **must** make the same changes 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.

Note: The xtnodeclasses2db command inserts the node-class list into the database, but it does not make any changes to the shared root.

6.1.4.9 Removing Specialization

If you specialized a node or class of nodes and, for example, you want to remove unique start-up scripts from them, you can remove this specialization with the `xtunspec` command:

`xtunspec [`*options*`]` *path*

You can unspecialize files for all nodes and classes (default), for a specified class of nodes or for a particular node. Cray strongly recommends that you unspecialize files from within the `xtopview` shell; if you do not unspecialize your files from within the `xtopview` shell (see Managing System Configuration With the `xtopview` Tool on page 129), you must also specify the path for the shared root.

Note: You can only use `xtunspec` on the boot node.

Example 71. Removing node specialization

To remove all versions of `/etc/fstab` specialized by node, type:

```
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`.

Example 72. Removing class specialization

To remove all versions of `/etc/fstab` that are specialized by, for example, class I/O (`io`), type:

```
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. To remove a file specialized by node, you must use the `xtunspec` command on the node (see Example 71).

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

6.1.4.10 Displaying 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. Execute the `xtoprlog` command from any service node.

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.

Example 73. Printing the latest version of a file

Use the `xtoprlog --version` option to print the latest version (revision) number of a specified file:

```
crayadm@nid00004:~ xtoprlog --version /etc/fstab
1.7
```

Example 74. Printing the RCS log for /etc/fstab in the node 3 view

Use the `xtoprlog -n` option to specify the `/etc/fstab` node view RCS log to print:

```
crayadm@nid00004:~ 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
...
```

Example 75. Displaying differences between two versions of the /etc/fstab file

Use the `xtoprlog -x` option with the `xtoprlog -r` option to display the differences between the current version of `/etc/fstab` and version 1.3:

```
crayadm@nid00004:~ 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
```

6.1.4.11 Checking 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.

Example 76. Checking out a version 1.2 copy of /etc/fstab

Use the `xtopco -r` option to specify the version of the file to check out:

```
boot:~ # xtopview
default/:/ # xtopco -r 1.2 /etc/fstab
```

Example 77. Recreating the file link for /etc/fstab to the current view's /etc/fstab file

To recreate the file link only, use the `xtopco --link` option:

```
boot:~ # xtopview
default/:/ # xtopco --link /etc/fstab
```

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

6.1.4.12 Listing 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.

Example 78. Printing specifications for login class specialized files

Use the xtoprdump -n option to specify the node view; set to all for all nodes:

```
boot:~ # xtopview
default/:/ # xtoprdump -n all
```

Example 79. Printing specifications for files modified in the default view and include any warning messages

The following xtoprdump command prints specifications for modified files (-m option) in the default view (-d option), including warning messages (-w option):

```
boot:~ # xtopview
default/:/ # xtoprdump -m -d -w
```

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

6.1.4.13 Performing 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.

Example 80. Adding files specified by specifications listed in `specfile` to an archive file

Use the following `xtoparchive` command to add files specified by the specifications listed in `specfile` to the archive file `archive.20110422`; create the archive file if it does not already exist:

```
boot:~ # xtopview
default/:/ # xtoparchive -a -f specfile archive.20110422
```

Example 81. Listing specifications for files currently in the `archive.20110422` archive file

Use the `xtoparchive -l` command to list specifications for files currently in the archive file `archive.20110422`:

```
boot:~ # xtopview
default/:/ # xtoparchive -l archive.20110422
```

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

6.1.5 Logging Shared-root Activity

All specialization activity is logged in the log file `/.shared/log`, which tracks additions, deletions, and modifications of files. To view the details of your changes, you must access the RCS logs that were created during the `xtopview` session.

Note: If you have exited `xtopview` with `Ctrl-c`, you do not log the operations you performed within the shell, The changes to the system are present nonetheless. This means that if you want to back out of changes, it is not sufficient to exit `xtopview`. You must submit the commands to undo what you have done.

6.2 PBS Professional Licensing Requirements for Cray Systems

The licensing scheme for PBS Professional uses a central license server to allow licenses to float between servers. The PBS server and scheduler are run on the service database (SDB) node, therefore, network connectivity must exist between the license server and the SDB node. For information about network configuration options for PBS, see Appendix G, PBS Professional Licensing for Cray Systems on page 355.

6.3 Disabling Secure Shell (SSH) on Compute Nodes

By default, the SSH daemon, `sshd`, is enabled on compute nodes. To disable `sshd` follow this procedure.

Procedure 27. Disabling SSH daemon (sshd) on CNL compute nodes

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; you must specify the *xtrelease* that is currently installed on the system set that you are using and located in the `CLEmedia` directory.

   ```
   smw:~ # /home/crayadm/install.xtrelease/CLEinstall
   --upgrade --debug \
   --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 *Installing and Configuring Cray Linux Environment (CLE) Software*.

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 Procedure 2 on page 65.

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
   ```

6.4 Modifying SSH Keys for Compute Nodes

The `dropbear` RPM is provided with the CLE release. Using `dropbear` SSH software, you can supply and generate site-specific SSH keys for compute nodes in place of the keys provided by Cray.

Procedure 28. Using `dropbear` to generate site-specific SSH keys

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. To generate a `dropbear` compatible RSA key, type:

```
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 AAAAB3NzaC1yc2EAAAADAQABAAAAgwCQ9ohUgsrrBw5GNk7w2H5RcaBGajmUv8XN6fxg/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
crayadm@smw:~/dropbear_ssh_keys>
```

To generate a `dropbear` compatible DSS key, type:

```
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 AAAAB3NzaC1kc3MAAACBAMEkThlE9N8iczLpfg0wUtuPtPcpIs7Y4KbG3Wg1T4CAEXDnfMCKSyuCy
21TMAvVGCvYd80zPtL04yc1eUtD5RqEKy0h8jSBs0huEvhaJGHx9FzKfGhWi1ZOVX5vG3R+UCOXG+71wZp3LU
yOcv/U+GWhalTWpUDaRU81MPRLW7rnAAAAFQCEqnqW61bouSORQ52d+MRiwp27MwAAAIEAho69yAfGrNzxEI/
kjyDE5IaxjJpIBF262N9UsxleTX6F65OjNoL84fcKqlSL6NV5XJ5O00SKgTuVZjpXO913q9SEhkcI0Zy0vRQ8
H5x3osZZ+Bq20QWof+CtWTqCoWN2xvne0NtET4lg81qCt/KGRq1tY6WG+a01yrvunzQuafQAAACASXvs8h8AA
EK+3TEDj57rBRV4pz5JqWSlUaZStSQ2wJ3Oy1pIJIhKfqGWytv/nSoWnr8YbQbvH9k1BsyQU8sOc5IJyCFu7+
Exom1yrxq/oirfeSgg6xC2rodcs+jH/K8EKoVtTak3/jHQeZWijRok4xDxwHdZ7e3l2HgYbZLmA5Y=
Fingerprint: md5 cd:a0:0b:41:40:79:f9:4a:dd:f9:9b:71:3f:59:54:8b
crayadm@smw:~/dropbear_ssh_keys>
```

4. As `root`, copy the SSH keys to the boot image template.

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

```
crayadm@smw:~/dropbear_ssh_keys> su root
```

For the RSA key:

```
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:

```
smw:/home/crayadm/dropbear_ssh_keys # cp -p ssh_host_dss_key.db \
/opt/xt-images/templates/default/etc/ssh/ssh_host_dss_key
```

5. Update the boot image to include these changes; follow the steps in Procedure 3 on page 66.

6.5 Configuring the System Environmental Data Collector (SEDC)

To configure the System Environmental Data Collector (SEDC), which collects data about internal cabinet temperatures, cooling system air pressures, critical voltages, etc., see *Using and Configuring System Environment Data Collections (SEDC)*.

6.6 Configuring Optional RPMs in the CNL Boot Image

You can configure which optional RPMs are installed into the CNL boot image for your system in one of two ways. First, several parameters are available in the `CLEinstall.conf` file to control whether specific RPMs are included during installation or upgrade of your system software. When you edit `CLEinstall.conf` prior to running `CLEintall`, set the `CNL_` parameters to either **`yes`** or **`no`** to indicate which optional RPMs should be included in your CNL compute node boot images. For example, to include all optional RPMs, change the following lines.

```
CNL_audit=yes
CNL_csa=yes
CNL_dvs=yes
CNL_ntpclient=yes
CNL_rsip=yes
CNL_cpr=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 Cray Audit, CSA, DVS, and RSIP RPMs, change the following lines.

Note: If you make changes to `/var/opt/cray/install/shell_bootimage_`*label*`.sh` directly, it is important that you make similar changes to the `CLEinstall.conf` file in order to avoid unexpected configuration changes during update or upgrade installations.

```
CNL_AUDIT=y
CNL_CSA=y
CNL_DVS=y
CNL_RSIP=y
```

6.7 Configuring 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.

Procedure 29. Adjusting the memory control group limit

You adjust the memory control group limit using one of two methods:

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 Preparing a Service Node and Compute Node Boot Image on page 64 to ensure the new value is used whenever the new boot image is used.

   c. Reboot the compute nodes.

   d. To ensure the change is not lost during an upgrade of CLE, copy the modified compute node `/etc/opt/cray/rca/rcad_svcs.conf` file into the default template directory in `/opt/xt-images/templates/`.

2. Alternatively, set the memory control group limit using the `mcgroup` option with the `apmgr` command while the compute node is booted. However, 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.

Procedure 30. Disabling 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-cnl` and set the `cgroup_disable` parameter to `memory`:

   ```
   cgroup_disable=memory
   ```

3. Rebuild the compute node boot image as detailed in Preparing a Service Node and Compute Node Boot Image on page 64.
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 happen run low on memory and memory control groups are disabled. However, most programs will not see this condition it is highly dependent on application and site configurations.

6.8 Configuring Cray Enhanced Linux Security Features

This section describes Cray extensions to Linux security auditing utilities and the `cray_pam` PAM module for logging failed login attempts.

6.8.1 Security Auditing and Cray Audit Extensions

Cray Audit is a set of Cray specific extensions to standard Linux security auditing. When the Cray Audit is configured, separate logs are generated for each audited node on the Cray system. Cray specific utilities simplify administration of auditing options and log files across a large number of nodes. For more information about standard Linux security auditing, see the following website: http://www.novell.com/linux.

Cray Audit includes the following components:

- **Cluster option to enable Cray Audit.** The `/etc/auditd.conf` includes a Cray specific option called `cluster` which, when configured on, will enable Cray Audit extensions. The standard Linux `auditd` daemon has been enhanced to implement this configuration option. The clustered configuration provides a mechanism to collect audit data on many nodes and store the data in a central location. The configuration script creates a separate directory for each node and names and manages the auditing log file in the same way as on a single-node system. This includes tracking log size, responding to size-related events, and rotating log files.

  Caution: If you run Linux security auditing on a Cray system without Cray Audit extensions, auditing data from the various nodes collide and generate a corrupt audit log. Because of this, Cray Audit extensions are enabled by default when Linux auditing is configured on.

  Note: The cluster option should **not** be used when auditing a boot node.

- `xtauditctl` **command.** The `xtauditctl` command distributes `auditctl` administrative commands to CNL compute nodes on the system. This command traverses a list of all running compute nodes and invokes commands that deliver a signal to the audit daemons on each node. This utility allows an administrator to apply configuration changes without having to restart every node in the system. For more information see the `xtauditctl`(8) and `auditctl`(8) man pages.

- `xtaumerge` **command.** The `xtaumerge` command merges clustered audit logs into a single log file. When you use this tool to generate a single audit file, you can also use Linux audit tools to report on and analyze system-wide audit data. An additional benefit is that `xtaumerge` maintains compatibility with the Linux audit tools; you can move audit data to another Linux platform for analysis. For more information see the `xtaumerge`(8) man page.

  Note: When you run `xtaumerge`, the resulting merged data stream loses one potentially useful piece of information: the node name of the node on which the event originated. In order to maintain compatibility with standard Linux utilities, the merged audit log does not include this information. Use Linux audit utilities directly on the per-node log files to find a specific record if you require that level of information.

- **ALPS interface to security auditing.** For Cray systems with CNL compute nodes, the Application Level Placement Scheduler (ALPS) supports security auditing functionality. ALPS instantiates an application on behalf of the user on specific compute nodes. After instantiating the application, the ALPS interface calls the auditing system to begin auditing the application. At job start and end, auditing system utilities write the audit record to the audit log.

By default, Cray Audit extensions are enabled but will have no impact until Linux security auditing is configured on. Linux security auditing is configured off by default. Follow Procedure 31 on page 147 to configure Cray Audit and Linux security auditing to audit boot, login and CNL compute nodes. This procedure will direct you to edit the `/etc/auditd.conf` and `/etc/audit.rules` files and define your audit configuration based on site-specific requirements. The file `/usr/share/doc/packages/audit/sample.rules` describes a sample rule set. Once you have established these configuration files, you can make temporary changes to your audit configuration using `xtauditctl` and standard Linux `auditctl` command options. For more information see the `xtauditctl`(8) and `auditctl`(8) man pages.

Cray recommends that you configure auditing to use a Lustre file system to hold the audit log files. Follow Procedure 31 on page 147, to specify the Lustre file system by setting `log_file =` *lustre_pathname*. For more information on specific Lustre file system requirements to run Cray Audit, see Lustre File System Requirements for Cray Audit on page 149.

Procedure 31. Configuring Cray Audit

By default, Linux security auditing is disabled and Cray Audit extensions are enabled. Follow these steps to define your site-specific auditing rules and enable standard Linux auditing.

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

1. Follow these steps to edit the auditing configuration files in the compute node image and enable auditing on CNL compute nodes.

   a. Copy the `auditd.conf` and `audit.rules` configuration files to the template directory so that modifications are retained when new boot images are created in the future.

```
smw:~# cp /opt/xt-images/xthostname-XT_version/compute/etc/auditd.conf \
/opt/xt-images/templates/default/etc/auditd.conf
smw:~# cp /opt/xt-images/xthostname-XT_version/compute/etc/audit.rules \
/opt/xt-images/templates/default/etc/audit.rules
```

   b. Edit `/opt/xt-images/templates/default/etc/auditd.conf` on the SMW and set the `log_file` parameter. For example, if the mount point for your Lustre file system is `mylusmnt` and you want to place audit logs in a directory called `auditdir`, type the following commands.

```
smw:~# vi /opt/xt-images/templates/default/etc/auditd.conf
log_file = /mylusmnt/auditdir/audit.log
```

Warning: If you run auditing on compute nodes without configuring the audit directory, audit records that are written to the local ram-disk could cause the ram-disk to fill.

c. Edit `/opt/xt-images/templates/default/etc/audit.rules` on the SMW. Change this file to set site-specific auditing rules for the compute nodes. At a minimum, you should set the `-e` option to `1` (one) to enable auditing.

```
smw:~# vi /opt/xt-images/templates/default/etc/audit.rules
```

Make your changes after the following line; for example:

```
# Feel free to add below this line.  See auditctl man page
-e 1
```

d. Create the following symbolic link.

```
smw:~# mkdir -p -m 755 /opt/xt-images/templates/default/etc/init.d/rc3.d
smw:~# cd /opt/xt-images/templates/default/etc/init.d/rc3.d
smw:/opt/xt-images/templates/default/etc/init.d/rc3.d # ln -s ../auditd S12auditd
```

e. If you set `CNL_audit=yes` in `CLEinstall.conf` before you ran the `CLEinstall` program, update the boot image by following the steps in Procedure 3 on page 66.

Otherwise, you must first edit the `/var/opt/cray/install/shell_bootimage_`*label*`.sh` script and set `CNL_AUDIT=y` and then update the boot image following the steps in Procedure 3 on page 66.

2. Follow these steps to enable and configure auditing on login nodes.

a. Log on to the boot node and use the `xtopview` command to access all login nodes by class.

```
smw:~# ssh root@boot
boot:~ # xtopview -c login -m "configuring audit files"
```

b. Specialize these files to the `login` class.

```
class/login:/ # xtspec -c login /etc/auditd.conf
class/login:/ # xtspec -c login /etc/audit.rules
```

c. Edit `/etc/auditd.conf` and set the `log_file` parameter. For example, if your Lustre file system is called *filesystem* and you want to place audit logs in a directory called *auditdir*, type the following commands.

```
class/login:/ # vi /etc/auditd.conf
log_file = /filesystem/auditdir/audit.log
```

d. Edit the `/etc/audit.rules` file to set site-specific auditing rules for the login nodes. At a minimum, you should set the `-e` option to `1` (one).

```
class/login:/ # vi /etc/audit.rules
```

Make your changes after the following line; for example:

```
# Feel free to add below this line.  See auditctl man page
-e 1
```

e. Exit `xtopview`.

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

3. You must configure auditing on the boot node to use standard Linux auditing. Follow these steps to turn off Cray audit extensions for the boot node. Configure the boot node to use the default `log_file` parameter in the `auditd.conf` file and set the `cluster` entry to `no`.

   a. While logged on to the boot node, edit the `/etc/auditd.conf` file.

   ```
   boot:~ # vi /etc/auditd.conf
   log_file = /var/log/audit/audit.log
   cluster = no
   ```

   b. Edit the `/etc/audit.rules` file to set site-specific auditing rules for the boot node. At a minimum, you should set the `-e` option to `1` (one).

   ```
   boot:~ # vi /etc/audit.rules
   ```

   Make your changes after the following line; for example:

   ```
   # Feel free to add below this line.  See auditctl man page
   -e 1
   ```

   c. Configure the audit daemon to start on the boot node.

   ```
   boot:~ # chkconfig --force auditd on
   ```

4. Create the log file directory. Log into a node that has the Lustre file system mounted and type the following commands:

   ```
   login:~# mkdir -p /filesystem/auditdir
   login:~# chmod 700 /filesystem/auditdir
   ```

5. Edit the boot automation file to configure your system to start the Cray audit daemon on login nodes by invoking `/etc/init.d/auditd start` on each login node.

6.8.1.1 Lustre File System Requirements for Cray Audit

The audit system stores audit data in a directory tree structure that uses a naming scheme based on the directory name provided by the `log_file` parameter. For example, if you set `log_file` to */lus/audit/audit.log*, the auditing system stores audit data in files named `/lus/audit/`*node_specific_path*`/audit.log`, where *node_specific_path* is a directory structure generated by Cray Audit.

Warning: If you run auditing on CNL compute nodes without configuring the audit directory, audit records are written to the local ram-disk which may consume all your resources and cause data loss.

With the exception of the boot node, each audited node in the system must have access to the Lustre file system that contains the audit directory. Because each node has its own audit log file, sufficient space must be made available to store audit data. You configure the log size in the `/etc/auditd.conf` file. The file system should be large enough to hold at least twice the maximum configured log size, multiplied by the number of log files retained and the number of audited nodes, plus enough space to avoid triggering out of space recovery actions. The following formula can be used to estimate a reasonable file system size:

```
(2 * num_logs * max_log_file * nnodes) + space_left
```

Where:

`num_logs` is the number of log files kept in rotation.

`max_log_file` is the maximum size of a log file in megabytes.

nnodes is the number of audited nodes

`space_left` is the amount of space in megabytes required to avoid out of space recovery actions.

The `num_logs`, `max_log_file`, and `space_left` parameters are set in the `/etc/auditd.conf` file. The default `/etc/auditd.conf` file is shown in Example 82.

Note: This formula assumes that you use the default destination for the output of `xtaumerge`, placing the merged log file and the per-node log files on the same file system. This roughly doubles the size of the disk space needed to hold the audit trail.

Example 82. Default `/etc/auditd.conf` file

```
#
# This file controls the configuration of the audit daemon
#

log_file = /var/log/audit/audit.log
cluster = yes
log_format = RAW
priority_boost = 3
flush = INCREMENTAL
freq = 20
num_logs = 4
#dispatcher = /usr/sbin/audispd
disp_qos = lossy
max_log_file = 5
max_log_file_action = ROTATE
space_left = 75
space_left_action = SYSLOG
action_mail_acct = root
admin_space_left = 50
admin_space_left_action = SUSPEND
disk_full_action = SUSPEND
disk_error_action = SUSPEND
```

6.8.1.2 System Performance Considerations for Cray Audit

With auditing turned off there is no performance impact from this feature. With auditing turned on, system performance is impacted. The performance costs for running Linux audit and the associated Cray extensions vary greatly, depending on the site-defined audit event selection criteria. Auditing of judiciously chosen events, for example login or `su` attempts, do not impact overall system performance. However, auditing of frequently used system calls has a negative impact on system performance because each occurrence of an audited system call triggers a file system write operation to the audit log.

It is the responsibility of the administrator or auditor to design the site security policy and configure auditing to minimize this impact.

6.8.2 Using the `cray_pam` 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 that you configure 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, you must configure the `pam_tally` and `cray_pam` PAM modules. The PAM configuration files provided with the CLE software allow you 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 log in attempt, a per-user counter is updated. When a successful log in 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. You can find an example `faillog.conf` file in `/opt/cray/pam/`*pamrelease-version*`/etc`.

You 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.

Limitations:

- If a login attempt fails, `cray_pam` in 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.

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

Procedure 32. Configuring `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.

   Note: 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.

   Example 83 shows these files after they have been modified to report failed login using 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] (as the FIRST entry)
auth required pam_tally.so [file=alternatepath] (as the LAST entry)
```

Your modified `/etc/pam.d/common-auth` file should look like this:

```
#%PAM-1.0
#
# This file is autogenerated by pam-config. All changes
# will be overwritten.
#
# Authentication-related modules 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
auth    required        pam_env.so
auth    required        pam_unix2.so
auth    required        pam_tally.so
```

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]
```

Your modified `/etc/pam.d/common-account` file should look like this:

```
#%PAM-1.0
#
# This file is autogenerated by pam-config. All changes
# will be overwritten.
#
# Account-related modules 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
```

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]
```

Your modified `/etc/pam.d/common-session` file should look like this:

```
#%PAM-1.0
#
# This file is autogenerated by pam-config. All changes
# will be overwritten.
#
# 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
```

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
boot:~ #
```

Example 83. Modified PAM configuration files configured to report failed login by using an alternate path

If you configure `pam_tally` to save tally information in an alternate location by using the `file=` option, each entry for `cray_pam` must also include the `file=` option to specify the alternate location.

Your modified `/etc/pam.d/common-auth` file should look like this:

```
#
# /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
```

Your modified /etc/pam.d/common-account file should look like this:

```
#
# /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
```

Your modified /etc/pam.d/common-session file should look like this:

```
#
# /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
```

6.9 Configuring cron Services

Optional: 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 Procedure 33 on page 155.

On the shared root, how you configure cron for CLE depends on whether you have set up persistent /var. If you have persistent /var follow Procedure 34 on page 156; if you have not set up persistent /var, follow Procedure 35 on page 157.

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 you must manually enable any scripts you want to use.

Procedure 33. Configuring cron for the SMW and the boot node

Note: 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 on the SMW:

```
smw:~# chkconfig cron
cron on
```

On the boot node:

```
boot:~ # chkconfig cron
cron off
```

2. Use the `chkconfig` command to configure the `cron` daemon to start. For example, to enable `cron` on the boot node, type the following command:

```
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 you do not have a persistent `/var`, Cray recommends that you follow Procedure 35.

Procedure 34. Configuring `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
```

2. 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 your 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 {} \;
```

3. Exit `xtopview`.

```
default/:/ # exit
boot:~ #
```

Procedure 35. Configuring 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 you must turn it on with the `xtservconfig` command to specify which nodes you want the daemon to run on.

1. 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
```

2. 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
```

3. 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.

   a. 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
```

   b. 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 $2 }'  | tr -d [:space:]`
CRONSPOOL=`xtgetconfig CRON_SPOOL_BASE_DIR`
if [ "$MYCLASS" = "login" -a -n "$CRONSPOOL" ];then
  mv /var/spool/cron /var/spool/cron.$$
  ln -sf $CRONSPOOL /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 your 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
boot:~ #
```

f. Use the `xtservconfig` command to enable the `cron` service on a single login node; in this example, node 8.

```
boot:~ # xtopview -n 8
node/8:/ # xtservconfig -n 8 add CRON
node/8:/ # exit
```

The `cron` configuration becomes active on the next reboot. For more information, see the `xtservconfig`(8) man page.

6.10 Configuring the Load Balancer

Optional: 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 `lbnamed` 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 `lbnamed` on the SMW. The `lbnamed` 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 `lbnamed` in `/opt/cray-xt-lbnamed` on the SMW and in `/opt/cray/lbcd` on all service nodes. Configure `lbnamed` by using the `lbnamed.conf` and `poller.conf` configuration files on the SMW. For more information about configuring `lbnamed`, see the `lbnamed.conf`(5) man page.

Procedure 36. Configuring lbnamed on the SMW

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

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

2. Edit the `lbnamed.conf` file on the SMW to define the `lbnamed` host name, domain name, and polling frequency.

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

   For example, if `lbnamed` 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`.

```
$poller_sleep = 30;
$hostname = "mysite-lb";
$lbnamed_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.

```
smw:/etc/opt/cray-xt-lbnamed/ # vi poller.conf
#
# groups
# ------------------------
# login     mycray1-mycray3

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

   Note: Because `lbnamed` 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.

Procedure 37. Installing the load balancer on an external "white box" server

Optional: Install `lbnamed` 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 `lbnamed` service.

1. Shut down and disable `lbnamed`.

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

2. Locate the `cray-xt-lbnamed` RPM on the `Cray CLE 3.1.UP`*nn* `Software` media and install this RPM on the "white box." Do **not** install the `lbcd` RPM.
3. Follow the instructions in the `lbnamed.conf`(5) man page to configure `lbnamed`, taking care to substitute the name of the external server wherever `SMW` is indicated, then enable the service.

6.11 Configuring 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 Chapter 8, Using the Application Level Placement Scheduler (ALPS) on page 247.

6.11.1 `/etc/opt/cray/nodehealth/nodehealth.conf` Configuration File

The NHC configuration file, `/etc/opt/cray/nodehealth/nodehealth.conf`, is located in the shared root. The CLE installation and upgrade processes automatically install and enable NHC software; there is no need for you to change any installation configuration parameters or issue any commands. However, you may edit the `/etc/opt/cray/nodehealth/nodehealth.conf` file to specify which NHC tests are to be run and to alter the behavior of NHC tests (including time-out values and actions for tests when they fail); configure time-out values for Suspect Mode and disable/enable Suspect Mode; or disable or enable NHC.

Note: After you modify the `/etc/opt/cray/nodehealth/nodehealth.conf` file, the changes are reflected immediately the next time NHC runs.

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 your site-specific modifications previously made to the file. However, you should compare your `/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 your `/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 Using `dumpd` to Automatically Dump and Reboot Nodes on page 88, the `dumpd`(8) man page, and the `dumpd.conf` configuration file on the System Management Workstation (SMW).

6.11.2 Configuring 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 the `Free Memory Check`, which starts with an action of `log`.

Important: Also read important test usage information in Guidance About NHC Tests on page 164.

- `Application Exited Check`, which verifies that any remaining processes from the most recent application have terminated.

  The `Application Exited Check` test checks locally on the compute node to see if there are processes running under the ID of the application (APID). If there are processes running, then NHC waits a period of time (set in the configuration file) to determine if the application processes exit properly. If the process does not exit within that time, then this test fails.

  The `Application Exited Check` test is enabled in the default NHC configuration file.

- `Apinit Ping`, which verifies that the ALPS daemon is running on the compute node and is responsive.

  The `Apinit Ping` test queries the status of the `apinit` daemon locally on each compute node; if the `apinit` daemon does not respond to the query, then this test fails.

  The `Apinit Ping` test is enabled in the default NHC configuration file.

- `Free Memory Check`, which examines how much memory is consumed on a compute node while applications are not running. The `Application Exited Check` test must complete before the `Free Memory Check` test begins, ensuring that the application has exited the compute node and is not inflating the memory usage.

  The `Free Memory Check` test is enabled in the default NHC configuration file; however, its action is `log` only.

- `Filesystem`, which ensures that the compute node is able to perform simple I/O to the specified file system. 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 `/etc/fstabs` file will be used and a `Filesystem` 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. You can specify multiple `FileSystem` tests by placing multiple `Filesystem` lines in the configuration file. One line could specify the implicit `Filesystem` test. The next line could specify a specific file system that does not appear in `/etc/fstab`. This could continue for any and all file systems.

If you enable the `Filesystem` test, you can place an optional line (such as, Excluding: `FileSystem-foo`) in the `/etc/opt/cray/nodehealth/nodehealth.conf` configuration file that allows you to list mount points that should **not** be tested by the `Filesystem` test. This allows you to intentionally exclude 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.

The `Filesystem` test is enabled in the default NHC configuration file.

- `Plugin`, which allows scripts and executables not built into NHC to be run, provided they are accessible on the compute node. No plugins are configured by default and the `Plugin` test is disabled in the default NHC configuration file so that local configuration settings may be used.

  For information about writing a plugin test, see *Writing a Node Health Checker (NHC) Plugin Test*.

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 you to specify as many additional tests as you want to write, or the `Filesystem` test, allowing you to specify as many additional file systems as you want. 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 so that you can 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.

6.11.2.1 Guidance About NHC Tests

Guidance about 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. ALPS must guarantee the following two things about the nodes in a reservation before releasing that reservation: 1) ALPS must guarantee that ALPS is functioning on the nodes, and 2) ALPS must guarantee that the previous application has exited from the nodes. Either those two things are guaranteed or the nodes must be set to some state other than `up`. When either ALPS has guaranteed the two things about the nodes or the nodes have been set to some state other than `up`, then ALPS can release the reservation. These two NHC tests guarantee those two things: 1) the `Apinit_ping` test guarantees that ALPS is functioning on the nodes, and 2) the `Application_Exited_Check` test 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`). In the end, either the nodes pass those 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. However, 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 about 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, then 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, then 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.

You can exclude tests by adding an "Excluding: *file system mount point*" line 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. You need to determine the criticality of each file system.

Cray recommends the following:

- Excluding 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 via explicit `Filesystem` tests. You can 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 you have a file system that is mounted as read-write but it has read-only permissions, you should exclude it from the implicit `Filesystem` test. NHC does not support such file systems.

Guidance about 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 fifteen 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 thirty minutes, which is twice the default `RECOVERY_TIME_HARD`. If you change the value of `RECOVERY_TIME_HARD`, you must also correspondingly change the time-out value of the NHC `Lustre` file system test.

The NHC time-out value is specified on this line in the NHC configuration file:

```
# Lustre: <warning time-out> <test time-out> <restart delay>
Lustre: 900 1800 60
```

If you change the `RECOVERY_TIME_HARD` value, you must change the `1800` seconds (thirty minutes) to reflect your new `RECOVERY_TIME_HARD` multiplied by two.

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 you change the NHC `Lustre` file system test time-out value, then you must also change the time-out value for Suspect Mode. 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 thirty-five minutes – thirty minutes for the Lustre test, plus an additional five-minute buffer. For more information about the `suspectend` variable, see Suspect Mode on page 169.

6.11.2.2 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:` *Frequency*

Determines how frequently NHC tests are run on the compute nodes. *Frequency* may be either `errors` or `always`. When the value `errors` is specified, the NHC tests are run only when an application terminates with a non-zero error code or terminates abnormally. When the value `always` is specified, the NHC tests are run after every application termination. If you do not specify the `Runtests` global variable, the implicit default is `errors`.

`Connecttime:` *TimeoutSeconds*

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 `connecttime` has elapsed, then the node is marked `admindown`. If Suspect Mode is enabled and a particular node does not respond after `connecttime` has elapsed, then the node is marked `suspect`. Then, NHC will attempt to contact the node with a frequency established by the `recheckfreq` variable. (For information about Suspect Mode and the `recheckfreq` variable, see Suspect Mode on page 169.)

If you do not specify the `Connecttime` global variable, then the implicit default TCP time-out value is used. NHC will not enforce time-out on the connections if none is specified. The `Connecttime`: *TimeoutSeconds* value provided in the default NHC configuration file is `60` seconds.

The following global variables control the interaction of NHC and `dumpd`, the SMW daemon that initiates automatic dump and reboot of nodes.

`Maxdumps:` *MaximumNodes*

Specifies the number of nodes that fail with the `dump` or `dumpreboot` action that will be dumped. For example, if NHC was checking on 10 nodes that all failed tests with the `dump` or `dumpreboot` actions, only the number of nodes specified by `Maxdumps` would be dumped, instead of all of them.

To disable dumps of failed nodes with `dump` or `dumpreboot` actions, set `Maxdumps: 0`.

`downaction:` *action*

Specifies the action NHC takes when it encounters a down node. Valid actions are `log`, `dump`, `reboot`, and `dumpreboot`.

`downdumps:` *number_dumps*

Specifies the maximum number of dumps that NHC will dump for a given APID, assuming that the `downaction` variable is either `dump` or `dumpreboot`. These dumps are in addition to any dumps taken due to NHC test failures.

6.11.2.3 Standard Variables That Affect Individual NHC Tests

The following four 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.

Note: 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 you use 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.)

Note: This action is good for catching bugs because the state of the processes is preserved and can be dumped at a later time.

Each subsequent action includes the actions that preceded it; for example, the `die` action encompasses the `admindown` and `log` actions.

Note: 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 stay 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.

warntime Specifies the amount of test time, in seconds, that should elapse before `xtcheckhealth` logs a warning message to the console file. This allows an administrator to take corrective action, if necessary, before the *testtime* is reached.

testtime Specifies the total time, in seconds, that a test should run before an error is returned by `xtcheckhealth` and the specified *action* is taken.

restarttime

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.

6.11.3 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 Inc. recommends that you run NHC with Suspect Mode enabled.

If enabled, the default NHC configuration file provided from Cray Inc. uses the following Suspect Mode variables:

`suspectenable:`

Enables Suspect Mode; valid values are `y` and `n`. The `/etc/opt/cray/nodehealth/nodehealth.conf` configuration file provided from Cray Inc. has this variable set as `suspectenable: y`.

`suspectbegin:`

Sets the Suspect Mode timer. Suspect Mode starts after the number of seconds indicated by `suspectbegin` have expired. The `/etc/opt/cray/nodehealth/nodehealth.conf` configuration file provided from Cray Inc. has this variable set as `suspectbegin: 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. The `/etc/opt/cray/nodehealth/nodehealth.conf` configuration file provided from Cray Inc. has this variable set as `suspectend: 2100`.

Considerations when evaluating shortening the length of Suspect Mode:

- You can shorten the length of Suspect Mode if you do not have external file systems, such as Lustre, that NHC would be checking.
- The length of Suspect Mode should 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 Suspect Mode is, 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. The `/etc/opt/cray/nodehealth/nodehealth.conf` configuration file provided from Cray Inc. has this variable set as `recheckfreq: 300`. (For a detailed description about NHC actions during the recheck process, see the `intro_NHC`(8) man page.)

6.11.4 NHC Messages

NHC messages are sent though the `ec_console_log` event 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 `xtcleanup_after`'s call to `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
```

6.11.5 What if a Login Node Crashes While `xtcheckhealth` Binaries are Monitoring Nodes?

If a login node crashes while some `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.

`xtcheckhealth` will be launched with this APID, so it can test for any processes that may have been left behind by that application. This testing only takes place if the `Application_Exited_Check` test is enabled in the configuration file. (The `Application_Exited_Check` test is enabled in the default NHC configuration file.) If the `Application_Exited_Check` test is not enabled, when the recovery action takes place, NHC does not run the `Application_Exited_Check` test and will not check for leftover processes. However, it will run any other NHC tests that are enabled in the configuration file.

Nodes will be 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 you release the compute nodes owned by the crashed login node; see Procedure 38 on page 172. Also, see the `nhc_recovery`(8) man page for a description of the `nhc_recovery` binary usage.

Procedure 38. Recovering from a login node crash when a login node will not be rebooted

1. Create a *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. To list all of the `suspect` nodes in the system and which login nodes own those nodes, execute the following command; use the *nodelistfile* you created in step 1.

   `nhc_recovery -d` *nodelistfile*

3. Parse the `nhc_recovery` output for the NID of the login node that crashed. The file (for example, name it `nodelistfile_computenodes`) of this parsed list should contain all of the compute nodes owned by the crashed login node.

4. If you plan to recover the `suspect` nodes by using option 6 step 6.a below, then complete this step; otherwise, skip this step.

   Note: This recovery method is recommended.

   From the list you created in step 3, create *nodelistfile*s containing nodes that share the same APID to determine the nodes from the crashed login node. For example, your *nodelistfile*s can be named `nodelistfile-APID1`, `nodelistfile-APID2`, `nodelistfile-APID3`, and so on.

5. Using the file you created in step 3, release all of the `suspect` compute nodes owned by the crashed login node. Execute the following command:

   `nhc_recovery -r` *nodelistfile_computenodes*

6. All of these compute nodes have been released in the database. However, they are all still in `suspect` state. Determine what to do with these `suspect` nodes from the following three options:

   a. (Cray recommends this option) Rerun NHC on a non-crashed login node to recover the nodes listed in step 4. Invoke NHC for each *nodelistfile*. Supply as the APID argument 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` (one). 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:

   ```
   xtcleanup_after -s nodelistfile-APID1  APID1  0  1
   xtcleanup_after -s nodelistfile-APID2  APID2  0  1
   xtcleanup_after -s nodelistfile-APID3  APID3  0  1
   .
   .
   .
   ```

   b. These `suspect` nodes can be set to `admindown` and their fate determined by further analysis.

   c. These `suspect` nodes can be set back to `up`, but they were in Suspect Mode for a reason.

6.11.6 Disabling NHC

To disable NHC entirely, set the value of the `nhcon` global variable in the `/etc/opt/cray/nodehealth/nodehealth.conf` file to `off` (the default value in the file provided from Cray Inc. is `on`).

6.11.7 nodehealth Modulefile

To gain access to the NHC functions, the `nodehealth` module must be loaded. The `admin-modules` module file loads the `nodehealth` module, or you can load the `nodehealth` module by executing the following command:

```
module load nodehealth
```

The `Base-opts.default.local` file includes the `admin-modules` module file. For additional information about the `Base-opts.default.local` file, see System-wide Default Modulefiles on page 118.

6.11.8 Configuring the Node Health Checker to Use SSL

Note: Although configuring NHC to use secure sockets layer (SSL) protocol is an optional procedure, Cray recommends that all sites configure NHC to use SSL.

If your site requires authentication and authorization to protect access to compute nodes, you 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 the following files must be setup in the `.nodehealth` directory within the home directory of the root user on both the login node(s) and compute nodes:

- `rsa_key`
- `servercsr`
- `rsa_cert`

The files and the `.nodehealth` directory must have their permissions set to `0700` for maximum security. The same files must be used on both the login and compute nodes. If the files are not identical on both the login and compute nodes, the node health infrastructure will not run and a message similar to the following will be displayed:

```
server authentication failed
node health configuration error
```

Procedure 39. Configuring the Node Health Checker (NHC) to use SSL

Follow these steps to configure NHC to use SSL.

Caution: This process should be performed with all compute nodes down.

Note: The shared root and compute image changes must both be made to ensure they both assume SSL is being used.

1. Create the SSL configuration in the shared root.
2. As user `root`, create SSL key information in the shared root and modify the correct permissions and ownership groups.

```
boot:~ # xtopview -m "Configuring NHC to use SSL"
boot:~ # mkdir /root/.nodehealth
boot:~ # chmod 700 /root/.nodehealth
boot:~ # openssl genrsa -out /root/.nodehealth/rsa_key 1024
boot:~ # openssl req -new -key /root/.nodehealth/rsa_key \
-out /root/.nodehealth/servercsr
    (answer the questions as appropriate - defaults work fine)
boot:~ # openssl x509 -req -days 365 -in /root/.nodehealth/servercsr \
-signkey /root/.nodehealth/rsa_key -out /root/.nodehealth/rsa_cert
    (the certificate expiration time is not used)
boot:~ # chmod 700 /root/.nodehealth/*
boot:~ # exit
```

3. As `root` and on the SMW, update the compute node image with the required libraries. These libraries are `/usr/lib64/libssl.so` and `/usr/lib64/libcrypto.so`. Link and file name structures must be maintained exactly as they exist (note the version number (e.g. `0.9.8`) may be different).

```
lrwxrwxrwx 1 root root     15 Feb 10  09:24  /usr/lib64/libssl.so -> libssl.so.0.9.8
-r-xr-xr-x 1 root root 290728 Feb 10  09:24 /usr/lib64/libssl.so.0.9.8
lrwxrwxrwx 1 root root      18 Feb 10  09:24 /usr/lib64/libcrypto.so -> libcrypto.so.0.9.8
-r-xr-xr-x 1 root root 1464704 Feb 10  09:24 /usr/lib64/libcrypto.so.0.9.8
```

For example, on the SMW you would do the following:

```
smw:~ # mkdir -p /opt/xt-images/templates/default/usr/lib64
smw:~ # cd /opt/xt-images/templates/default/usr/lib64
smw:~ # scp -p root@boot:/rr/current/usr/lib64/libssl.so.0.9.8 .
smw:~ # ln -s libssl.so.0.9.8 libssl.so
smw:~ # scp -p root@boot:/rr/current/usr/lib64/libcrypto.so.0.9.8 .
smw:~ # ln -s libcrypto.so.0.9.8 libcrypto.so
```

Note: The above lines involving the `scp` instructions may be unnecessary because the libraries `libssl.so.0.9.8` and `libcrypto.so.0.9.8` are included as part of the compute node image for CLE 3.1 and beyond. It is **essential**, however, that you check that the version on the compute nodes is the same on the shared root. Compare the sizes of the libraries on the shared root with the compute node image to see that they are the same. Whether or not the libraries are copied from the shared root, the symbolic links still need to be created.

4. As `root` and on the SMW, copy the SSL key data from the shared root to the compute node image by copying the files `/rr/current/root/.nodehealth/rsa_key` and `/rr/current/root/.nodehealth/rsa_cert` that you created in step 2 to the `/opt/xtimages/templates/default/root/.nodehealth` directory.

   For example, if the compute node image was located in the directory `compute`, you would do:

```
smw:~ # cd /opt/xt-images/templates/default/root/
smw:~ # mkdir .nodehealth
smw:~ # chmod 700 .nodehealth
smw:~ # scp -p root@boot:/rr/current/root/.nodehealth/rsa_key .nodehealth
smw:~ # scp -p root@boot:/rr/current/root/.nodehealth/rsa_cert .nodehealth
```

5. Create a new compute node image.
6. Reboot the compute nodes.

6.12 Activating Process Accounting for Service Nodes

The GNU 6.4 accounting package uses Berkeley Software Design (BSD) type process accounting. The GNU 6.4 process accounting is enabled for the Cray system's service nodes. The package name is `acct`; it can be activated using the `acct` boot script. To enable the `acct` boot script, execute the following command on the boot node root and/or shared root:

```
boot:~ # chkconfig acct on
```

The GNU 6.4 process accounting utilities process V2 and V3 format records seamlessly, even if the data is written to the same file. Output goes to an accounting file, which by default is `/var/account/pacct`. The accounting utilities provided for administration use are: `ac`, `lastcomm`, `accton`, and `sa`. The related man pages are accessible by using the `man` command.

6.13 Configuring Failover for Boot and SDB Nodes

The boot node is integral to the operation of a Cray system. Critical services like 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 standby boot and SDB nodes that automatically act as a backup in the event of primary node failure. Failover allows the system to keep running without an interrupt to the system or system services.

Note: The boot-node and SDB node failover features do 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 your system.

The `CLEinstall` program provides the capability to change the default virtual network configuration, however, the default values are acceptable is most cases. For more information, see *Installing and Configuring Cray Linux Environment (CLE) Software* or the `CLEinstall.conf`(5) man page.

6.13.1 Configuring Boot-node Failover

When you configure a secondary (backup) boot node, boot-node failover occurs automatically when the primary boot node fails.

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` (for information about configuring ALPS, see Chapter 8, Using the Application Level Placement Scheduler (ALPS) on page 247)

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 L0 begins the heartbeat checking algorithm to determine if the primary boot node has failed. When the L0 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 each 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.

The following is a list of requirements for configuring your system for boot-node failover:

- The backup boot node must have a Fibre Channel card connected to the boot RAID.

  Note: You must configure the backup boot node in the same zone as the primary boot node.

- You must ensure that the boot RAID host port can see the desired LUNs; for DDN, use the host port mapping; for LSI (Engenio), use SANshare in the SANtricity Storage Manager.
- The backup boot node also requires a Gigabit Ethernet card connected via 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).
- You must enable the STONITH capability 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 your primary boot node is located on a separate blade from services with conflicting STONITH requirements, such as Lustre.

Procedure 40. Configuring boot-node failover

Note: If you configured boot-node failover during your CLE software installation or upgrade (as documented in the *Installing and Configuring Cray Linux Environment (CLE) Software*), 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 your system is shut down before you invoke the `xtcli halt` command.

```
crayadm@smw:~> xtcli halt primary_id, backup_id
```

2. Update the default boot configuration used by the boot manager to boot nodes by using the `xtcli` command:

```
crayadm@smw:~> xtcli boot_cfg update -b primary_id, backup_id -i /bootimagedir/bootimage
```

Or

If you are using `/raw0`, use the following command:

```
crayadm@smw:~> xtcli boot_cfg update -i /raw0
```

If you are using partitions, use the following command to designate the primary boot node and the backup boot node:

```
crayadm@smw:~> xtcli part_cfg update pN -b primary_id,backup_id -i /bootimagedir/bootimage
```

Or

If you are using `/raw0`, use the following command:

```
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 when you do your next upgrade.
4. Boot the boot node.
5. 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. You must ensure that your primary boot node is not assigned to a blade that hosts services with conflicting STONITH requirements, such as Lustre.

a. Use the `xtdaemonconfig` command to determine the current STONITH setting on your primary boot node. For example, if the primary boot node is `c0-0c0s0n1` located on blade `c0-0c0s0`, type this command:

Note: If you have a partitioned system, invoke these commands with the `--partition p`*n* option.

```
crayadm@smw:~> xtdaemonconfig c0-0c0s0 | grep stonith
c0-0c0s0: stonith=false
```

b. To enable STONITH on your primary boot node, execute the following command:

```
crayadm@smw:~> xtdaemonconfig c0-0c0s0 stonith=true
c0-0c0s0: stonith=true
The expected response was received.
```

c. The STONITH setting does not survive a power cycle. You can maintain the STONITH setting for the primary boot node by adding the following line to your boot automation file:

```
# boot bootnode:
lappend actions {crms_exec "xtdaemonconfig c0-0c0s0 stonith=true"}
```

6. Boot the system.

Procedure 41. Disabling boot-node failover

- To disable boot-node failover, type these commands; in this example procedure, the primary boot node is `c0-0c0s0n1` and the backup boot node is `c2-0c1s7n1`.

```
crayadm@smw:~> xtcli halt c0-0c0s0n1,c2-0c1s7n1
crayadm@smw:~> xtcli boot_cfg update -b c0-0c0s0n1,c0-0c0s0n1
crayadm@smw:~> xtdaemonconfig c0-0c0s0 stonith=false
```

6.13.2 Configuring SDB Node Failover

When you configure a secondary (backup) SDB node, 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 following list summarizes requirements to implement SDB node failover on your Cray system.

- Designate a service node to be the alternate or backup SDB node. The backup SDB node requires a QLogic 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 your 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 via boot automation scripts. In a SDB node failover situation, these commands must be invoked on the new (backup) SDB node. For example, you may include commands to start batch system software (if not started via `chkconfig`) or commands to add a route to an external license server.

If at any time you reconfigure your system to use a different primary SDB node, you must enable STONITH for the new SDB node and disable STONITH for the previous node.

For procedures to configure SDB node failover during a CLE software installation, see *Installing and Configuring Cray Linux Environment (CLE) Software*.

6.13.3 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.

6.14 Creating Logical Machines

Logical Machines on page 60, introduces logical machines. 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.

6.14.1 Creating 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.

6.14.1.1 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, you break up the configuration in the x-dimension by grouping a number of chassis together. Thus, you need to know the order in which the chassis are cabled together to define your partitions. 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, you must take your particular configuration and the logical chassis ordering shown in Table 5 into account. For example, if you have a three-cabinet (nine-chassis) topology class 0 system, you can partition your 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.

6.14.1.2 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, you can:

- Group together a consecutive number of even (or odd) cabinets. For example, you can 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, you can 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, you can define three logical machines containing cabinets 0-5; 6,8,10; and 7,9,11, respectively.

6.14.1.3 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 so you 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.

6.14.1.4 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. You 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. You can take a subset of the number of rows to make a partition. Taking corresponding cabinets from all rows leaves the y-dimension torus intact, which in general helps performance.

6.14.2 Configuring 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.

Procedure 42. Configuring a logical machine

- 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.

Example 84. Creating a logical machine with a boot node and SDB node specifying the boot image path

```
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
```

Note: 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, you must specify boot node and SDB node IDs.

For instructions on booting a logical machine, see Booting a Logical Machine on page 186.

For information about configuring boot-node failover, see Configuring Boot-node Failover on page 177.

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.

6.14.3 Booting a Logical Machine

The `xtbootsys --partition p`*N* option enables you to indicate which logical machine (partition) to boot. If you do not specify a partition name, the default partition `p0` (component name for the entire system) is booted. Alternatively, if you do not specify a partition name and you use the `CRMS_PARTITION` environment variable, this variable is used as the default partition name. Valid values are in the form `p`*#*, where *#* ranges from `0` to `31`.

Each file in `/var/opt/cray/log/bootlogs` has a partition name suffix.

To boot a partition, see Booting the System on page 69.

6.15 Updating Boot Configuration

The HSS `xtcli boot_cfg` command allow you to specify the primary and backup boot nodes and the primary and backup SDB nodes for `s0` or `p0` (the entire system).

Example 85. Updating boot configuration

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
```

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 information about configuring failover, see Configuring Failover for Boot and SDB Nodes on page 176.

6.16 Modifying Boot Automation Files

Your boot automation files should be located in `/opt/cray/etc`. There are several automation files; for example, `auto.generic.cnl` and `auto.min.cnl`.

For boot automation scripts, when running CNL on compute nodes, the Lustre file system should start up before the compute nodes.

Note: You 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 *Installing and Configuring Cray Linux Environment (CLE) Software*.

If you use boot automation files, 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.

6.17 Callout to `rc.local` During Boot

The file `/etc/init.d/rc.local` is available for local customization of the boot process. If this file/script is present, it is executed during the compute node boot. This script is executed after `/init`, before any of the scripts in `/etc/init.d/rc3.d` and before `/etc/fstab` is processed.

Note: DRAM machine check exceptions (MCEs) reporting is enabled when a MCE threshold is reached; by default, the MCE threshold is `10` exceptions.

You may want to change the MCE threshold setting to identify marginal hardware. To change the MCE threshold setting, add code to `/etc/init.d/rc.local` in the compute node initramfs. (The `rc.local` file can be created using templates; for information about creating a boot image, see Procedure 3 on page 66.) The change will take effect the next time the system is booted with the new compute node initramfs.

For example, set the threshold to `5` exceptions with the following:

```
limit=5
 for cpu in /sys/devices/system/machinecheck/machinecheck*/ \
 threshold_bank4/misc0 ;do
  echo $limit > ${cpu}/threshold_limit
  echo 0 > ${cpu}/error_count
  echo 1 > ${cpu}/interrupt_enable
 done
```

When a threshold is reached for `bank4`, a log entry is sent to the console. This indicates that the node should be set to `admindown` and that the memory should be replaced as soon as possible. A console log entry would be similar to the following sample:

```
CPU 12: Machine Check Exception:       0 Bank 165: 0000000000000000

 Bank 165 is the key here. To decode:
 reported-bank = K8_MCE_THRESHOLD_BASE + bank * NR_BLOCKS + block;
 K8_MCE_THRESHOLD_BASE=129
 NR_BLOCKS=9
 bank=4
```

In this case, the block is `0`.

The `xtopteronmca` command (for Cray Service personnel use) cannot decode threshold log entries:

```
crayadm@smw:~> xtopteronmca 165 0000000000000000
 Failure converting or invalid MCA bank argument: Invalid argument
```

6.18 Changing the System Software Version to be Booted

Release switching enables you to change between versions and releases of the CLE software that are installed concurrently on the system.

You must boot the operating system to switch CLE releases on your Cray system. You cannot change a release while the mainframe is running. You must reboot each time you change versions; however, you do not need to reboot the SMW.

Minor release switching allows you to select one of the CLE software versions that are installed within a single system set and have the same base operating system release (for example, switching from 4.0.22, back to 4.0.21). Switching is achieved by modifying sets of symbolic links in the file system to refer to the requested release.

Major release switching requires that you have a separate set of disk partitions for each major operating system (for example, switching from 3.1.72, to 4.0.25). 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 your CLE system software. Each system set can be an alternative location for an installation or upgrade of your 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.

6.18.1 Minor Release Switching 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 module files 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.

To accomplish minor release switching, you must set the `bootimage_xtrel` parameter to `yes` in your `CLEinstall.conf` installation configuration file. This will include the release version in your boot image parameters file. If you routinely switch between minor levels, you may find it 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.

Note: The `xtrelswitch` command does not support switching between major release levels, for example from CLE 4.0 to CLE 3.1.

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

6.18.2 Major Release Switching using Separate System Sets

When you use system sets to change the Cray software booted on your Cray system, you boot an entirely different file system. 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 the *Installing and Configuring Cray Linux Environment (CLE) Software* and the `sysset.conf`(5) man page.

Procedure 43. Booting a system set

1. Choose which system set in the `/etc/sysset.conf` file should 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 example, to 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 boot_cfg update -i /raw0
   ```

   Or, if you are working with a partitioned system, p*N*:

   ```
   smw:~ # xtcli part_cfg update pN -i /raw0
   ```

6.19 Changing 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 Updating Database Tables on page 192) that enable you to examine values in the SDB tables and update them when your 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.

6.19.1 Service Database Tables

Table 6 describes the SDB tables, which belong to the XTAdmin database.

Table 6. Service Database Tables

Table Name	Function
`attributes`	Stores compute node attribute information
`lustre_failover`	Updates the database when a node's Lustre failover configuration changes
`lustre_service`	Updates the database when a node's Lustre service configuration changes
`filesystem`	Updates the database when a Lustre file system's configuration changes
`processor`	Stores master list of processing elements and their status
`segment`	For nodes with multiple NUMA nodes, stores attribute information about the compute node and its associated NUMA nodes
`service_cmd`	Stores characteristics of a service
`service_config`	Stores processing element services that the resiliency communication agent (RCA) starts
`service_processor`	Stores nodes and classes (boot, login, server, I/O, or network)
`textfiles`	Stores text
`version`	Stores the database schema version

6.19.2 Database Security

Access to MySQL databases requires a user name and password. The MySQL accounts and privileges are shown in Table 7. For security purposes, Cray recommends changing the account passwords on a regular basis. Default MySQL account passwords and an example of how to change them are documented in *Installing and Configuring Cray Linux Environment (CLE) Software*. To change the default MySQL passwords, also see Changing Default MySQL Passwords on the SDB on page 110.

Table 7. Database Privileges

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

6.19.3 Updating Database Tables

The CLE command pairs shown in Table 8 enable you to update tables in the SDB. One command converts the data into an ASCII text file that you can edit; the other writes the data back into the database file.

Table 8. Service Database Update Commands

Get Command	Put Command	Table Accessed	Reason to Use	Default File
`xtdb2proc`	`xtproc2db`	`processor`	Updates the database when a node is taken out of service	`./processor`
`xtdb2attr`	`xtattr2db`	`attributes`	Updates the database when node attributes change (see Setting and Viewing Node Attributes on page 198)	`./attribute`
`xtdb2nodeclasses`	`xtnodeclasses2db`	`service_processor`	Updates the database when a node's class changes (see	`./node_classes`

Get Command	Put Command	Table Accessed	Reason to Use	Default File
			Changing Nodes and Classes on page 194)	
xtdb2segment	xtsegment2db	segment	For nodes with multiple NUMA nodes, updates the database when attribute information about node changes (see Using the XTAdmin Database segment Table on page 202)	./segment
xtdb2servcmd	xtservcmd2db	service_cmd	Updates the database when characteristics of a service change	./serv_cmd
xtdb2servconfig	xtservconfig2db	service_config	Updates the database when services change (see Changing Services on page 195)	./serv_config
xtdb2etchosts	none	processor	Manages IP mapping for service nodes	none
xtdb2lustrefailover	xtlustrefailover2db	lustre_failover	Updates the database when a node's Lustre failover state changes	./lustre_failover
xtdb2lustreserv	xtlustreserv2db	lustre_service	Updates the database when a file system's failover process is changed	./lustre_serv

Get Command	Put Command	Table Accessed	Reason to Use	Default File
`xtdb2filesys`	`xtfilesys2db`	`filesystem`	Updates the database when a file system's status changes	`./filesys`
`xtprocadmin`	none	`processor`	Displays or sets the current value of processor flags and node attributes in the service database (SDB). The batch scheduler and ALPS are impacted by changes to these flags and attributes.	none
`xtservconfig`	none	`service_config`	Adds, removes, or modifies service configuration in the SDB `service_config` table see Changing Services on page 195)	none

6.19.3.1 Changing Nodes and Classes

The `service_processor` table tracks node IDs (NIDs) and their classes (see Class Name on page 57). 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 you are adding login nodes.

Note: If you make changes to `/etc/opt/cray/sdb/node_classes`, you MUST make the same changes to the node class settings in `CLEinstall.conf` before performing an update or upgrade installation; otherwise, the install utility will complain about the inconsistency.

Note: 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 (see Changing the Class of a Node on page 136).

For more information, see the xtdb2nodeclasses(8) and xtnodeclasses2db(8) man pages.

6.19.3.2 Changing Services

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 you are adding the PBS-MOM service.

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.

Example 86. Identifying services in the service_config table

```
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+
```

Procedure 44. Updating the service_config table when services change

1. Use the xtservconfig command to modify the services that run on each node. The xtservconfig command can be executed from any service node but is normally run from the boot node. You must be user root to make a change using the xtservconfig command. For example, to add the PBS-MOM service, type the following command:

   ```
   boot:~ # xtservconfig -n 12 add PBS-MOM
   ```

2. 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. Type:

      ```
      nid00012:~ # killall -HUP fomd
      ```

   This causes the failover manager to read the database.

   For example, to effect the change for node 12, type:

   ```
   nid00012:~ # pdsh -w 12 "killall -HUP fomd"
   ```

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

For information about providing SSH keys for computes nodes, see Modifying SSH Keys for Compute Nodes on page 141.

6.20 Viewing the Service Database Contents with MySQL Commands

The service database is configured as part of the system installation (see the *Installing and Configuring Cray Linux Environment (CLE) Software*).

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.

Procedure 45. Examining the service databases with MySQL commands

1. As user `crayadm`, on the SDB node, enter the MySQL shell.

```
crayadm@sdb:~> mysql -u basic -p
Enter password: ***********
mysql> show databases;
+-----------+
| Database  |
+-----------+
| XTAdmin   |
1-----------+
1 row in set (0.04 sec)
```

2. Select the `XTAdmin` database.

```
mysql> use XTAdmin;
Database changed
```

3. Display the tables in the `XTAdmin` database.

```
mysql> show tables;
+-------------------+
| Tables_in_XTAdmin |
+-------------------+
| attributes        |
| filesystem        |
| lustre_failover   |
| lustre_service    |
| processor         |
| segment           |
| service_cmd       |
| service_config    |
| service_processor |
| version           |
+-------------------+
10 rows in set (0.00 sec)
```

4. 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)
```

5. 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)
```

6. 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)
```

6.21 Configuring the Lustre File System

For a description of the Lustre file system and how to configure it, see *Managing Lustre for the Cray Linux Environment (CLE)*.

6.22 Configuring Cray Data Virtualization Service (Cray DVS)

For a description of the Cray DVS parallel I/O forwarding service and how to configure it, see *Introduction to Cray Data Virtualization Service*.

6.23 Enabling 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, you must use the `flock` option for `mount`. For more information, see *Managing Lustre for the Cray Linux Environment (CLE)*.

Example 87. Sample mount line from compute node `/etc/fstab`

```
4@gni:136@gni:/filesystem  /lus/nid00004   lustre  rw,flock               0 0
```

6.24 Setting and Viewing 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 `cnselect` command to specify node-selection criteria. The `cnselect` 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.

Note: To meet specific user needs, you can modify the `cnselect` script. For additional information about the `cnselect` script, see the `cnselect`(1) man page.

6.24.1 Setting Node Attributes Using the `/etc/opt/cray/sdb/attr.xthwinv.xml` and `/etc/opt/cray/sdb/attr.defaults` Files

In order for users to select desired node attributes, you 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.

6.24.1.1 Generating 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 contains default values for additional attributes not generated from the `attr.xthwinv.xml` file. The `xtprocadmin`(8) man page includes a description of the attributes fields used by these two files.

- The `/etc/opt/cray/sdb/attr.xthwinv.xml` file contains information to generate the hardware attributes for each node. The hardware attributes listed in the `attr.xthwinv.xml` file apply to all nodes and include:

  `clockmhz` The processor clock speed in megahertz.

  `availmem` The amount of physical memory on the node.

  `coremask` A bit mask that shows which cores are available on a node. For a single-core processor, the value is `1`. For a dual-core processor where both cores are available, the value is `3`. For a quad-core processor where all cores are available, the value is `15`.

  To generate the `/etc/opt/cray/sdb/attr.xthwinv.xml` file, invoke the `xthwinv --x` command on the System Management Workstation (SMW), redirecting the output to the `/etc/opt/cray/sdb/attr.xthwinv.xml` file on the boot node, which is run from the boot node; for example:

  ```
  boot:~ # ssh smw xthwinv -x s0' >
  /etc/opt/cray/sdb/attr.xthwinv.xml
  ```

  For additional information about the `xthwinv` command, see the `xthwinv`(8) man page.

  Note: If you have blades powered down when you want to upgrade your software, see the `CLEinstall`(8) man page for which `xthwinv` file to use during your upgrade process.

- The `/etc/opt/cray/sdb/attr.defaults` file contains default values for additional attributes not generated from the `attr.xthwinv.xml` file.

  In addition to hardware characteristics, you can specify additional attributes in the `/etc/opt/cray/sdb/attr.defaults` file. This file can contain attribute settings for attributes in the following list. The attributes can be applied to all nodes or to a given set of nodes.

  Note: Do **not** 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.

  `archtype` The architecture type: Cray XE =`2`; Default=`2`.

  `osclass` The compute node's operating system: CNL=`2`; Default=`2`. This setting does not impact the booting process; the compute node operating system to boot must still be specified at boot time.

  `pageszl2` The log base 2 of the page size. For example, if `pageszl2` is 12, the page size is 4K (2^{12} = 4096, or 4K). Default=`12`

`label0`
`label1`
`label2`
`label3` Each label is a string of up to 32 characters; the string cannot contain any spaces or shell-sensitive characters.

To create the `attr.defaults` file, copy the example file provided in `/opt/xt-boot/default/etc/opt/cray/sdb/attr.defaults.example`. Edit the file to modify the existing attribute settings and to create site-specific attributes as needed. If you have run `CLEinstall` 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 you 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*.

Note: Spaces are **not** allowed between the comma-separated list of nodes and ranges. When listing multiple attributes for a set of nodes, separate the attributes by a single space, for example,

```
nodeid=234,245-248 archtype=2 osclass=2
```

Example 88. Using node attribute labels to assign nodes to user groups

The following 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
```

6.24.2 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 describe` command:

```
mysql> describe attributes;
+----------+-----------------+------+-----+---------+-------+
| Field    | Type            | Null | Key | Default | Extra |
+----------+-----------------+------+-----+---------+-------+
| nodeid   | int(32) unsigned |      | PRI | 0       |       |
| archtype | int(4) unsigned  |      |     | 2       |       |
| osclass  | int(4) unsigned  |      |     | 2       |       |
| coremask | int(4) unsigned  |      |     | 1       |       |
| availmem | int(32) unsigned |      |     | 0       |       |
| pageszl2 | int(32) unsigned |      |     | 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    |       |
+----------+-----------------+------+-----+---------+-------+
```

The service database command pair `xtdb2attr` and `xtattr2db` enables you to update the `attributes` table in the SDB. For additional information about updating SDB tables using command pairs, see Updating Database Tables on page 192.

6.24.3 Setting Attributes Using the `xtprocadmin` Command

You can use the `xtprocadmin -a` *attr=value* command to temporarily set certain site-specific attributes. Using the `xtprocadmin -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 must be specified in the `attr.defaults` file.

> **Note:** For compute nodes, `xtprocadmin` changes to attributes requires that you restart the `apbridge` daemon on the boot node in order for ALPS to detect changes that the `xtprocadmin` 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; you must be user `root` and execute the `xtprocadmin` command from a service node, and the SDB must be running:

```
boot:~ # xtprocadmin -n 350 -a label1=eedept
```

The output is:

```
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
```

6.24.4 Viewing Node Attributes

Use the `xtprocadmin` command to view current node attributes. The `xtprocadmin -A` option lists all attributes of selected nodes. The `xtprocadmin -a` *attr1*`,`*attr2* option lists selected attributes of selected nodes.

6.25 Using 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. (For more information about XTAdmin database tables, see Changing the Service Database (SDB) on page 191.)

The `segment` table contains the following fields:

- `node_id` is 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` is the number of integer cores per node; in systems with accelerators this only applies to the host processor (CPU).
- `coremask` is 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 0xFFFFFF).

  Note: `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`.

When you change the hardware on the machine, at the next system boot, you must invoke the SMW `xthwinv` utility to populate the `attribute` and `segment` tables. The `/etc/opt/cray/sdb/attr.xthwinv.xml` file, which contains information to generate the hardware attributes for each node, populates the `segment` table. Like the `attributes` table, you must reinitialize the `segment` table at boot. Any changes that you make manually to the table do not persist at reboot. For additional information about using the `xthwinv` command, see Generating the `/etc/opt/cray/sdb/attributes` File on page 198 and the `xthwinv`(8) man page.

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, you can log on to any login node or the SDB node as root and execute the `apmgr resync` command to request ALPS to reevaluate the configuration node segment information and update its information.

Note: If ALPS or any portion of the feature fails in relation to segment scheduling, ALPS reverts to the standard scheduling procedure.

6.26 Configuring Networking Services

6.26.1 Changing the High-speed Network (HSN)

To change your system interconnection network (HSN) address ranges, see *Installing and Configuring Cray Linux Environment (CLE) Software*.

6.26.2 Network File System (NFS)

The Network File System (NFS) version 4 distributed file system protocol is supported. NFS is enabled on 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. If you have a larger Cray system (greater than 50 service I/O nodes), contact your Cray service representative for assistance changing the default setting.

6.26.3 Configuring 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.

Procedure 46. Configuring 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
node/2:/ #
```

2. Create and specialize the following files: `/etc/sysconfig/network/ifcfg-bond0`, `/etc/sysconfig/network/ifcfg-eth0`, and `/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 your 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`

```
node/2:/ # exit
```

For more information on Ethernet link aggregation, see the Linux documentation file `/usr/src/linux/Documentation/networking/bonding.txt`, installed on your system.

6.26.4 Configuring a Virtual Local Area Network (VLAN) Interface

This procedure configures an 802.1Q standard VLAN.

Procedure 47. Configuring a Virtual Local Area Network (VLAN) interface

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
   node/2:/ #
```

2. Create and specialize a file named `/etc/sysconfig/network/ifcfg-vlan`*N*, where *N* is the VLAN ID. The following 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-vlan`*N* file to include your usual network settings. It must also include variable `ETHERDEVICE` that 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. The following 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`.

   ```
   node/2:/ # exit
   ```

6.26.5 Increasing Size of ARP Tables

To increase the size of ARP tables, change the `ARP_OVERHEAD` parameter in the `/etc/sysconfig/xt` file. `ARP_OVERHEAD` should be set to a value greater than the number of hosts in all locally attached external networks; the current default is `0`.

6.26.6 Configuring 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.

Note: 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. You configure 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 your Cray system, RSIP servers must be service nodes with an external IP interface such as a 10-GbE network interface card (NIC). You can configure multiple RSIP servers using multiple service nodes, however only one RSIP daemon (`rsipd`) and one external interface is allowed per service node. Cray requires that you configure RSIP servers as dedicated network nodes.

Warning: Do not configure 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 will disrupt network functionality.

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 your Cray system, first determine which service nodes and associated Ethernet devices will be used to provide RSIP services. Optionally, determine if you will configure service nodes with no external IP interfaces (isolated service nodes) to act as RSIP clients. After selecting RSIP servers based on your machine-specific networking hardware configuration, follow Procedure 48 on page 208 to complete a default RSIP configuration and setup.

Enhancements to the default RSIP configuration require a detailed analysis of specific site configuration and requirements. Contact your Cray representative for assistance in changing the default RSIP configuration.

6.26.6.1 Using the `CLEinstall` Program to Install and Configure RSIP

The `CLEinstall` program can be configured to automatically install RSIP either during a system software upgrade or as a separate event. In either case, you will need to update the compute node boot image and restart your Cray system before RSIP is functional.

When you set the following RSIP-specific parameters in the `CLEinstall.conf` file, `CLEinstall` will load the RSIP RPM, modify `rsipd.conf` and invoke the appropriate `xtrsipcfg` commands to configure RSIP for your system.

`rsip_nodes=`

Specifies the RSIP servers. Populate with the node IDs of the nodes you have identified as RSIP servers.

`rsip_interfaces=`

Specifies the IP interface for each RSIP server node. List the interfaces in the same order specified by the `rsip_nodes` parameter.

`rsip_servicenode_clients=`

Specifies a space separated integer list of service nodes you would like to use for 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.

`CNL_rsip=`**`yes`**

Enables the RSIP client on CNL compute nodes. Optionally, you can edit the `/var/opt/cray/install/shell_bootimage_`*label*`.sh` script and set `CNL_RSIP=`**`y`**.

If you are configuring RSIP for the first time during an installation or upgrade of your CLE system software, follow RSIP-specific instructions in the *Installing and Configuring Cray Linux Environment (CLE) Software*. If you are configuring RSIP as a separate event, follow Procedure 48 on page 208. If you already configured RSIP and want to add isolated service nodes as RSIP clients, follow Procedure 49 on page 210.

For additional information about configuring RSIP, see the `xtrsipcfg`(8), `rsipd`(8), and `rsipd.conf`(5) man pages.

Procedure 48. Installing, configuring, and starting RSIP clients and servers

1. Edit `CLEinstall.conf` for your RSIP configuration. For example, to configure nodes 16 and 20 as RSIP servers with an external interface named `eth0` and node 64 as an RSIP server with an external interface named `eth1`; and node 0 as a service node RSIP client, make these changes.

   ```
   smw:~ # vi /home/crayadm/install.xtrelease/CLEinstall.conf
   rsip_nodes=16 17 64
   rsip_interfaces=eth0 eth0 eth1
   rsip_servicenode_clients=0
   CNL_rsip=yes
   ```

2. Invoke the `CLEinstall` program on the SMW; you **must** specify the *xtrelease* that is currently installed on the system set you are using and located in the `CLEmedia` directory.

```
smw:~ # /home/crayadm/install.xtrelease/CLEinstall --upgrade --debug \
--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 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*.

4. Rebuild the boot image using /var/opt/cray/install/shell_bootimage_*label*.sh, 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 Procedure 3 on page 66.

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 you are 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. For example, if the IP address of the RSIP server is 10.128.0.17 and the isolated service node is nid00000, make these changes.

```
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" }
```

7. Boot your Cray system; for example:

   ```
   crayadm@smw:~> xtbootsys -a auto.xthostname
   ```

   Note: 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. From a login node, log on to an RSIP client node (compute node) and ping the IP address of the SMW or other host external to your Cray system. For example, if *nid00074* is a compute node and *10.3.1.1* is a valid external IP address, type these commands.

```
crayadm@login:~> ssh root@nid00074
root@nid00074's password:
Welcome to the initramfs
# ping 10.3.1.1
10.3.1.1 is alive!
#
```

Procedure 49. Adding isolated service nodes as RSIP clients

You can configure service nodes that are isolated from the network as RSIP clients. This procedure assumes that RSIP is already configured and functional on your Cray system. If you have not installed and configured RSIP on your system, follow Procedure 48 on page 208, 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 of your RSIP servers to provide access for the isolated service node. In this example, we have chosen the RSIP server `nid00016`.

2. Log on to the boot node and invoke `xtopview` in the node view for the RSIP server you have selected; for example:

```
boot:~ # xtopview -n 16
node/16:/ #
```

   Modify `max_clients` in the `rsipd.conf` file to add an additional client for each isolated service node you are configuring. For example, if you configured 300 RSIP clients (compute nodes), change 300 to 301.

```
node/16:/ # vi /etc/opt/cray/rsipd/rsipd.conf
max_clients 301
```

3. 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 you selected in step 1. For example, if the IP address of the RSIP server is `10.128.0.17` and the isolated service node is `nid00023`, type these commands.

```
boot:~ # ssh nid00023
nid00023:~ #  modprobe krsip ip=10.128.0.17
```

4. Edit the boot automation file to start the RSIP client. Using the example from the previous steps, make these changes.

```
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 "nid00023" "root" "modprobe krsip ip=10.128.0.17" }
```

6.26.7 IP Routes for CNL Nodes in the /etc/routes File

You can edit the /etc/routes file in the compute node template image on the SMW to provide route entries for compute nodes. This provides a simple mechanism for you 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.

6.27 Updating the System Configuration After A Hardware Change

When hardware is added to or removed from a Cray system, there are several steps that need to be done to update the configuration of the system. Possible scenarios are:

- Adding new cabinets
- Removing old cabinets
- Adding a blade
- Removing a blade
- Changing the blade type in a slot
- Changing the routing topology

If you have blades powered down when you want to update your hardware configuration, you can run the CLEinstall program, but you must use the xthwinv and xthwinv.xml files that were generated when the full system was available. See the CLEinstall(8) man page for the options needed to specify these files for CLEinstall.

Procedure 50. Updating the system configuration after hardware changes

Note: If you added or removed a service node, also complete Procedure 51 on page 214 so that CLE sets up the shared root for them properly.

1. Run the xtdiscover command to update the system configuration to reflect the changed hardware configuration.

```
smw:~ # xtdiscover
Using ini file '/opt/cray/hss/default/etc/xtdiscover.ini'
```

```
xtdiscover is about to discover new hardware.
This operation may significantly modify the system database.

Please enter 'c' to continue, or 'a' or 'q' to abort [c]: c


Please enter network type (s=SeaStar, g=Gemini, q=quit): g

Is this system a Single-Slot Tester? y/n, q=quit [n]: n

Setting system type to Gemini
Discovering Gemini-based system...

Enter maximum X cabinet size [1-64], q=quit: 5
Enter maximum Y cabinet size [1-16], q=quit: 1
Adding hosts and routes for 5 cabinets...done.

Enter your system's network topology class [0]: 0
Setting topology class to 0

Suspending State Manager for discovery phase 1...
Suspend successful.
Saving current configuration...done.

Discovering cabinets:
[5 out of 5]
Finished waiting for cabinet heartbeats; found 5 out of 5
The following cabinets were not detected by heartbeat:

c2-0   c3-0   c4-0


Found 5 cabinets.

xtdiscover will create a single system partition (p0)
containing all discovered cabinets. If you need to create
additional partitions, use 'xtcli part_cfg add'.

Enter the boot node name [c0-0c0s0n1]:
Enter the SDB node name [c0-0c0s2n1]:
Enter the absolute pathname to the default boot image [/raw0]:

Gathering base cabinet attributes:
[5 out of 5]
Finished gathering cabinet attributes.

Clearing database...done.
Verifying phase 1 configuration...done.
Storing base cabinet data...done.
Resuming State Manager for power-up and bounce...Resume successful.

Discovery Phase 1 of 3 complete.

xtdiscover is about to power on the cabinets.
*** IF YOU NEED TO DISABLE COMPONENTS TO AVOID THEM
*** BEING POWERED ON, PLEASE DO SO NOW USING 'xtcli disable'

Please enter 'c' to continue, or 'a' or 'q' to abort [c]:

Suspending State Manager for discovery phase 2...
Suspend successful.

Loading base component data for discovery phase 2...done.
```

```
Powering on cabinets...
5 cabinets will be powered on:
[5 out of 5]
Cabinets powered on.

Discovering component phase 2 (blade) state:
[480 out of 480]
Finished discovering component phase 2 (blade) state.

Discovering component phase 2 (blade) attributes:
[480 out of 480]
Finished discovering component phase 2 (blade) attributes.
Verifying phase 2 configuration...done.

Summary of blades discovered:
Total:    480    Service:     36  Empty:    0    Disabled:     0

Storing attribute data...done.

Discovery Phase 2 of 3 complete.

Resuming State Manager for bounce...Resume successful.

288 blades should be bounced using the command
in file /opt/cray/hss/default/etc/xtdiscover-bounce-cmd

In a separate window, please bounce the system now to continue discovery.

After bounce completes, enter 'c' to complete discovery
or 'q' or 'a' to abort [c]:

Suspending State Manager for discovery phase 3...
Suspend successful.

Discovering component phase 3 (blade/node) attributes:
[480 out of 480]
Finished discovering component phase 3 attributes.
Verifying configuration...
INFO: 3 newly discovered components were added.
INFO: 836 components in previous configuration were deleted.
INFO: Added the following hardware:
    3 cabinets
           24 slots

INFO: Configuration change details are in /opt/cray/hss/default/etc/xtdiscover-config-changes.diff
done.
Storing component attribute data...done.
Updating component location history...done.

Restarting RSMS daemons for normal operation:
Stopping RSMS services: cm sedc_manager bm dm rm pm nm sm erd        done
Starting RSMS services: erd sm nm pm rm dm bm sedc_manager cm
Flushing and installing cabinet routes...done.
                                                                     done
Done.

Discovery complete
*********** xtdiscover finished ***********
```

2. Update the SDB database to provide ALPS with the new configuration at the next boot.

a. Capture system configuration into the `/etc/opt/cray/sdb/attr.xthwinv` file on the boot root.

   For the entire system, `s0`:

   ```
   smw:~ # xthwinv -x s0 > xthwinv.x.s0.out
   ```

   If the system set is used to boot a partition and not the entire machine, instead of using `s0` on the command line, use p*N* for the partition p*N*.

   ```
   smw:~ # xthwinv -x pN > xthwinv.x.pN.out
   ```

b. Install the files on the boot node root file system; the `attr.xthwinv.xml` file is automatically copied to the boot node at system boot time.

   ```
   smw:~ # mount /dev/sdXX /bootroot0
   smw:~ # cp -p /bootroot0/etc/opt/cray/sdb/attr.xthwinv \
   /bootroot0/etc/opt/cray/sdb/attr.xthwinv.YYYYMMDD
   smw:~ # cp -p xthwinv.s0.out \
   /bootroot0/etc/opt/cray/sdb/xttr.xthwinv
   smw:~ # umount /bootroot0
   ```

Procedure 51. Updating the system configuration after services nodes were added or removed

Some actions need to be done regardless of the function of the service node. At a minimum, the boot node and the SDB node should be booted when doing these steps. If reconfiguring RSIP servers, then all service nodes should be up.

1. Update the `/etc/opt/cray/sdb/node_classes` file on the boot root to add entries for new service nodes and classes or remove old service nodes and classes.
2. 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.
3. After adding a new service node, prepare the node view for it:

   a. Run the `xtcloneshared` command on the boot node in the shared root default view to copy an existing service node that is of similar type.

      ```
      default/:/ # xtcloneshared -n from_node to_node
      ```

   b. If the new service node will be in a new class, copy an existing service class. Run the `xtcloneshared` command on the boot node in the shared root default view to copy an existing service class.

      ```
      default/:/ # xtcloneshared -c from_class to_class
      ```

4. After removing a service node from a certain class, run the `xtcloneshared` command on the boot node in the shared root default view to reset to generic default.

   ```
   default/:/ # xtcloneshared -n to_node
   ```

5. If Lustre service nodes are added or removed, the Lustre configuration will need to be modified. For more information about Lustre configuration, see *Managing Lustre for the Cray Linux Environment (CLE)*.

6. If using RSIP and the new service node will be an RSIP server, then run `CLEinstall` to reconfigure RSIP. For more information, see Configuring Realm-specific IP Addressing (RSIP) on page 206.

   The boot image will need to be rebuilt in step 8.

7. Update the `/etc/hosts` file on the boot root, the shared root default view, and the CNL image. When the boot node is booted, the `bnd` daemon will update the boot root `/etc/hosts` file and copy it to the shared root default view. This file should be copied from the boot root to the templates directory on the SMW and the boot image should be rebuilt. If using partitions, the template directory will be `/opt/xt-images/templates/default-p`*N*, where `p`*N* is the partition number.

```
smw:~ # cp -p /opt/xt-images/templates/default/etc/hosts /opt/xt-images/templates \
/default/etc/hosts.save
smw:~ # scp -p boot:/etc/hosts /opt/xt-images/templates/default/etc/hosts
```

   The boot image will need to be rebuilt in step 8.

8. Rebuild the boot image and reboot. Make sure you rerun the `shell_bootimage_`*label*`.sh` script to re-clone the boot image and to pick up a change to the templates. For information about creating a boot image, see Procedure 3 on page 66.

9. Update the `CLEinstall.conf` file with the service node changes before the next CLE software update so that it matches the new configuration of the machine. If you do not update the `CLEinstall.conf` file at this time, the next time `CLEinstall` is run it will fail due to the inconsistency.

   a. Update the `node_class`[*X*] variables. These variables should be kept current with the system configuration.

   b. If using RSIP, update `rsip_nodes` and `rsip_interfaces` variables if the new service node will be an RSIP server.

   c. If any of the new service nodes will be the primary or backup boot node, primary login node, UFS node, or syslog node, then update those variables. These variables are only used during an installation, not an update, but the `CLEinstall.conf` file should be kept current with the system configuration.

10. Update `ssh` keys for new service nodes. If service nodes were added or removed from the configuration, `ssh` keys for them must be cached on the boot node.

    After running `CLEinstall` and booting the boot node, SDB node, and all other service nodes, run the `/var/opt/cray/install/shell_ssh.sh` script so that proper `ssh` host keys get cached in

boot:/root/.ssh/known_hosts. These keys are used for ssh commands to the nodes, 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.

```
smw:~ # shell_ssh.sh
```

6.28 Changing the Location to Log syslog-ng Information

CLE uses the Linux syslog-ng daemon and associated syslog-ng.conf configuration file to log system messages. For more information see the syslog-ng(8) and syslog-ng.conf(5) man pages. You can modify the /etc/syslog-ng/syslog-ng-conf file to change where the log information is saved.

Procedure 52. Configuring syslog-ng system message logs

Follow these steps to modify the default syslog-ng configuration.

1. Log on to the boot node and edit the syslog-ng.conf configuration file. For more information on this file and its configuration options, see the syslog-ng.conf(5) man page.

   ```
   smw:~# ssh root@boot
   boot:~ # vi /etc/syslog-ng/syslog-ng.conf
   ```

2. Restart the syslog-ng daemon on the boot node.

   ```
   boot:~ # /etc/init.d/syslog restart
   ```

3. Edit the configuration file on the syslog node and make the desired changes.

   ```
   boot:~ # xtopview -n 5
   node/5:/ # vi /etc/syslog-ng/syslog-ng.conf
   node/5:/ # exit
   ```

4. Restart the syslog-ng daemon on the syslog node.

   ```
   boot:~ # ssh syslog /etc/init.d/syslog restart
   ```

5. Edit the configuration file on other service nodes by using xtopview in the default view and make the desired changes.

   ```
   boot:~ # xtopview
   default/:/ # vi /etc/syslog-ng/syslog-ng.conf
   default/:/ # exit
   ```

6. Restart the syslog-ng daemon on the remaining service nodes. For each service node, type the following command.

   ```
   boot:~ # ssh nodename /etc/init.d/syslog restart
   ```

Managing Services [7]

This chapter describes how to manage Cray system services to best use the system or to modify a service.

For a list of administrator accounts that enable you to access these functions, see Administering Accounts on page 113.

7.1 Configuring the SMW to Synchronize to a Site NTP Server

The components of the Cray system synchronize time with the System Management Workstation (SMW) via 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. Use the following procedure to configure the SMW to synchronize to a site NTP server.

Procedure 53. Configuring 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.

7.2 Synchronizing 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.

Note: 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, you 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.

7.3 Adding and Starting a Service Using Standard Linux Mechanisms

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.

7.4 Adding and Starting 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.

7.4.1 Adding a Service to List of Services Available under RCA

Before a service can be attributed to a node or nodes, it must first be made available in the SDB database.

Procedure 54. Adding a service to list of services available under RCA

1. Modify the `service_cmd` table of the Service database (SDB) to include new service information (see Changing Services on page 195).
2. Send a `SIGHUP` signal to the failover manager to reread the database.

7.4.2 Indicating 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. You must be user `root` to make a change using the `xtservconfig` command.

Example 89. Adding the `PBS-MOM` service for a specific node

To add the `PBS-MOM` service for node 5, type:

```
boot:~ # xtservconfig -n 5 add PBS-MOM
```

After you configure a new service, reboot the node or send a `SIGHUP` signal to the service (in this example, `PBS-MOM`) on the affected node.

Example 90. Force the `fomd` to update its configuration information about a new or updated service on a node

Log on to the affected node as user `root` and type:

```
# killall -HUP fomd
```

The `killall -HUP fomd` command causes the failover manager to read the database.

Example 91. Effect a change for a new or updated service on a group of nodes

To effect a change for login nodes 1 through 9, type:

```
boot:~ # pdsh -w login[1-9] "killall -HUP fomd"
```

7.5 Creating a Snapshot of `/var`

The `/var` directory on a Cray system can be configured either as persistent (see *Installing and Configuring Cray Linux Environment (CLE) Software*) or nonpersistent. 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 re-created from the central skeleton file, so 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 (for example, 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 you are configuring the shared-root file system. The `xtpkgvar` command is used by the `CLEinstall` utility.

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

7.6 Setting Soft and Hard Limits to Prevent Login Node Hangs

A login node can be caused to hang or become nearly unresponsive by having all available processes on the node in use. 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.

Procedure 55. Preventing login node hangs by setting soft and hard limits

1. On the boot node type the following in order to make changes to the shared root, where `login` is the class name for login nodes.

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

2. Next, add the following lines to the `/etc/security/limits.conf` file, where *soft_lim_num* and *hard_lim_num* are the number of processes at which you would like the hard and soft limits 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's comments for further options).

   ```
   class/login:/ # vi /etc/security/limits.conf
   * soft nproc soft_lim_num
   * hard nproc hard_lim_num
   ```

   Save the file.

3. Verify that the following line is included in the appropriate PAM configuration files for any authentication methods for which you want limits enforced; the PAM configuration files are located in the `/etc/pam.d/` directory. For example, to enforce limits for users connecting via `ssh`, add the `pam_limits.so` line to the file `/etc/pam.d/sshd`. Other applicable authentication methods to include also 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. Type `exit` to return to the normal prompt on the boot node; the changes you made should be effective immediately on login nodes.

   ```
   class/login:/ # exit
   boot:~ #
   ```

5. To test that the limits are in place, from a login node type the following command, which should return the number specified as the soft limit for the number of processes available to a user, for example:

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

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

7.7 Rack-mount SMW: Creating a Cray System Management Workstation (SMW) Bootable Backup Drive

The following procedure creates a bootable backup drive for a rack-mount SMW in order 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.

Procedure 56. Rack-mount SMW: Creating an SMW bootable backup drive

Important: The disk device names shown in this procedure are only examples. You should substitute the actual disk device names for your system. The boot disk is `phy7` and is slot 0, and the bootable backup disk is `phy6` and is slot 1.

Caution: Shut down the Cray system before you begin 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:~ # ls -al /dev/disk/by-path
total 0
drwxr-xr-x 2 root root 380 Mar 15 13:21 .
drwxr-xr-x 6 root root 120 Mar 11 18:42 ..
lrwxrwxrwx 1 root root   9 Mar 11 18:42 pci-0000:00:11.0-scsi-0:0:0:0 -> ../../sr0
lrwxrwxrwx 1 root root   9 Mar 11 18:42 pci-0000:00:12.2-usb-0:3:1.0-scsi-0:0:0:0 -> ../../sdf
lrwxrwxrwx 1 root root   9 Mar 11 18:42 pci-0000:00:12.2-usb-0:3:1.1-scsi-0:0:0:0 -> ../../sr1
lrwxrwxrwx 1 root root   9 Mar 11 18:42 pci-0000:00:12.2-usb-0:3:1.1-scsi-0:0:0:1 -> ../../sdg
lrwxrwxrwx 1 root root   9 Mar 11 18:42 pci-0000:00:13.2-usb-0:2.1:1.0-scsi-0:0:0:0 -> ../../sde
lrwxrwxrwx 1 root root  10 Mar 11 18:42 pci-0000:00:13.2-usb-0:2.1:1.0-scsi-0:0:0:0-part1 -> ../../sde1
lrwxrwxrwx 1 root root  10 Mar 11 18:42 pci-0000:00:13.2-usb-0:2.1:1.0-scsi-0:0:0:0-part2 -> ../../sde2
lrwxrwxrwx 1 root root   9 Mar 11 18:42 pci-0000:05:00.0-sas-phy4:1...
lrwxrwxrwx 1 root root  10 Mar 14 15:57 pci-0000:05:00.0-sas-phy4:1-...
lrwxrwxrwx 1 root root   9 Mar 11 18:42 pci-0000:05:00.0-sas-phy5:1-...
lrwxrwxrwx 1 root root  10 Mar 14 16:00 pci-0000:05:00.0-sas-phy5:1-...
lrwxrwxrwx 1 root root   9 Mar 11 18:42 pci-0000:05:00.0-sas-phy6:1-...
lrwxrwxrwx 1 root root  10 Mar 15 13:21 pci-0000:05:00.0-sas-phy6:1-...
lrwxrwxrwx 1 root root  10 Mar 15 13:21 pci-0000:05:00.0-sas-phy6:1-...
lrwxrwxrwx 1 root root   9 Mar 11 18:42 pci-0000:05:00.0-sas-phy7:1-...
lrwxrwxrwx 1 root root  10 Mar 11 18:42 pci-0000:05:00.0-sas-phy7:1-...
lrwxrwxrwx 1 root root  10 Mar 11 18:42 pci-0000:05:00.0-sas-phy7:1-...
```

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.

   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. (Example output spacing was modified to fit on the printed page.)

```
mw:~ # fdisk -lu /dev/disk/by-path/pci-0000:05:00.0-sas-phy7:1-0x4433221107000000:0-lun0
Disk /dev/disk/by-path/pci-0000:05:00.0-sas-phy7:1-0x4433221107000000:0-lun0: 160.0 GB, \
160041885696 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-phy7:1-0x4433221107000000:0-lun0-part1 \
                                    Boot   Start       End     Blocks  Id  System
                                             63  16771859   8385898+  82  Linux swap / Solaris
/dev/disk/by-path/pci-0000:05:00.0-sas-phy7:1-0x4433221107000000:0-lun0-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 your changes to the disk. For help, type **`m`** within `fdisk` (see the following sample output, spacing was modified to fit on the printed page.)

```
smw:~ # fdisk -u /dev/disk/by-path/pci-0000:05:00.0-sas-phy6:1-0x4433221106000000:0-lun0

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-phy6:1-0x4433221106000000:0-lun0: 160.0 GB, \
160041885696 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-phy6:1-0x4433221106000000:0-lun0-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-phy6:1-0x4433221106000000:0-lun0-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 your `phy7` sector information.

```
smw:~ # fdisk -lu /dev/disk/by-path/pci-0000:05:00.0-sas-phy6:1-0x4433221106000000:0-lun0
Disk /dev/disk/by-path/pci-0000:05:00.0-sas-phy6:1-0x4433221106000000:0-lun0: 160.0 GB, \
160041885696 bytes
255 heads, 63 sectors/track, 19457 cylinders, total 312581808 sectors
```

3. Initialize the swap device.

```
smw:~ # mkswap /dev/disk/by-path/pci-0000:05:00.0-sas-phy6:1-0x4433221106000000:0-lun0-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
```

4. Update the `grub` device table to recognize any new drives added since the initial operating system installation.

   a. Back up the current `grub` device mapping file.

```
smw:~ # mv /boot/grub/device.map /boot/grub/device.map-20110315
```

   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)

[ 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> quit
```

The file /boot/grub/device.map is now updated to reflect all drives, utilizing the standardized drive naming. This file can be viewed for verification; for example:

```
smw:~ # cat /boot/grub/device.map
(fd0)   /dev/fd0
(hd0)   /dev/sda
(hd1)   /dev/sdb
(hd2)   /dev/sdc
(hd3)   /dev/sdd
(hd4)   /dev/sde
```

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-phy6:1-0x4433221106000000:0-lun0-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.
```

6. Mount the new backup root file system on /mnt.

```
smw:~ # mount /dev/disk/by-path/pci-0000:05:00.0-sas-phy6:1-0x4433221106000000:0-lun0-part2 /mnt
```

7. 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 /.

8. 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-phy7:1-0x4433221107000000:0-lun0-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. Install the `grub` boot loader.

   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 `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.

      Note: The Linux `grub` utility and boot system **always** refer to drives as `hd`, regardless of the actual type of drives.

For example:

```
smw:~ # grub
    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: With SLES 11 SP1, `grub` recreates `device.map` with the short names, not the persistent names. The `/dev/sd`*x* names must not be trusted. Always use `find` the next time you execute `grub` because it is very likely that `grub root` will **not** be `hd2` the next time you execute `grub`.

10. Unmount the backup root partition.

    ```
    smw:~ # umount /mnt
    ```

    The drive is now bootable once plugged in or cabled as the primary drive.

7.8 Desk-side SMW: Creating an System Management Workstation (SMW) Bootable Backup Drive

The following procedure creates a System Management Workstation (SMW) bootable backup drive for a desk-side SMW in order 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.

Note: In the following procedure, `/dev/sd`*X*`2` is the SMW disk (either `/dev/sdb2` or `/dev/sdc2`).

Procedure 57. Desk-side SMW: Creating an SMW bootable backup drive

Important: The disk device names shown in this procedure are only examples. You should substitute the actual disk device names for your system. 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`.

Caution: Shut down the Cray system before you begin 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 your 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. Update the grub device table to recognize any new drives added since the initial operating system installation.

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, utilizing the standardized drive naming. This file can be viewed for verification; for example:

```
smw:~ # cat /boot/grub/device.map
(fd0)   /dev/fd0
(hd0)   /dev/sda
(hd1)   /dev/sdc
```

5. 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:~ #
```

6. Mount the new backup root file system on `/mnt`.

```
smw:~ # mount /dev/sdb2 /mnt
```

7. 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 `/`.

8. 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
```

9. Install the `grub` boot loader.

   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 `grub` boot facility.

```
smw:~ # cd /
smw:~ # touch /mnt/THIS_IS_SDX
```

   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.

      Note: The Linux `grub` utility and boot system **always** refer to drives as `hd`, regardless of the actual type of drives.

For example:

```
smw:~ # grub
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
```

10. Unmount the backup root partition.

    ```
    smw:~ # umount /dev/sdb2
    ```

    The drive is now bootable once plugged in or cabled as the primary drive.

7.9 Rack-mount SMW: Setting Up the Bootable Backup Drive as an Alternate Boot Device

The following procedure modifies a bootable backup drive for a rack-mount SMW in order to boot from and run the rack-mount SMW from the backup root partition.

Important: 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.

Procedure 58. Rack-mount SMW: Setting up the bootable backup drive as an alternate boot device

Note: This procedure will **not** provide a usable backup drive that can be booted in the event of a primary drive failure.

Caution: The disk device names shown in this procedure are only examples. You should substitute the actual disk device names for your 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-phy6:1-0x4433221106000000:0-lun0-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:

   - Modify the title to uniquely identify the backup boot entry.
   - Modify the `root (hd0,1)` directive to reflect the `grub` name of the backup drive.
   - Modify the `root=` and `resume=` specifications to reference the backup drive device.

   An example `/boot/grub/menu.lst` file follows. 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 Thu Mar 17 12:30:29 EDT 2011
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 SP1 - 2.6.32.23-0.3
    root (hd0,1)
    kernel /boot/vmlinuz-2.6.32.23-0.3-default \
root=/dev/disk/by-path/pci-0000:05:00.0-sas-phy7:1-0x4433221107000000:0-lun0-part2 \
resume=/dev/disk/by-path/pci-0000:05:00.0-sas-phy7:1-0x4433221107000000:0-lun0-part1 \
splash=silent crashkernel=256M-:128M@16M showopts vga=0x31a
    initrd /boot/initrd-2.6.32.23-0.3-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 SP1 - 2.6.32.23-0.3
    root (hd2,1)
    kernel /boot/vmlinuz-2.6.32.23-0.3-default \
root=/dev/disk/by-path/pci-0000:05:00.0-sas-phy6:1-0x4433221106000000:0-lun0-part2 \
resume=/dev/disk/by-path/pci-0000:05:00.0-sas-phy6:1-0x4433221106000000:0-lun0-part1 \
splash=silent crashkernel=256M-:128M@16M showopts vga=0x31a
initrd /boot/initrd-2.6.32.23-0.3-default

###Don't change this comment - YaST2 identifier: Original name: failsafe###
title Failsafe -- SUSE Linux Enterprise Server 11 SP1 - 2.6.32.23-0.3
    root (hd0,1)
    kernel /boot/vmlinuz-2.6.32.23-0.3-default \
root=/dev/disk/by-path/pci-0000:05:00.0-sas-phy7:1-0x4433221107000000:0-lun0-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-2.6.32.23-0.3-default
```

3. Modify the backup drive's `/etc/fstab` file to reference the secondary drive slot rather than the first drive slot.

a. Examine the backup drive's `fstab` file.

```
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-phy7:1-0x4433221107000000:0-lun0-part1 \
swap    swap   defaults        0 0
/dev/disk/by-path/pci-0000:05:00.0-sas-phy7:1-0x4433221107000000:0-lun0-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
```

b. 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 `phy6:1-`....

```
smw:~ # vi /mnt/etc/fstab
smw:~ # cat /mnt/etc/fstab
/dev/disk/by-path/pci-0000:05:00.0-sas-phy6:1-0x4433221106000000:0-lun0-part1 \
swap    swap   defaults        0 0
/dev/disk/by-path/pci-0000:05:00.0-sas-phy6:1-0x4433221106000000:0-lun0-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.

7.10 Desk-side SMW: Setting Up the Bootable Backup Drive as an Alternate Boot Device

The following procedure modifies a bootable backup drive for a desk-side SMW in order to boot from and run the desk-side SMW from the backup root partition.

Important: 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.

Procedure 59. Desk-side SMW: Setting up the bootable backup drive as an alternate boot device

Note: This procedure will **not** provide a usable backup drive that can be booted in the event of a primary drive failure.

Caution: The disk device names shown in this procedure are provided as examples only. Substitute the correct disk devices for your system. 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`.

1. Mount the backup drive's root partition.

   ```
   smw:~ # mount /dev/sdX2  /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:

   - Modify the title to uniquely identify the backup boot entry.
   - Modify the `root (hd0,1)` directive to reflect the `grub` name of the backup drive. If you do not know the `grub` name of the backup drive, it is provided in the `/boot/grub/device.map` file on the primary drive.
   - Modify the `root=` and `resume=` specifications to reference the backup drive device.

An example `/boot/grub/menu.lst` file follows. Note the new entry at the end of the file. This example references `/dev/sda` as the primary drive and `/dev/sdc` as the backup drive.

```
smw:~ # cat /boot/grub/menu.lst
# Modified by YaST2. Last modification on Wed Dec 9 15:09:52 UTC 2009
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 - 2.6.27.19-5
    root (hd0,1)
    kernel /boot/vmlinuz-2.6.27.19-5-default root=/dev/sda2 \
    resume=/dev/sda1 splash=silent crashkernel=256M-:128M@16M showopts vga=0x31a \
    initrd /boot/initrd-2.6.27.19-5-default

###Don't change this comment - YaST2 identifier: Original name: failsafe###
title Failsafe -- SUSE Linux Enterprise Server 11 - 2.6.27.19-5
    root (hd0,1)
    kernel /boot/vmlinuz-2.6.27.19-5-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 \
    initrd /boot/initrd-2.6.27.19-5-default

###Don't change this comment - YaST2 identifier: Original name: floppy###
title Floppy
    rootnoverify (fd0)
    chainloader +1

### 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 - 2.6.27.19-5
    root (hd0,1)
    kernel /boot/vmlinuz-2.6.27.19-5-default root=/dev/sdc2 \
    resume=/dev/sdc1 splash=silent crashkernel=256M-:128M@16M showopts vga=0x31a \
    initrd /boot/initrd-2.6.27.19-5-default
```

3. Modify the backup drive's `/etc/fstab` file to reference the secondary drive slot rather than the first drive slot.

   a. Examine the backup drive's `fstab` file.

```
smw:~ # cat /mnt/etc/fstab
/dev/sda1   swap                swap       defaults             0 0
/dev/sda2   /                   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
```

b. Edit the `/mnt/etc/fstab` file, changing `/dev/sda1` and `/dev/sda2` to reference the backup drive. In the following example, the backup drive is `/dev/sdc`.

```
smw:~ # vi /mnt/etc/fstab
/dev/sdc1   swap                swap       defaults              0 0
/dev/sdc2   /                   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 /dev/sdX2
   ```

   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.

7.11 Archiving the SDB

The service database (SDB) can be archived by using the `mysqldump` command. For more information, see http://www.dev.mysql.com/doc/.

7.12 Backing Up Limited Shared-root Configuration Data

Cray recommends that if you cannot make a full copy, make a backup copy of the `.shared` root structure before making significant changes to the shared root. You can use the `xtoparchive` utility or the Linux utilities (`cp`, `tar`, `cpio`) to save the shared-root file system. Run these procedures from the boot node.

7.12.1 Using the `xtoparchive` Utility to Archive the Shared-root File System

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.

Example 92. Using the `xtoparchive` utility to archive the shared-root file system

Use the following `xtoparchive` command to add files specified by the specifications listed in `specfile` to the archive file `archive.042208`; create the archive file if it does not already exist:

```
% xtoparchive -a -f specfile archive.042208
```

Note: To archive any specialized files that have changed, invoke the `archive_etc.sh` script. You can do this while your 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. 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.

7.12.2 Using Linux Utilities to Save the Shared-root File System

Use the Linux utilities (`cp`, `tar`, `cpio`) to save the shared-root file system.

Procedure 60. Backing up limited shared-root configuration data

Cray recommends that if you cannot make a full copy, make a backup copy of the `.shared` root structure before making significant changes to the shared root. Run this procedure from the boot node.

1. Change to the shared root directory that you are backing up.

   ```
   boot:/rr # cd /rr/current
   ```

2. Create a `tar` file for the directory.

   ```
   boot:/rr/current # tar czf /rr/dot_shared-20050929.tgz .shared
   ```

3. Change to the `/rr` directory.

   ```
   boot:/rr/current # cd /rr
   ```

4. Verify that the file exists.

   ```
   boot:/rr # ls -al dot_shared-20050929.tgz
   -rw-r--r--    1 root     root      7049675 Sep 29 14:21
   dot_shared-20050929.tgz
   boot:/rr #
   ```

For more information, see the `cp`(1), `tar`(1), and `cpio`(1) man pages.

7.13 Backing Up Boot Root and Shared Root

Before you back up your boot root and shared root, consider the following issues.

- You must be root to do this procedure.
- 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 your site.
- If the backup file systems have not been used yet, you may need to run `mkfs` first.
- File systems should be quiescent and clean (`fsck`) to get an optimal dump and restore.

You can back up the boot root and the shared root by using the `xthotbackup` command or by using the Linux `dump` and `restore` commands.

7.13.1 Using 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.

Note: By default, `xthotbackup` forces file system checks by using `fsck -f`, unless you use the `xthotbackup -n` option. All `fsck` activity is done in parallel (by default, `xthotbackup` uses the `fsck -p` option to check all file systems in parallel), unless you use the `xthotbackup -l` option.

Load the `cray-install-tools` module to access the `xthotbackup` utility and the `xthotbackup`(8) man page.

Example 93. Using the `xthotbackup` command to create a bootable backup system set

Enter the following to dump all of the partitions from the source label, `BLUE`, to the backup label, `GREEN`, and then make them bootable.

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.

```
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.

Example 94. Using the `xthotbackup` command to copy selected file systems from source to the backup system set

The following example dumps only the `SDB` and `SYSLOG` partitions in the system set labelled `BLUE` to the system set labelled `GREEN`.

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.

```
smw:~ # xthotbackup -f SDB,SYSLOG BLUE GREEN
```

7.13.2 Using dump and `restore` Commands to Back Up Boot Root and Shared Root

Procedure 61. Backing up the boot root and shared root using the dump and `restore` commands

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
    ```

7.14 Backing Up User Data

Backing up user data is a site-specific activity. You can use Linux utilities to back up user files and directories.

7.15 Rebooting a Stopped SMW

If the SMW is down or being rebooted (that is, not fully working), the L0 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 L0 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 uncorrupted configuration files on a local SMW disk.

Procedure 62. Rebooting a stopped SMW

1. Verify that your configuration files contain the most recent system configuration.
2. Boot the SMW.

7.15.1 SMW Recovery

Procedure 63. SMW primary disk failure recovery

The following procedure describes how to recover an SMW primary disk failure. To find out how to create a System Management Workstation (SMW) bootable backup drive, see Procedure 57 on page 228. To find out how to modify a bootable backup drive, in order to boot from and run the SMW from the backup root partition, see Procedure 58 on page 233.

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.

To recover an SMW, you must reorder the drives at the front of the SMW. No BIOS or software configuration changes are required.

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.

7.16 Recovering from Service Database Failure

If you notice problems with the SDB, for example, if commands like `xtprocadmin` do not work, restart the service-node daemons.

Example 95. Recovering from an SDB failure

Type the following command on the SDB node:

```
sdb:~ # /etc/init.d/sdb restart
```

Commands in this file stop and restart MySQL.

7.16.1 Database Server Failover

The SDB uses dual-ported local RAID to store files.

If you have SDB node failover 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.

7.16.2 Rebuilding Corrupted SDB Tables

The boot process creates all SDB tables except the accounting and boot tables. If you notice a small corruption and you do not want to reboot, you can change the content of a database table manually by using the tools in Table 8. If you cannot recover a database table in any other way, as a last resort reboot the system.

7.17 Using Persistent SCSI Device Names

Important: The information provided in this section 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, you **must** use persistent device names for file systems on your Cray system.

Cray recommends that you use 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 *Installing and Configuring Cray Linux Environment (CLE) Software*.

Alternatively, you 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: You 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.

7.17.1 Using `cray-scscidev-emulation` Device Naming

Cray provides a utility (`cray-scscidev-emulation`) that emulates the basic functionality of the obsolete `scsidev` method for SCSI device naming.

The device alias is created in `/dev/scsi` for any devices that match entries in the alias file, `/etc/scsi.alias`. Format the alias file as follows, where *SN* is the serial number of the hard disk, *PN* is the partition number of the disk, and *devname* is the desired alias name.

`serial_number="`*SN*`", devtype=disk, [partition=`*PN*`,] alias=`*devname*

For example, the following entry creates a device entry `/dev/scsi/cab2-3-c1-shroot` for partition `2` on the disk with the serial number `030B9ED30300`.

```
serial_number="030B9ED30300", devtype=disk, partition=2, alias=cab2-3-c1-shroot
```

Note: The following limitations apply when using `cray-scsidev-emulation`:

- This capability is not implemented in the `initramfs`; it cannot be used to specify the boot root.
- Only the format shown is supported; `scsidev` supported a number of additional formats.
- Only one alias per disk is supported.
- Only symbolic links are supported.

7.18 Using a Linux `iptables` Firewall to Limit Services

You can set up a firewall to limit services that are running on your 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` and firewall scripts, see the `iptables`(8) man page, http://www.iptables-tutorial.frozentux.net/iptables-tutorial.html, and http://www.linuxguruz.com/iptables/.

7.19 Handling Single-node Failures

A single-node failure is visible when you use the `xtnodestat` command.

You can parse the `syslog` to look for failures.

You can use the Cray Management Services (CMS) log manager to collect, analyze, and display messages from the system. For additional information, see *Using Cray Management Services (CMS)*.

If you suspect problems with a node, 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 your service representative.

To see cabinet status, use the System Environmental Data Collections (SEDC); see *Using and Configuring System Environment Data Collections (SEDC)*.

For more information, see the `xtnodestat`(1), `xtcli`(8), and `xtsedcviewer`(8) man pages.

7.20 Increasing the Boot Manager Time-out 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 default time-out value, as shown in Example 96.

Example 96. Increasing 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
```

7.21 RAID Failure

System RAID has its own recovery system that the manufacturer supplies. For more information, refer to the manufacturer documentation.

Using the Application Level Placement Scheduler (ALPS) [8]

ALPS (Application Level Placement Scheduler) is the Cray supported mechanism for placing and launching applications on CNL 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 also supports interactive application placement and launch.

Note: 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.

8.1 ALPS Functionality

ALPS performs the following functions:

- Assigns application IDs.
- Manages compute node resources.
- Provides a configurable node selection algorithm for placing applications. (See the `ALPS_NIDORDER` configuration parameter in `/etc/sysconfig/alps` Configuration File on page 255 for more information.)
- Launches applications.
- Delivers signals to applications.
- Returns `Supports` 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 ALPS and Node Health Monitoring Interaction on page 266). For additional information about the CLE Node Health software, see Configuring Node Health Checker (NHC) on page 160.
- If running Cray Checkpoint Restart (Cray CPR), assists in application checkpoint and restart; for information about using Cray CPR, see Chapter 10, Using Checkpoint/Restart on Cray Systems (Deferred implementation) on page 289.

8.2 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 (for detailed descriptions, see ALPS Clients on page 249 and the man page for each ALPS client):

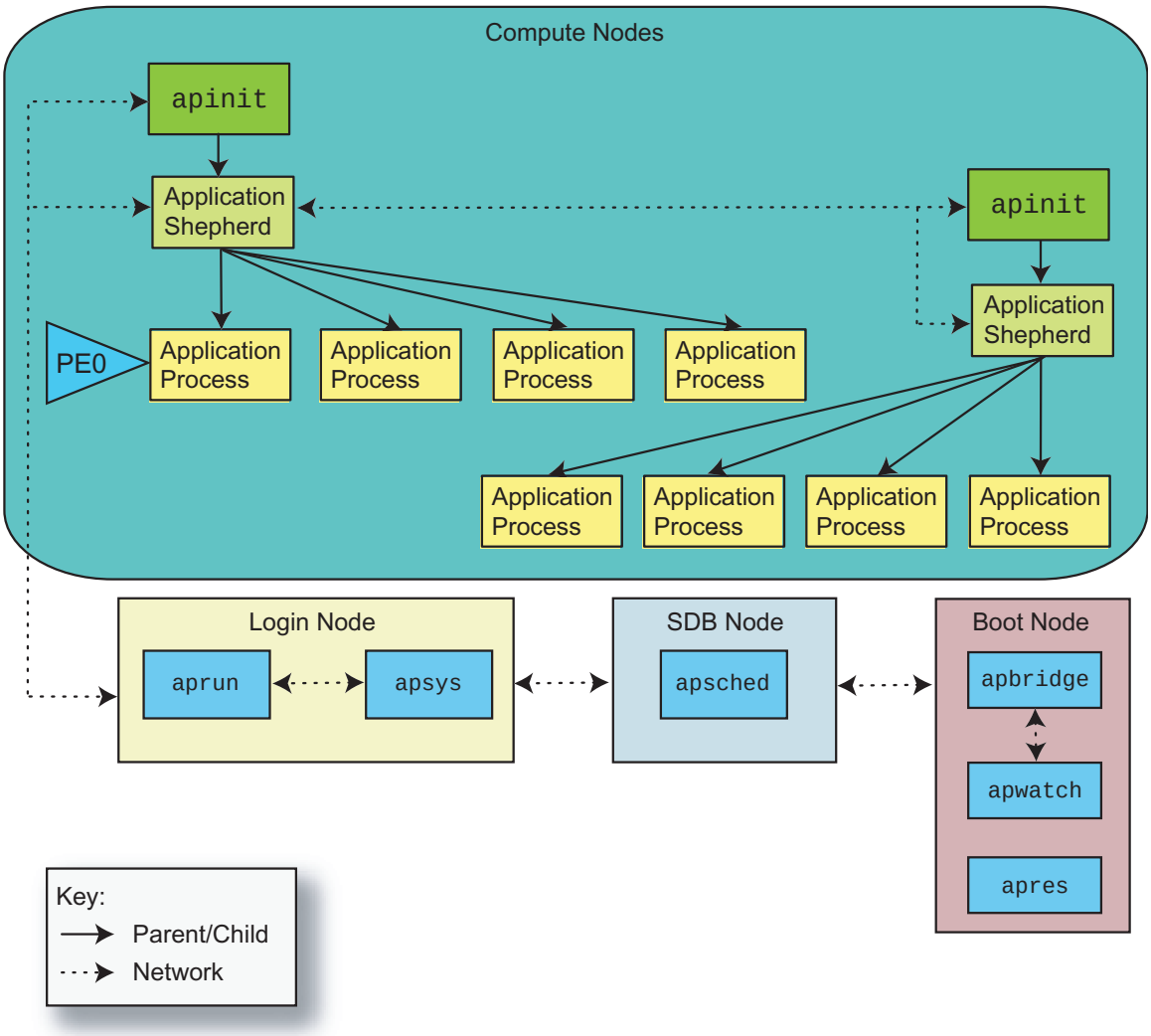
- `aprun`: Application submission
- `apstat`: Application placement and reservation status
- `apkill`: Application signaling
- `apmgr`: Collection of functions usually used by the system administrator in exceptional circumstances to manage ALPS
- `apbasil`: Workload manager interface

ALPS daemons (for detailed descriptions, see ALPS Daemons on page 252 and the man page for each ALPS daemon):

- `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

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, illustrates the ALPS process.

Figure 4. ALPS Process



8.2.1 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.

8.2.1.1 The aprun Client

The aprun client is used for application submission. Specifically, a user executes the aprun command to run a compiled 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.

Note: 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.

8.2.1.2 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.

8.2.1.3 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.

8.2.1.4 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 other than the boot node.

For information about using the apmgr subcommands, see the apmgr(8) man page.

8.2.1.5 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.

8.2.2 ALPS Daemons

ALPS daemons provide support for application submission, placement, execution, and cleanup on the system.

8.2.2.1 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.

8.2.2.2 The apsched Daemon

The apsched daemon manages memory and processor resources of applications running on compute nodes.

Note: 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.

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

8.2.2.3 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 `/etc/sysconfig/alps` Configuration File on page 255.

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

8.2.2.4 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.

8.2.2.5 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/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.

8.2.2.6 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.

8.2.2.7 ALPS Log Files

Each of the ALPS daemons writes information to its log file in /var/log/alps 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 /var/log/alps 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 apbridge*mmdd*, for example, apbridge1027.

8.2.2.8 Changing Debug Message Level of apsched and apsys Daemons

The level of debug messages written by the apsched and apsys daemons is defined in the /etc/alps.conf configuration file. You can change the debug level dynamically by modifying the alps.conf file and sending a SIGHUP signal to apsched or apsys, as applicable, to read the alps.conf file.

8.3 Configuring ALPS

ALPS uses the following three files:

- `/etc/sysconfig/alps` configuration file
- `/etc/alps.conf` configuration file
- `/etc/init.d/alps` file, which is used to start and stop ALPS components and does not require customization

Note: 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* and the *Installing and Configuring Cray Linux Environment (CLE) Software*.

8.3.1 `/etc/sysconfig/alps` Configuration File

The `/etc/sysconfig/alps` file is in both the boot root and in the shared root. If you defined the ALPS-related parameters in your `CLEinstall.conf` file, after installation the parameters and settings are placed into your `/etc/sysconfig/alps` file.

If you do not define the ALPS-related parameters in your `CLEinstall.conf` file, to use ALPS you must define the parameters in your `/etc/sysconfig/alps` file (required parameters are indicated) and then start the ALPS daemons.

Note: 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 (`uname -n`) of the service node that runs `apsched`. Cray recommends that the SDB node be used as the `ALPS_MASTER_NODE`. For example:
`ALPS_MASTER_NODE="nid00005"`

`ALPS_BRIDGE_NODE`

(Required) Specifies the node name (`uname -n`) of the service node that runs `apbridge`. This is usually the boot node. Network connectivity between the SMW and the `ALPS_BRIDGE_NODE` parameter is required. (Such connectivity is guaranteed to exist from the boot node.) This default value is enforced in the `/etc/init.d/alps` file. For example:
`ALPS_BRIDGE_NODE="boot"`

ALPS_MOUNT_SHARED_FS

Specifies if a separate file system is to be mounted at ALPS startup to hold control data; default is no. 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: ALPS_MOUNT_SHARED_FS="no"

ALPS_SHARED_DEV_NAME

Specifies the device to mount at ALPS start-up. If it is null and ALPS_MOUNT_SHARED_FS is yes, the device is determined by /etc/fstab. This parameter is not used unless yes is specified for ALPS_MOUNT_SHARED_FS. For example: ALPS_SHARED_DEV_NAME="ufs:/ufs/alps_shared"

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="192.168"

ALPS_NIDORDER

Specifies the sorting order to use for assigning nodes for an application's use. For example: ALPS_NIDORDER="-Ox"

-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.

Note: Cray recommends this option for Cray XE systems. Use of this option results in better application performance for larger applications run on Cray XE systems.

-On Assigns order of nodes based solely on ascending numerical Node ID (NID) order. This is the default sorting order.

`-Ox` Assigns order of nodes by maximum dimension as the outer dimension; the smallest dimension will change most quickly. For example, a system whose topology is described as a 6x12x8 system would have the y dimension varied last and the x dimension varied most rapidly, ordering 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.

`-Oy` Specifies to order by y-axis **last** (for Cray XE systems of more than three cabinets.)

`-Or` Assigns order of nodes by minimum dimension, a reverse of `-Ox`; in the example above, the y dimension would vary most quickly, the x least.

If `ALPS_NIDORDER` is not specified, the default action is `-On`.

Note: Because this is a system-wide setting, Cray recommends that you change this option only when you reboot your system to ensure `apbridge` and `apsched` are restarted in the correct sequence.

`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/librsmsevent.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"`

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`.

The following example shows a sample `/etc/sysconfig/alps` configuration file.

Example 97. Sample `/etc/sysconfig/alps` configuration file

```
#ALPS Configuration File

ALPS_MASTER_NODE="sdb"

ALPS_BRIDGE_NODE="boot"

ALPS_NIDORDER="-Ox"

ALPS_MOUNT_SHARED_FS="no"

## 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=""

APWATCH_LIBRARY_PATH=
"/opt/gnet/2.0.5/64/lib:/opt/glib/2.4.2/64/lib:/opt/cray/lib64:/opt/gnome/lib64"

APWATCH_ERD="smw"
```

8.3.2 `/etc/alps.conf` Configuration File

The `/etc/alps.conf` file is in the shared root and contains ALPS static configuration information used by the `apsched` and `apsys` daemons. The configuration parameters are described in this section.

Note: You can change the parameter settings dynamically by modifying the `alps.conf` file and sending a `SIGHUP` signal to `apsched` or `apsys`, as applicable, to re-read the `alps.conf` file.

`bridge` Enables the `apbridge` daemon to provide dynamic rather than static information about the system node configuration to `apsched`. Cray strongly recommends setting the `bridge` parameter to use the `apbridge` daemon. By default, it is set to `1` (enabled).

`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. By default, it is set to `0`.

`fanout` This field is set to a default level of `32`. This value controls the width of the ALPS TCP/IP network fan-out tree used by `apinit` on the compute nodes for ALPS application launch, transfer, and control messages.

`debug` This field is set to a default level of `1` for both `apsched` and `apsys`. For information about valid values, see the `apsched`(8) and `apsys`(8) man pages.

`cpuAffinity`

Supports switchable default CPU affinity in `apsched`. Valid values are `cpu`, `none`, and `numa`; the default value is `cpu`. `aprun` checks for and uses the default `cpuAaffinity` string from `apsched`. If the user has not explicitly set the `aprun -cc` option, `aprun` will use the default supplied by `apsched`. If there is not a default from `apsched`, `aprun` sets a default of `cpu`. For more information, see the `aprun`(1) man page.

`lustreFlush`

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.

Valid values are `0` (disabled) and `1` (enabled); the default value is `1` (enabled). `Apsched` provides this default `lustreFlush` value to the `apinit` daemon to enable or disable Lustre cache flushing as part of application exit processing.

Note: This value cannot be set on an individual application basis; it is a system-wide setting.

`nodeShare` Controls which compute node cores and memory are put into an application cpuset on the compute node. The valid values are `exclusive` and `share`. The default value is `exclusive`.

The `exclusive` setting puts all of a compute node's cores and memory resources into an application-specific cpuset 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 `aprun -cc` option.

The `share` setting restricts the application specific cpuset 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, then only NUMA node 0 cores and memory will be contained within the application cpuset. The application 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/alps.conf` on an individual basis, use the `aprun -F` option. For more information, see the `aprun`(1) man page.

`cleanup_version`

Specifies which cleanup routines are used. Valid values are `1` or `2`. Value `1` indicates using existing cleanup (scalar, limited parallel actions); `2` indicates cleanup that is scaled for larger systems (highly parallel.) Default for systems installed before SMW 5.1 is `1`; default for systems installed with SMW 5.1 or later is `2`.

`cleanup_cto`

Specifies the maximum amount of time, in milliseconds, allowed for the connect system call to respond before assuming the target node is down. Default value is `1000` milliseconds.

Note: The `cleanup_cto` value applies only when you have also specified `cleanup_version=2`.

`pTagFreeDelay`

Specifies the number of seconds to delay allocation of a PTag used previously. (The current PTag allocation implementation rotates through the PTags range.) Default value is `30` seconds.

`cms` (Deferred implementation) Indicates whether or not ALPS will use CMS to store reservation and claim information. Valid settings are `no` and `yes`; the default setting is `no`.

`pTagGlobalNodes`

The minimum application size in nodes that forces ALPS to assign a global pTag value.

`cmsTimeout` Maximum time in milliseconds that calls to CMS have before timing out.

`cmsLogTime` Logs the calls that take longer than the time specified for `cmsTimeout`.

`sharedDir`

Note: `ALPS_SHARED_DIR_PATH` was removed from `/etc/sysconfig/alps`.

(Required) Specifies the directory path to the file that contains ALPS control data. If `ALPS_MOUNT_SHARED_FS` is set to `yes`, this is assumed to be a mount point. Default is `/ufs/alps_shared`. For example: `sharedDir /ufs/alps_shared`

The following example shows a sample `/etc/alps.conf` configuration file.

Example 98. Sample `/etc/alps.conf` configuration file

```
# ALPS configuration file

apsched
        alloc           0
        bridge          1
        fanout          32
        debug           1
# Default CPU affinity: values cpu (default), none, numa
#       cpuAffinity     cpu
# Default lustre cache flushing at app exit: values 0, 1
#       lustreFlush     1
# Default app node share mode for cores and memory: values exclusive, share
#       nodeShare       exclusive
#
# Value used for systems with Gemini/Aries network:
# - pTagGlobalNodes: if set to X, apps >= than X nodes
#   will use a global PTag (and NOT use the NTT)
#       pTagGlobalNodes 5000
#
# CMS variables:
# - cms: if yes (no default), apsched will send reservations/claims to CMS
# - cmsTimeout: if set to X, calls to CMS will timeout in X msec
# - cmsLogTime: if set to X, log calls to CMS taking longer than X msec
#               (X < 0: no logging, X = 0: log all calls)
#
        cms             yes
        cmsTimeout      5000
        cmsLogTime      1000
#
# Version of ALPS cleanup code: 1 or 2
#
        cleanup_version 2
/apsched

apsys
        debug           1
        cleanup_version 2
/apsys
```

8.4 Resynchronizing 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.

8.5 Identifying 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. You can include multiple application IDs, but it must be a space-separated list of IDs. 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   dnsp3+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]: dnsp3+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. You can include multiple reservation IDs, but it must be a space-separated list of IDs. 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
```

8.6 Terminating a Batch Job

To terminate a batch job, use the job ID from the `apstat -r` display.

8.7 Setting a Compute Node to Batch or Interactive Mode

To set a node to be either batch or interactive mode, use the `xtprocadmin` 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.

Example 99. Retrieving node allocation status

The `apstat -n` command displays the application placement status of the nodes that are UP and their allocation mode (`B` for batch or `I` for interactive) in the State column.

Note: 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
```

8.8 Manually Starting and Stopping ALPS Daemons on Service Nodes

ALPS is automatically loaded and started when CNL is booted on compute nodes.

You can manually start and stop the ALPS daemons on the service nodes as shown in the following procedures.

Procedure 64. Starting and stopping ALPS daemons on a specific service node

1. To start the ALPS daemons on a specific service node, log on to that service node as `root` and type the `/etc/init.d/alps start` command; for example, to start the ALPS daemons on the boot node:

   ```
   boot:~ # /etc/init.d/alps start
   ```

2. To stop the ALPS daemons on a specific service node, log on to that service node as `root` and type the `/etc/init.d/alps stop` command; for example, to stop the ALPS daemons on the boot node:

   ```
   boot:~ # /etc/init.d/alps stop
   ```

Procedure 65. Restarting ALPS daemon on a specific service node

- To restart the ALPS daemon on a specific service node, log on to the service node as `root` and type the `/etc/init.d/alps restart` command; for example, to restart the ALPS daemons on the boot node:

  ```
  boot:~ # /etc/init.d/alps restart
  ```

  The `/etc/init.d/alps restart` command stops and then starts the ALPS daemons on the node.

8.9 Manually Cleaning ALPS and PBS After Downed Login Node

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 PBS.

Procedure 66. Manually cleaning up ALPS and PBS after a login node goes down

1. Verify that the batch job still appears in the `qstat` output.

```
crayadm@smw:~> qstat -as 106728.sdb

sdb:
                                                            Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  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)
```

2. Purge job from PBS and verify that it was purged. On the SDB node, type:

   ```
   sdb:~ # qdel -W force 106728.sdb
   ```

3. Verify that the job no longer exists.

   ```
   sdb:~ # qstat -as 106728.sdb
   qstat: Unknown Job Id 106728.sdb
   sdb:~ #
   ```

4. Restart `apsched` on the SDB node:

   ```
   sdb:~# /etc/init.d/alps restart
   ```

5. 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
```

6. Use `apstat` to verify that the reservation no longer exists.

```
sdb:~ # apstat -r | grep 106728
sdb:~ #
```

8.10 Verifying 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.

Example 100. Verifying that ALPS is communicating with Cray system compute nodes

```
crayadm@login:~> cd /tmp
crayadm@login:/tmp> aprun -b -n 16 -N 1 /bin/cat /proc/sys/kernel/hostname
```

8.11 ALPS and Node Health Monitoring Interaction

ALPS and node health monitoring cooperate in performing application cleanup following an application exit. The Node Health Checker (NHC) is automatically invoked by ALPS upon the termination of an application.

During normal operations, applications are run on a set of nodes, complete successfully, then those node resources are freed up to be reallocated for other applications. When an application exit is considered *orderly*, a set of up to four unique application process exit codes and exit signals is gathered and consolidated by ALPS on each compute node within the application placement list. Once all of the application processes on a compute node have exited, that compute node adds its local exit information to this consolidated list of exit data.

The exit information is sent to `aprun` over the ALPS application specific TCP fan-out tree control network. All of the application processes must have completely exited before this exit information is received by `aprun`. `aprun` forwards the compiled exit 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. An orderly exit means that exit information about the application was received by `aprun` and forwarded to `apsys`. `apsys` sends an exit message to `apsched`, which releases the reserved resources for another application.

An *unorderly* exit means that exit information has not been received by `apsys` prior to an `aprun` exit. A typical occurrence of an unorderly exit consists of a `SIGKILL` signal being sent to `aprun` by the batch system after the application's wall time limit is exceeded.

Since there is no exit information available to `apsys` during 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 free those application resources for another application.

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 application resources for reallocation until all of the application processes have exited or node health monitoring has marked applicable compute nodes `admindown` or `suspect`. Until that time, the application will continue to be shown in `apstat` displays.

8.11.1 aprun Actions

The `aprun` command is the ALPS application launch command on login nodes and the SDB node. `aprun` has a persistent TCP connection to a local `apsys`. `aprun` also has 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. `aprun` 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 as described in following sections.

An unorderly exit may occur for various reasons. The usual causes of an unorderly exit include the following cases:

- The batch system sends a SIGKILL signal to aprun due to the application wall time expiring
- apkill or kill are used to send a SIGKILL signal to aprun
- aprun 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
- aprun receives a fatal read, write or unexpected close error on the TCP socket it uses to communicate with apinit

8.11.2 apinit Actions

apinit 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.

8.11.3 apsys Actions

`apsys` 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 unorderly 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 version 1 (`apmgrcleanup`) or cleanup version 2, and the `apsys` agent blocks until cleanup completes. (Which cleanup version is controlled by the `cleanup_version` configuration parameter in `alps.conf` file; see `/etc/alps.conf` Configuration File on page 259 for more information.)

At the start of application cleanup, the `/var/log/alps/apsys`*MMDD* log file displays data similar to the following messages.

Cleanup version 1 messages:

```
14:00:20: [32606] Agent unexpected close of peer connection 6, apid 227061
14:00:20: [32606] Agent invoking cleanup v1 for apid 227061
14:00:22: /usr/bin/apmgrcleanup [32824] invoking
/opt/cray/nodehealth/default/bin/xtcleanup_after /tmp/apsysiY08fc 227061 0 with 1 entries
```

Cleanup version 2 messages:

```
14:00:20: [32606] Agent unexpected close of peer connection 6, apid 227061
14:00:20: [32606] Agent invoking cleanup v2 for apid 227061
14:00:20: Beginning cleanup of apid 227061, iteration 1
14:00:21: Post-cleanup: apid 227061 definitely resident on 1/1 nodes, maybe on 0
others
14:00:21: Beginning cleanup of apid 227061, iteration 2
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):
14:00:21: Post-cleanup: apid 227061 definitely resident on 0/1 nodes, maybe on 0
others
14:00:21: Invoking health check: /opt/cray/nodehealth/default/bin/xtcleanup_after
/tmp/apsysWRPpna 227061 0
14:00:30: Successfully cleaned up apid 227061 on 1 nodes
```

After `apmgrcleanup` returns, the `apsys` log file contains something similar to the sample message below:

```
14:02:30: [32606] Agent sending ALPSMSG_EXIT message to apsched fd 7, apid 227061
14:02:30: [32606] Agent received ALPSMSG_EXITCONFIRM from apsched fd 7, apid 227061
```

In the above example, `apsched` has been told that the resources assigned to that `aprun` can now be reallocated to another application. The `apstat` display will no longer show information about this application.

8.11.4 Cleanup Version 1 Actions (`apmgrcleanup`)

Note: Which cleanup version used is controlled by the `cleanup_version` configuration parameter in `alps.conf` file; see `/etc/alps.conf` Configuration File on page 259 for more information.

When configured to use cleanup version 1, `apsys` invokes `apmgrcleanup` for each application unorderly exit. `apmgrcleanup` is a shell script that is invoked to do application cleanup for a specific application. As part of this cleanup activity, `apmgrcleanup` calls another script, which may invoke node health monitoring. `apmgrcleanup` executes with the permissions of the `apsys` caller, which runs as `root`. You must be `root` to edit the `apmgrcleanup` file.

`apmgrcleanup` works with a placement list of assigned compute nodes for a specific application. This application cleanup activity will guarantee that a new application is not placed on this set of compute nodes prematurely. A new application placed on these compute nodes prematurely would result in application failure due to compute node core and/or memory resources still being assigned to the current application.

`apmgrcleanup` contacts every node in the placement list supplied to it. `apmgrcleanup` first uses `apmgr` to send a kill request message for a specific application to each node on the placement list, then requests status information about an application on that compute node.

apmgrcleanup uses apmgr to send status request messages to the apinit on that set of compute nodes to find out when all of the local application processes have exited. The kernel may not immediately deliver a SIGKILL signal to application processes if those processes are involved in I/O activity.

apmgrcleanup begins by calling apmgr to send a ping kill message to the apinit daemon on each compute node in the placement list for the given application. If there are more than 500 nodes in the list, apmgrcleanup uses nway to perform eight apmgr invocations at a time, in a sliding window fashion, for parallelization. apmgrcleanup continues to loop until the list of nodes reaches zero.

apmgr writes messages to the syslog after each successfully sent ping kill message. These messages mean only that a message was received by the compute node apinit daemon. The application processes may still exist if the SIGKILL delivery to an application process remains pending due to I/O activity. Below is a sample of ping kill messages written to the syslog:

```
Apr 13 06:55:31 nid00016 apmgr[20277]: apid=821502, killed on nid=587
Apr 13 06:55:31 nid00016 apmgr[20279]: apid=821502, killed on nid=591
Apr 13 06:55:31 nid00016 apmgr[20281]: apid=821502, killed on nid=772
Apr 13 06:55:31 nid00016 apmgr[20283]: apid=821502, killed on nid=776
```

Inside its main loop, apmgrcleanup calls the xtcleanup_after script with the initial (full) placement list of compute nodes for the application. Each invocation includes a randomly generated file name (/tmp/apsys*XXXX*) that holds the node list and an invocation count.

```
Apr 13 06:55:31 nid00016 06:55:31: /usr/bin/apmgrcleanup [18964] invoking
/opt/xt-service/default/bin/snos64/xtcleanup_after /tmp/apsysdbajiE 821502
0 with 5 entries
```

Then invocation count tells xtcleanup_after if this is the first or subsequent call of the script. The xtcleanup_after script typically calls node health monitoring. The script is site configurable to modify its behavior as desired; however, modifying this script is not recommended.

On return from xtcleanup_after, apmgrcleanup will wait one or more seconds, depending on machine size, to avoid looping too quickly, then it rechecks the list of nodes. First, apmgrcleanup invokes apstat and checks for compute nodes that are not marked up, removing them from the /tmp/apsys*XXXX* file. Then, it calls apmgr to send a ping status request to the apinit daemon on the remaining compute nodes.

A compute node is removed from the /tmp/apsys*XXXX* file whenever the apinit on that compute node responds to the ping status request stating that no application processes remain on that compute node, or when the node is no longer marked up. The ping status request has a five-second time limit. Any nodes remaining, (that is, not heard from, still marked up) will stay in the file of nodes for the next iteration of the apmgrcleanup loop.

When the /tmp/apsys*XXXX* file is empty, apmgrcleanup will exit. Then, apsys writes a message into the syslog and can tell apsched to release the aprun claim for that set of compute nodes. The syslog message includes both the batch job ID and the aprun exit code, making it easier to track.

```
May 19 08:26:52 nid00029 apsys[27933]: apid=200075, Finishing, user=10320,
exit_code=0, exitcode_array=0, exitsignal_array=0

May 19 08:34:48 nid00029 apsys[2376]: apid=200175, Finishing, user=10320,
exit_code=139, exitcode_array=130:0, exitsignal_array=11:9:0
```

After the initial apmgr ping, kill messages are sent to the apinit daemon on the set of compute nodes within the /tmp/apsys*XXXX* file, apmgrcleanup calls the xtcleanup_after script to invoke node health monitoring. If node health monitoring is enabled, compute nodes may be marked admindown or suspect by node health monitoring as described in Node Health Checker Actions on page 273.

8.11.5 Cleanup Version 2 Actions

Note: Which cleanup version used is controlled by the cleanup_version configuration parameter in alps.conf file; see /etc/alps.conf Configuration File on page 259 for more information.

When configured to use cleanup version 2, apsys uses an internal library to perform cleanup for each application unorderly exit. The apmgrcleanup and apmgr commands, used in cleanup version 1 are not used. Cleanup version 2 interacts with the /tmp/apsys/*XXXX* file and the xtcleanup_after script in the same way as in cleanup version 1, and makes the same guarantee that new applications will not be placed on compute nodes prematurely (See Cleanup Version 1 Actions (apmgrcleanup) on page 270 for details). The principle differences between Cleanup version 1 and Cleanup version 2 are the signal delivery and application query mechanism, and the scalability characteristics.

Cleanup version 2 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):
```

After cleanup version 2 completes, or after every iteration of cleanup starting with the third iteration, the `xtcleanup_after` script is invoked in an identical fashion to cleanup version 1.

8.11.6 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 specified 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. If not, it 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 Configuring Node Health Checker (NHC) on page 160.

8.11.7 Verifying Application 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/log/alps/apsys`*MMDD* log files for that *apid*; verify cleanup version 1 (`apmgrcleanup`) or cleanup version 2 has been invoked.
- If using cleanup version 1 on that same login node, use `ps` to check if `apmgrcleanup` is still executing.
- Check the applicable node health monitoring log file (`/var/log/xtcheckhealth_log`) for that *apid*.
- Check the SMW `/var/opt/cray/log/bootlogs/console.`*YYMMDDHHMM* log file for that *apid*.

Using Comprehensive System Accounting [9]

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

Specific third-party software releases are required for batch system compatibility with CSA on Cray systems. For more information, access the **3rd Party Batch SW** link on the CrayPort website at http://www.crayport.cray.com.

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 OST nodes do not support CSA.

9.1 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.

9.2 CSA Configuration File Values

The CSA configuration file, `csa.conf`, is included with the Cray Linux Environment (CLE) software release package. By default, on Cray systems, this file is located at `/etc/opt/cray/csa/csa.conf` for login and compute nodes.

This file contains default settings for several configuration parameters you must change to tailor CSA to your individual site configuration.

Note: The `csa.conf` file exists on the shared root and also on the CNL image for compute nodes; the two copies of this file **must** be identical (except for the `NODE_PROCESS_ACCOUNT` parameter, which may be different). You must create a new version of the CNL compute node image after editing the `csa.conf` file.

Each Cray system that runs CNL has its own unique hardware configuration. This includes the number of nodes on the system and the physical location of the nodes. In addition, each installation contains its own unique file system configuration.

Since the file system and node configurations for each Cray system is unique, the default `/etc/opt/cray/csa/csa.conf` file can only be used as a template. 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 9. CSA Parameters That Must Be Specific to Your System

Parameter	Description
ACCT_SIO_NODES	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 df command to display the possible file system mount points. The actual maximum number of ACCT_SIO_NODES that may be specified is limited by the number of file systems available on your system.
ACCT_FILE_SYSTEM_00 ... ACCT_FILE_SYSTEM_*nn*	Must be one entry for each declared file system mount point. Numbering must begin with 00, and numbers must be consecutive. For example, if you have specified ACCT_SIO_NODES 1, you will only define ACCT_FILE_SYSTEM_00. If you have specified ACCT_SIO_NODES 2, you will also need to define ACCT_FILE_SYSTEM_01.
_lus_nid00023_csa_XT	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 ACCT_FILE_SYSTEM declared. **Note:** The program that parses the configuration file does not allow any special characters, other than the underscore character (_) in configuration names. Therefore, in the file system paths used in the mount point description, each forward slash character (/) character must be represented by an underscore (_) character. This also means that an account file system mount point cannot have a _ character in the pathname.
SYSTEM_CSA_PATH	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.
NODE_PROCESS_ACCOUNT	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.

9.3 Configuring CSA

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.

Note: You must include the CSA RPM in your CNL boot image. To do this either set `CNL_csa=`**`yes`** in the `CLEinstall.conf` before the `CLEinstall` program is run or edit the `shell_bootimage_`*label*`.sh` script and specify `CNL_CSA=`**`y`** prior to updating your CNL boot image. If you do 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.

9.3.1 Obtaining File System and Node Information

Procedure 67. Obtaining file system and node information

1. 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/nid00023
30@gni:/ib54ex       114821632416   5588944 108983420416   1% /lus/ib54ex
```

2. 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.

3. Determine the hardware node configuration on your system.

Run the `xtprocadmin` command to get a complete list of nodes.

```
login:~ > xtprocadmin

   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.

9.3.2 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. By default, on Cray systems, this file is located at `/etc/opt/cray/csa/csa.conf` for login and 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.

Procedure 68. Editing CSA parameters for the example system

1. Set the number for the `ACCT_SIO_NODES` parameter.

   From Procedure 67 on page 278, you know that both `/lus/nid00011` and `/lus/nid000064` will be used to host individual node accounting files. The number of file systems (in this case two) to be used to contain accounting files is the value for the `ACCT_SIO_NODES` parameter. Since this example shows using `/lus/nid00011` and `/lus/nid00064` to contain accounting files, set `ACCT_SIO_NODES` to `2`:

   ```
   ACCT_SIO_NODES  2
   ```

2. Declare a file system mount point for each SIO node specified.

   Note: The program that parses the configuration file does not allow any special characters, other than the underscore character (_) in configuration names. Therefore, in the file system paths used in the mount point description, each forward slash character (/) character must be represented by an underscore (_) character. This also means that an account file system mount point cannot have a _ character in the pathname.

   The `df` command from the previous procedure showed a mount point on `/lus/nid00011` and another one on `/lus/nid00064`, these are the two mount points that need to be declared. Just because there are multiple mount points, however, does not mean that you need to use them. You may choose to have all accounting files written to a single file system. Since in this example you are configuring two mount points, you must specify `ACCT_FILE_SYSTEM_00` and `ACCT_FILE_SYSTEM_01` parameters:

   ```
   ACCT_FILE_SYSTEM_00  _lus_nid00011
   ACCT_FILE_SYSTEM_01  _lus_nid00064
   ```

3. Determine the node range values for the account system mount point parameters.

   All accounting file directories have `csa` as the first element of the path name, following the mount point. The next element in the path name after `csa` describes the node type. For Cray node types, the next element of the path name is `XT`.

   For Cray systems, the CSA software uses the node name, otherwise known as the *cname*, when creating pathnames for accounting files. For example, node name `c1-0c2s7n3` has a pathname of `cab1/row0/chassis2/slot7/mcomp3`. This path name is appended to applicable accounting system mount point name in order to create a full path name for the accounting file.

   The `xtprocadmin` command output from the previous procedure shows that the system has 4 cabinets, `c0-c3`. One simple way to configure the accounting file systems so that the files are divided fairly evenly between the two file systems in this example would be to specify that cabinet `0` and cabinet `1` have their data written to `/lus/nid00011`, and cabinet `2` and cabinet `3` have their data written to `/lus/nid00064`.

   Using the pathname conventions described above, and the node name data from Procedure 67 on page 278, you can define the file system mount point parameters:

   ```
   _lus_nid00011_csa_XT   c0-0c0s0n0--c1-0c2s7n3
   _lus_nid00064_csa_XT   c2-0c0s0n0--c3-0c2s7n3
   ```

4. Define the SYSTEM_CSA_PATH parameter.

   The SYSTEM_CSA_PATH parameter describes the file pathname for the system wide csa directories that are used for CSA work areas, and for containing the system-wide pacct file. The system-wide pacct file contains the merged contents of the individual node pacct files. Since the file pathname for the SYSTEM_CSA_PATH is not used as an input to the configuration file parser, the file path name is allowed to contain the / character.

   Usually the SYSTEM_CSA_PATH parameter uses an account file system mount point as its base directory, however, this is not required. The SYSTEM_CSA_PATH parameter is only used on the login node where CSA file processing is performed. It is not necessary to set this parameter in the compute node image of /etc/opt/cray/csa/csa.conf, but setting it there does not cause any problems.

   For this example, use the /lus/nid00011 mount point for the CSA work areas:

   ```
   SYSTEM_CSA_PATH    /lus/nid00011/csa
   ```

5. Define the NODE_PROCESS_ACCOUNT parameter.

   The NODE_PROCESS_ACCOUNT parameter defines if all process accounting records from nodes are to be collected. This parameter may be set differently for /etc/opt/cray/csa/csa.conf in the compute node image than in /etc/opt/cray/csa/csa.conf in the shared root file system. The NODE_PROCESS_ACCOUNT parameter allows your site to determine how much detailed accounting data is to be collected, processed, and saved from the nodes on the system.

   To understand the usefulness of this parameter, it may be helpful to know how CSA accounting records are handled on Cray systems. When ALPS launches an application to the compute nodes on a Cray system, CSA process accounting occurs on each compute node. All CSA job and process accounting records for each compute node are written to an in-memory file system on the node, and the records remain there until the application terminates. When the application terminates, ALPS notifies the CSA software on each compute node to process the accounting data for that node. The NODE_PROCESS_ACCOUNT parameter allows CSA to make a decision whether to write all of the individual process accounting records for each compute node to the common file system, or to read the individual process accounting records and combine them into a single application summary record that represents the total resources used by the application on the compute node. By choosing to have application summary records, the amount of data transferred from each compute node to the common file system may be substantially reduced. In doing so, the amount of internal system network traffic and the amount of data moved from compute nodes to

disk can be decreased. Also, the total amount of CSA accounting data that must be processed later for creating usage reports, and the amount of CSA data to be permanently stored can be reduced.

You may want to set NODE_PROCESS_ACCOUNT off for compute nodes, and to set it on for service nodes. This provides more accounting process detail on the login nodes where such information may be more useful. Therefore, this single parameter may be set differently on the shared root image than it is set on the compute node image of /etc/opt/cray/csa/csa.conf.

To use this split configuration, specify the following:

```
# Shared root version of /etc/opt/cray/csa/csa.conf:
NODE_PROCESS_ACCOUNT    ON

# Compute node image of /etc/opt/cray/csa/csa.conf:
NODE_PROCESS_ACCOUNT    OFF
```

6. Change the parameter that defines the group name used for setting the ownership and group on accounting files. This parameter is named CHGRP and defaults to:

   ```
   CHGRP          csaacct
   ```

   If you use a different group name, change the parameter to match your system configuration.

9.3.3 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 Managing System Configuration With the xtopview Tool on page 129 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 to reflect the changes you have made:

  ```
  /usr/sbin/pwconv
  ```

- 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 Setting Up Job Accounting on page 285.

9.3.4 Creating a CNL Image with CSA Enabled

After you have modified the compute node copy of `csa.conf`, you must rebuild the compute node image. For more information about how to rebuild the compute node image, see Preparing a Service Node and Compute Node Boot Image on page 64.

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 Managing System Configuration With the `xtopview` Tool on page 129 or the `xtopview`(8) man page.

9.3.5 Setting Up 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.

Procedure 69. Setting up CSA project accounting

The project database resides on the system SDB node as a MySQL database. To set up a CSA project accounting for your system, perform the following 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. Create and edit the `/etc/opt/cray/projdb/projects` file 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. Create and edit the `/etc/opt/cray/projdb/useracct` file 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. 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. Change the permissions and owner on the `~crayadm/.my.cnf` file, as follows:

   ```
   chmod 600 ~crayadm/.my.cnf
   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, edit the `/etc/opt/cray/projdb/projdb.conf` project accounting configuration file so that it contains site-specific values for the parameters listed in Table 10.

Table 10. 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 `MYSQL` (default) to `CUSTOM`.
`CUSTOM_VALIDATE`	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 `/usr/local/sbin/validate_account`. Input parameters to the validation function are position order dependent, as follows: `user_name  account_number`

8. 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 via the job PAM module are assigned a default account ID at the time of system access.

```
login:/home/crayadm:~> projdb -c -p /etc/opt/cray/projdb/projects -u
/etc/opt/cray/projdb/useracct -v
```

Note: The project database package commands are installed in `/opt/cray/projdb/default/bin`, which must be in your `PATH` variable to access the commands.

9.3.5.1 Disabling Project Accounting

If you do not want to use project accounting on your site, either as provided by the MySQL database, or by a separate customer-supplied LDAP server, use the following procedure to disable project accounting.

Procedure 70. Disabling project accounting

1. In the `/etc/opt/cray/projdb/projdb.conf` file, set the `PROJDBTYPE` parameter to `CUSTOM`.
2. In the `/etc/opt/cray/projdb/projdb.conf` file, declare a `CUSTOM_VALIDATE` parameter and define it as `/usr/local/sbin/validate_account`.
3. As `root`, create the `/usr/local/sbin/validate_account` file with file permissions set to `755` and the following contents:

```
#!/bin/sh
echo 0
```

9.3.6 Setting Up Job Accounting

Note: You must include the `csa` RPM in your CNL boot image. To do this either set `CNL_csa=`**`yes`** in the `CLEinstall.conf` before the `CLEinstall` program is run or edit the `shell_bootimage_`*label*`.sh` script and specify `CNL_CSA=`**`y`** prior to updating your CNL boot image. If you do 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.

Procedure 71. Setting up CSA job accounting

- Use the `xtopview` command to edit the `/etc/pam.d/common-session` file on the shared root image to make sure that jobs are created whenever a login occurs via `ssh`. Add the following entry to the `/etc/pam.d/common-session` file:

```
session  optional   /opt/cray/job/default/lib64/security/pam_job.so
```

After you make sure this is working for all `sshd` session logins, you may want to change the entry to:

```
session   required    /opt/cray/job/default/lib64/security/pam_job.so
```

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 Managing System Configuration With the `xtopview` Tool on page 129 or the `xtopview`(8) man page.

9.4 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).

9.4.1 `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`.

Example 101. 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
```

9.4.2 `csarun` `cron` Job

You normally execute the `csarun` script at defined intervals to generate a set of system accounting reports.

Example 102. 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.

9.4.3 `csaperiod` `cron` Job

You can invoke the `csaperiod` script to run periodic accounting at different intervals than the regular system accounting interval.

Example 103. Running periodic accounting at different intervals than the regular system accounting interval

To run `csaperiod` every four hours at 5 minutes before the hour, use a `crontab` entry of the following form:

```
55 3,7,11,15,19,23 * * * /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.

9.5 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.

9.6 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.

Using Checkpoint/Restart on Cray Systems (Deferred implementation) [10]

Checkpoint/Restart (CPR) provides a way to suspend and snapshot the state of a running application. This snapshot can then be used to restart the application at a later time for use in application recovery (after a failure) or coarse grained scheduling. This is useful in case of failure or in case you need to suspend a long-running application for some other reason.

This chapter provides Cray CPR details. For complete information about BLCR, see the Berkeley Lab Checkpoint/Restart documentation available on the website at https://ftg.lbl.gov/projects/CheckpointRestart/.

Note: In the Berkeley Lab Checkpoint/Restart documentation, pay special attention to the caveats: in particular, that files open for writing are truncated to the file position at the time of checkpoint. Because each process accessing a shared file will most likely have a different file position, the file will be truncated to the smallest file position at the time of checkpoint.

10.1 Requirements and/or Limitations for Checkpoint/Restart

10.1.1 Using Current Cray MPT Libraries

The Cray systems CPR feature is built upon the Berkeley Lab Checkpoint/Restart (BLCR) for Linux. CPR jobs on Cray systems also require a library which has integrated BLCR support; for that reason, applications must be linked with the currently supported Cray MPT libraries. For performance monitoring of applications that may be checkpointed and restarted, CrayPat (Cray performance analysis tool) 5.0.2 release is also required.

Note: Only applications using the MPI and SHMEM programming models are checkpointable.

10.1.2 Specifying Batch System Software Releases

Specific third-party batch system software releases are required for checkpoint/restart support. For current information, access the **3rd Party Batch SW** link on the CrayPort website at http://www.crayport.cray.com.

10.1.3 Setting File System Access Pattern

Due to the known file-per-node I/O access of checkpoint/restart, the checkpoint directory's file system setting should be optimized for this access pattern. For Lustre, set the checkpoint directory stripe count to one for optimal performance.

```
lfs setstripe checkpoint_dir -s 0 -i -1 -c 1
```

10.1.4 Disabling `mmap` Mechanism

Due to a known BLCR limitation, the checkpoint/restart of a previously checkpointed application fails if the `mmap` mechanism of the Name Service Cache Daemon (NSCD) is enabled.

BLCR recommends disabling the NSCD `mmap` mechanism by add the following lines (or modifying similar ones) in the file `/etc/nscd.conf`:

```
shared passwd no
shared group no
shared hosts no
```

Restart NSCD for the change to take effect.

10.2 Installation and Configuration

Several entities need to be installed and configured before CPR can be used with Cray applications.

10.2.1 Cray Installation and Configuration Options

The Cray Linux Environment (CLE) installation tool handles most of the details of installing the software necessary for checkpoint/restart support.

To enable checkpoint/restart set `cpr=yes` in the `CLEinstall.conf` before the `CLEinstall` program is run. To include the RPM for the CPR client in your CNL boot image, either set `CNL_cpr=yes` in the `CLEinstall.conf` before the `CLEinstall` program is run or edit the `shell_bootimage_`*label*`.sh` script and specify `CNL_cpr=y` prior to updating your CNL boot image. If you do 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. For more specific installation instructions, see *Installing and Configuring Cray Linux Environment (CLE) Software*.

Your batch system must also recognize and interpret CPR directives.

If you have not done so, invoke the following command sequence from the boot node as `root`:

```
boot# xtopview -x /etc/opt/cray/sdb/node_classes -c login
login> chkconfig blcr on
login> exit
```

On the subsequent system boot, this starts up CPR services on all nodes of that class.

10.2.2 Configuring TORQUE and Moab to Work with CPR

Recent TORQUE releases have support for checkpoint/restart on the Cray platform.

Cray support is compiled in by default, so no additional configuration options are necessary. (TORQUE includes configuration option `--enable-blcr`.)

To enable the TORQUE's CPR support, certain variables must be set in the MOM configuration file, `mom_priv/config` in the TORQUE server home directory.

```
$checkpoint_script /opt/cray/cprbatchutils/default/libexec/checkpoint.torque
$restart_script /opt/cray/cprbatchutils/default/libexec/restart.torque
$checkpoint_run_exe /opt/cray/blcr/default/bin/cr_run
```

Files `checkpoint.torque` and `restart.torque` are part of the `cray-cprbatchutils` package; file `cr_run` is a part of the `blcr` package.

> **Note:** In typical Cray systems with TORQUE, TORQUE's server home directory is `/var/spool/torque`, which resides in a persistent `/var` file system. Therefore, you must make the configuration changes for each persistent `/var` directory associated with each node that runs a TORQUE MOM.

In addition, the destination location for checkpoint files must be on a file system accessible from the compute nodes (like Lustre). Because the TORQUE server default checkpoint directory is not on such a file system, you should override this value on a per queue basis using the following command:

`qmgr -c "set queue` ***queuename*** **`checkpoint_dir=`*`checkpoint_dir`*`"`**

Due to the known file-per-node I/O access of checkpoint/restart, the checkpoint directory's file system setting should be optimized for this access pattern. For Lustre, it is optimal to set the stripe count to one.

`lfs setstripe` ***checkpoint_dir*** **`0 -1 1`**

10.2.3 Configuring PBS Professional to Work with CPR

Deferred Implementation: The PBS Professional integration with CPR on Cray systems will be available in a future release.

To enable CPR support in PBS Professional, you must set variables as shown below in the MOM configuration file, `mom_priv/config` in the PBS home directory:

```
$action checkpoint 300
!/opt/cray/cprbatchutils/default/libexec/checkpoint.pbspro %sid %jobid %uid %gid %path 0
$action checkpoint_abort 300
!/opt/cray/cprbatchutils/default/libexec/checkpoint.pbspro %sid %jobid %uid %gid %path 9
$action restart 300
!/opt/cray/cprbatchutils/default/libexec/restart.pbspro %sid %jobid %uid %gid %path
$restart_transmogrify true
$checkpoint_path checkpoint_dir
```

Files `checkpoint.pbspro` and `restart.pbspro` are part of the `cray-cprbatchutils` package; file `cr_run` is a part of the BLCR package.

Note: In typical Cray systems with PBS Professional, the PBS Professional home directory is `/var/spool/PBS`, which resides in a persistent `/var` file system. Therefore, you must make the configuration changes for each persistent `/var` directory associated with each node that runs a PBS Professional MOM.

10.3 Using Checkpoint/Restart

To use CPR, an application must be linked with the currently supported Cray MPT libraries and with BLCR. In addition, the batch system must recognize and interpret CPR directives.

10.3.1 Compiling Applications

Applications must be linked with the currently supported Cray MPT libraries to have the code necessary to support checkpointing. Thus, only applications using the MPI and SHMEM programming models are checkpointable. To enable the MPT checkpoint support, the application must also be linked with BLCR. Loading the BLCR module automatically adds the necessary options to Cray compiler scripts to link this library:

```
module load blcr
```

Because the Cray checkpoint/restart solution uses Berkeley Lab's Checkpoint/Restart (BLCR) software, it inherits its caveats and limitations in addition to the Cray MPT requirement. For more information, refer to the BLCR documentation: http://www.upc-bugs.lbl.gov/blcr/doc/html/index.html.

10.3.2 Using Checkpoint/Restart with TORQUE and Moab

For complete details about checkpointing and restarting with TORQUE and Moab, see the TORQUE documentation at http://www.clusterresources.com/torquedocs21/2.6jobcheckpoint.shtml.

The following examples show typical user tasks.

Example 104. Submit a job to TORQUE

To submit a job and tell TORQUE it is checkpointable:

```
login:~ # qsub -c enabled jobscript
```

Example 105. Submit a job to TORQUE that checkpoints every 30 minutes

To submit a job that checkpoints every 30 minutes:

```
login:~ # qsub -c periodic,interval=30 jobscript
```

Example 106. Checkpoint and terminate a job using TORQUE

To checkpoint and terminate a job that is checkpointable:

```
login:~ # qhold jobid
```

Example 107. Restart a held job using TORQUE

To restart a held job:

```
login:~ # qrls jobid
```

Example 108. Restart a checkpointed job using TORQUE

To restart a checkpointed job in the completed state:

```
login:~ # qrerun jobid
```

10.3.2.1 Common Checkpoint/Restart Error Messages

For TORQUE and Moab batch system checkpoint failures, error messages are reported in the "comment" field of job status command (`qstat -f $JOBID`).

Table 11. BLCR Reported Checkpoint Error Messages

Message	Explanation
`Checkpoint failed:  Checkpoint of application nodes failed`	A problem was encountered checkpointing application nodes. See application `stderr` and Table 2.
`Checkpoint failed:  Checkpoint tool helper launch failed`	The batch system checkpoint directory has permissions that prevent checkpointing. The user application needs write access to this directory, as this is required for application checkpoints. If connectivity between nodes isn't functioning properly, this may also cause this error.

Any of the error messages shown in Table 12 also can be printed by `aprun` to `stderr` of the job/application.

Table 12. Checkpoint/Restart Error Messages

Message	Explanation
`Checkpoint of application 3136 failed:  Support missing from kernel`	The checkpoint failed because BLCR is not installed or loaded on compute nodes.
`Checkpoint of application 3139 failed:  Checkpoint support not linked into one or more processes`	The checkpoint failed because the `libcr` BLCR library was not linked into the user application. Users must specify `module load blcr` before compiling code. Loading the BLCR module automatically adds the necessary options to Cray compiler scripts to link this library.
`Checkpoint of application 463346 failed:  No such file or directory` or `Checkpoint of application 890274 failed:  Permission denied`	The checkpoint failed because a Lustre directory was not specified for the checkpoint data or was not writable by the application user ID.
`Checkpoint of application 1474693 action:  Unsupported programming model`	The checkpoint failed because the target application was not linked with the currently supported MPT libraries.

In addition, a checkpoint request sends a signal to the user application; the following functions (and others) can return early or with `EINTR` due to interruption from signals: `poll`(2), `select`(2), `sleep`(3), `read`(2), and `write`(2). Applications may have unexpected results if the usage of the these functions is not POSIX compliant and does not account for signal interruption.

Table 13. Hardware Error Messages

Message	Explanation
`aprun: Apid 357428: Checkpoint failed: Unknown error 512 _pmii_daemon(SIGCHLD): PE 0 exit signal Killed [NID 00002] 2010-10-19 19:00:41 Apid 357428: Cray HSN detected critical error 0x40c[ptag 0]. Please contact admin for details. Killing pid 9383(`*job_name*`)`	These messages may appear in the console log for applications (perhaps third party) linked with older MPI versions which are checkpointed. Make certain to relink with the correct libraries.

10.3.3 Using Checkpoint/Restart with PBS Professional

Deferred Implementation: The PBS Professional integration with CPR on Cray systems will be available in a future release.

For complete details about using checkpoint/restart with PBS Professional, see the PBS Professional documentation.

The following examples show typical user tasks.

Example 109. Submit a job to PBS Professional

To submit a job and tell PBS Professional it is checkpointable:

```
login:~ # qsub -c s jobscript
```

Example 110. Submit a job to PBS Professional that checkpoints every 3 minutes of CPU time

To submit a job to PBS Professional that checkpoints every 3 minutes of CPU time:

```
login:~ # qsub -c c=3 jobscript
```

Example 111. Checkpoint and terminate a job using PBS Professional

To checkpoint and terminate a job that is checkpointable using PBS Professional:

```
login:~ # qhold jobid
```

Example 112. Restart a held job using PBS Professional

To restart a held job using PBS Professional:

```
login:~ # qrls jobid
```

Example 113. Restart a checkpointed job using PBS Professional

To restart a checkpointed job in the completed state using PBS Professional:

```
login:~ # qrerun jobid
```

Dynamic Shared Objects and Cluster Compatibility Mode in the Cray Linux Environment [11]

11.1 Configuring the Compute Node Root Runtime Environment (CNRTE) Using `CLEinstall`

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, you can also take advantage of this feature. 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 *Installing and Configuring Cray Linux Environment (CLE) Software* and *Workload Management and Application Placement for the Cray Linux Environment*.

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.

To configure the compute node root runtime environment for CLE, do the following:

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 don't 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 you to run the service node image 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 you choose 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.

For more information on repurposing compute nodes for service node roles, see *Repurposing Compute Nodes as Service Nodes on Cray XE and Cray XT Systems*.

2. When editing the `CLEinstall.conf` file and running the `CLEinstall` program , modify the parameters specific to shared object support according to your site-specific configuration.

   When you set the following parameters in the `CLEinstall.conf` file, the `CLEinstall` program will automatically configure your system for the compute node root runtime environment.

   DSL=**`yes`** This variable enables dynamic shared objects and libraries for CLE. The default is `no`.

   Note: Setting this option to `yes` 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.

3. Follow the appropriate procedures in *Installing and Configuring Cray Linux Environment (CLE) Software* 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
```

11.2 Configuring Cluster Compatibility Mode

A Cray XE series system is not a cluster but a massive parallel processing (MPP) computer. An MPP is simply one computer with many networked processors used for distributed computation, and, in the case of Cray XE architectures, a high-speed communications processor 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. CCM supports ISV applications running in four simultaneous cluster jobs on up to 256 CNL compute nodes per job instance. It is built on top of the Compute Node Root Runtime Environment (CNRTE), the infrastructure used to provide dynamic library support in Cray systems.

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. Essentially, CCM uses the batch system to logically designate part of the Cray system as an emulated cluster for the duration of the job as shown in Figure 5 and Figure 6.

Figure 5. Cray System Job Distribution Cross-section

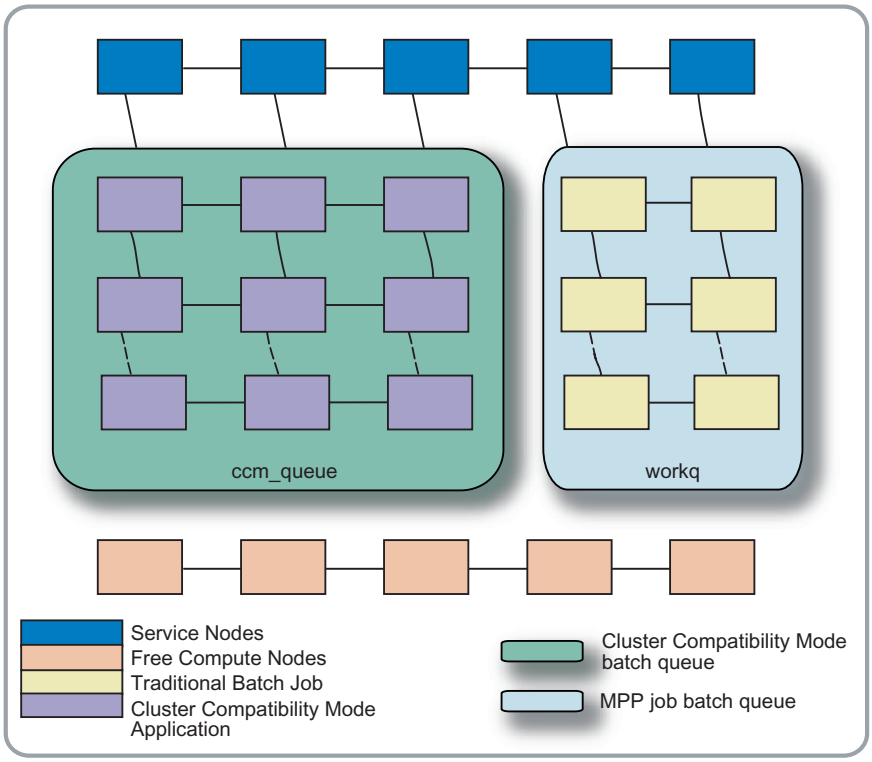
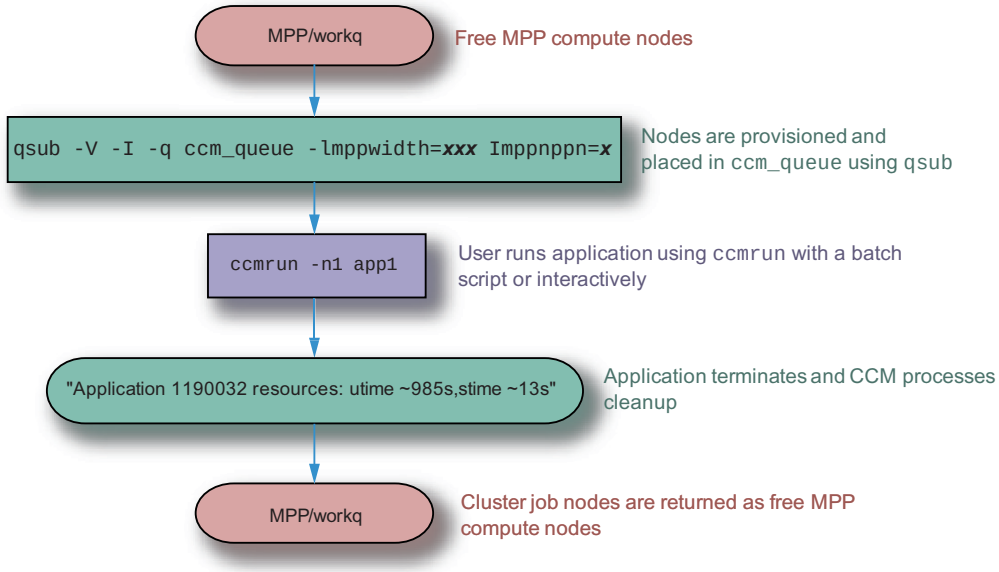


Figure 6. CCM Job Flow Diagram



11.2.1 Preconditions

- Dynamic library support is installed.
- (Optional) RSIP must be installed if you have applications that need access to a license server; see *Installing and Configuring Cray Linux Environment (CLE) Software*.
- PBS 10.2RC2 (Emerald) or Torque-2.4.1b1-snap.200908271407 or later versions are installed.

11.2.2 Configuration Options Relevant to Installation

The following variables are used for installation of CCM in `CLEinstall.conf`. Variables such as `CCM_ENABLERSH` and `CCM_QUEUES` can be changed in `/etc/opt/cray/ccm/ccm.conf` after installation. For more information on how to install CCM, please see *Installing and Configuring Cray Linux Environment (CLE) Software*.

`CCM=`**`yes`** Set this parameter to **`yes`** to enable Cluster Compatibility Mode and install the appropriate RPMs.

`CCM_ENABLERSH=`**`yes`**

Optional: Enables services or daemons that most ISV applications need to run. Examples of these services are `xinetd`, `portmap`, `rsh`, and `rlogin`. If you set `CCM_ENABLERSH` to `no` some ISV applications will not work. If you don't specify this parameter, `rsh` is enabled by default.

`CCM_QUEUES=`***"ccm_queue1,ccm_queue2"***

Specifies one or more batch queues used in the workload management system. The default value is `ccm_queue`.

Important: The syntax in the configuration file, `ccm.conf`, and the installer variable `CCM_QUEUES` differ. In the installer, the queues are listed as comma-separated values. In the configuration file they are space-separated.

After your batch system software is installed, you must manually create the queues you specify here. For steps required to create CCM batch queues, see Procedure 75 on page 306.

`CCM_WLM=`**`pbs`**

Specifies the batch processing system; valid values are `pbs`, `torque`, and `lsf`.

Note: If `CCM_WLM=lsf` is specified and any non-null values are set for `CRAY_QSTAT_PATH` and `CRAY_BATCH_VAR`, a message is displayed by `CLEinstall` stating that the settings of `CRAY_QSTAT_PATH` and `CRAY_BATCH_VAR` will be ignored and the variables will be set to `""` in `/etc/opt/cray/ccm/ccm.conf` on the shared root.

CRAY_QSTAT_PATH=***/opt/pbs/default/bin***

Specifies the path to the batch system software `qstat` command. The default value is the path for PBS Professional; the path for Moab/TORQUE is included as a comment in the configuration file.

This variable is ignored if `CCM_WLM=lsf`.

CRAY_BATCH_VAR=***/var/spool/PBS***

Specifies the path to the batch system software `/var` directory.

CCM_ENABLENIS=***no***

Optional: This option can be set to `yes` to start `ypservices` on the compute node. If NIS is not properly configured, calls will time-out to the network, significantly slowing down CCM startup, so this option is disabled by default.

11.2.3 Post-install Options and Configuration

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_INADDRANYBIND=***yes***

This option tells RSIP to bind `INADDR_ANY` requests to the local network interface rather than using the RSIP address space. Changing this to `no`will cause `INADDR_ANY` bind requests to consume RSIP ports and may prevent application scaling.

To configure `yp`, `/etc/defaultdomain` and `/etc/yp.conf` must be properly configured on the compute node specialized view. Cray recommends that you use the `cnos` class within `xtopview` to set up this specialized view.

Procedure 72. Using DVS to mount home directories on the compute nodes for CCM

For each DVS server node you have configured, follow these steps to mount `/ufs` from the NFS server `ufs`.

1. Create the /ufs mount point on the DVS server by using xtopview in the node view. For example, if your 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:/ # mkdir -p /ufs
```

2. Specialize and add a line to the /etc/fstab file for the node class.

```
node/27:/ # xtspec -n 27 /etc/fstab
node/27:/ # vi /etc/fstab
ufs:/ufs        /ufs    nfs     tcp,rw  0 0
node/27:/ # exit
```

3. Log into each DVS server and mount the file system:

```
boot:~ # ssh nid00027
nid00027:~ # mount /ufs
nid00027:~ # exit
```

4. 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
/ufs /ufs dvs path=/ufs,nodename=c0-0c0s2n3
```

5. 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/ufs
```

6. Update the boot image to include these changes; follow the steps in Procedure 3 on page 66.

   Note: You can defer this step and update the boot image **once** before you finish booting the system.

Procedure 73. Modifying CCM and Platform-MPI system configurations

1. If you wish to enable additional features such as debugging and Linux NIS (Network Information Service) support, 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
```

   If you wish to configure additional CCM debugging, set CCM_DEBUG=*yes*.

   If you wish to enable NIS support, set CCM_ENABLENIS=**yes**.

2. (Optional) You may have a site configuration where the paths for the qstat command is not at a standard location. Change the values in the configuration file for CRAY_QSTAT_PATH and CRAY_BATCH_VAR accordingly for your site configuration.

3. Save and close ccm.conf.

4. If your applications will use Platform-MPI (previously known as *HP-MPI*), Cray recommends you create the `/etc/hpmpi.conf` file with these values.

```
default/:/ # vi /etc/hpmpi.conf
MPI_IC_ORDER="TCP"
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"
```

5. Exit `xtopview`.

```
default/:/ # exit
boot:~ #
```

Procedure 74. Setting 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:

```
boot:~#  xtopview -m "CCM cnos setup" -c cnos
```

   Note: If the SDB has not been started, use the `-x /etc/opt/cray/sdb/node_classes` option to specify node/class relationships.

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 that you want to add or modify for the compute nodes.

3. Exit `xtopview`.

```
class/cnos:/# exit
```

   Note: You are prompted to type **c** and enter a brief comment describing the changes you made. To complete your comment, type **Ctrl-d** or a period on a line by itself. Do this each time you exit `xtopview` to log a record of revisions into a version control system.

Procedure 75. Linking the CCM prologue/epilogue scripts for use with PBS and Moab TORQUE on login nodes

Prerequisites: This procedure requires that you have already installed a workload management system such as PBS or Moab TORQUE.

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.`

Note: This procedure assumes that you are using `/bin/bash` as your 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 [ -f $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
```

Procedure 76. Using `qmgr` to 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 step 1 for additional application-specific queues, if desired.

Procedure 77. Configuring Platform LSF for use with CCM

Prerequisites: This procedure requires that you have already installed the Platform LSF workload management system.

1. Determine the path to the directory on your system 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.

   Example 114. Location of queue configuration files

   If

```
LSF_TOP=/opt/xt-lsfhpc
```

   and

```
LSF_CLUSTER_NAME=nid00196
```

   the full path to the directory containing the queue configuration files would be

```
/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 Procedure 76 on page 307 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_PREEXEC_RETRY=1
DESCRIPTION=ccm_queue
```

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
```

6. 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
```

7. Exit xtopview.

Once you have completed system configuration and started the system compute nodes, you should verify that write permissions are correct. You can accomplish this 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.

OpenFabrics Interconnect Drivers for CLE Systems [12]

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. OFED and InfiniBand RPMs are included in the CLE release and installed by default. However, OFED will not run on your Cray system until you configure the I/O nodes 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 your CLE software, see *Installing and Configuring Cray Linux Environment (CLE) Software*.

12.1 OFED Overview

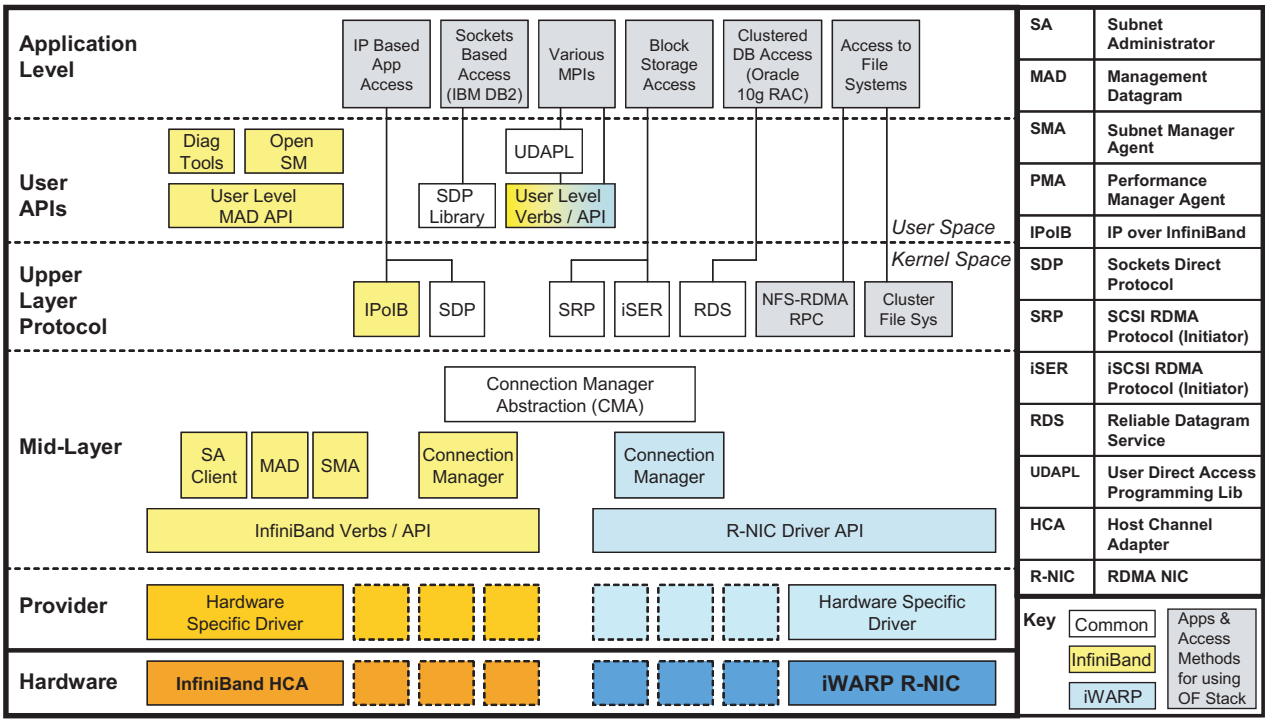
Cray supports InfiniBand as an I/O interconnect. Double data rate (DDR) IB host channel adapters (HCAs) accommodate user data transfers at up to 1.5 GB/s bidirectionally. IB also enables efficient zero-copy, low-latency RDMA transfers between network peers. As a result, IB gives Cray systems the most 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 OSS nodes as a storage interconnect between the Cray system and direct-attach IB storage, or it 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)



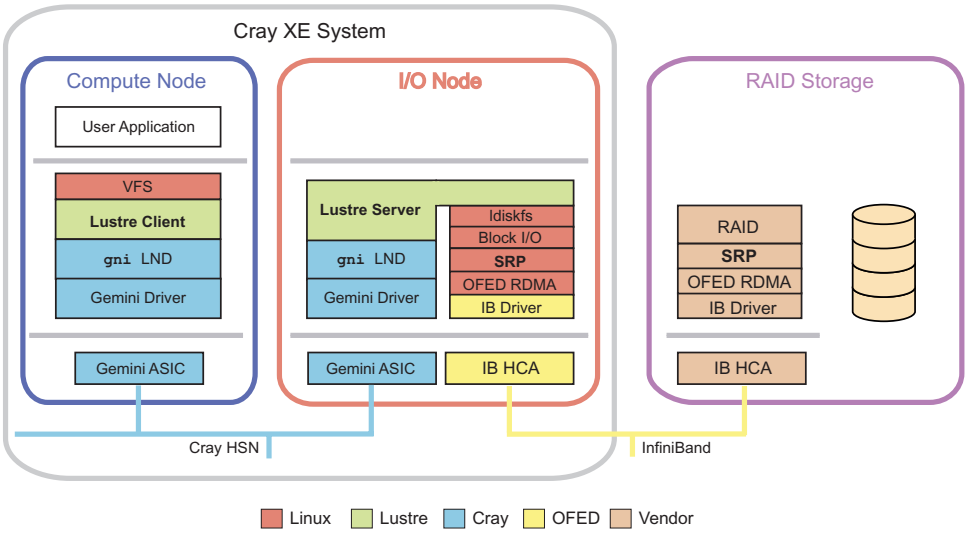
12.2 Using InfiniBand

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.

12.2.1 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. Figure 8 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 your site-specific third party storage vendor.

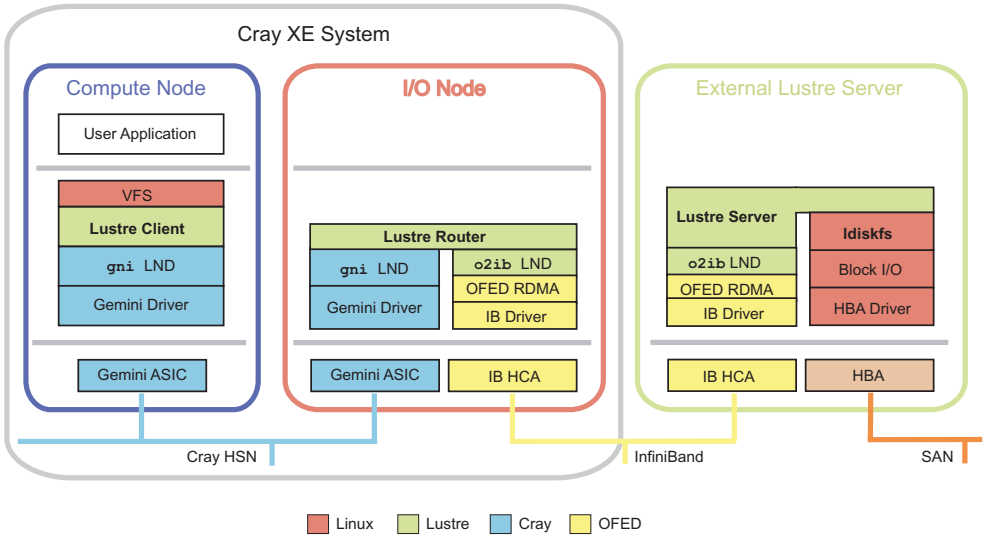
Figure 8. Cray System Connected to Storage Using SRP



12.2.2 Lustre Routing

Cray uses InfiniBand on the service nodes to connect Cray compute nodes to External Services File System (esFS) Lustre servers Figure 9. 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 your Cray service representative regarding an esFS solution for your Cray system.

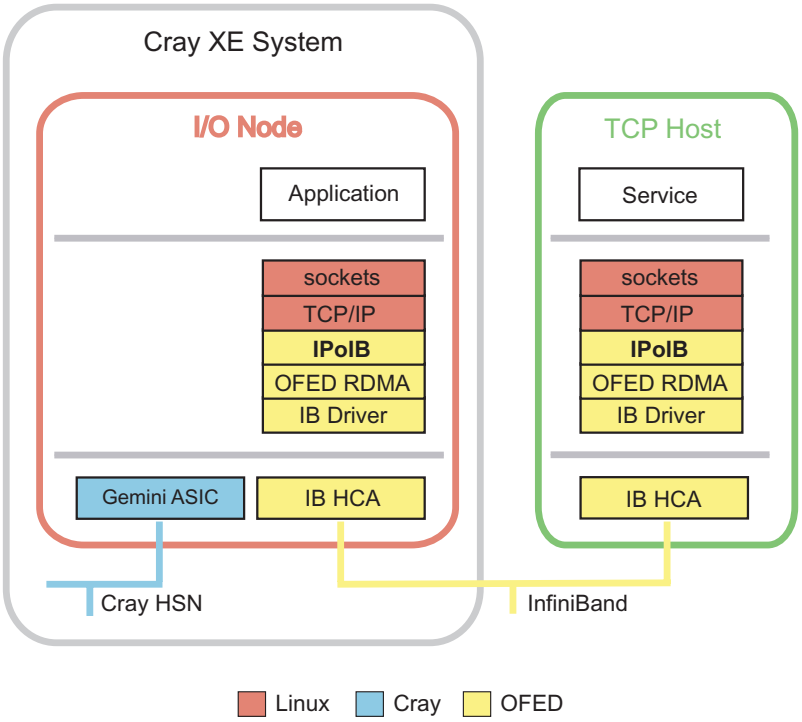
Figure 9. Cray Service Node Acting as an Infiniband Lustre Router



12.2.3 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 Figure 10.

Figure 10. Cray Service Node in IP over IB Configuration



12.3 Configuration

In addition to the OFED RDMA stack, Cray supports three upper layer protocols (ULPs) on its service nodes as shown in Table 14. Because all ULPs use the OFED stack, the InfiniBand Configuration (3.1) must be followed for all IB service nodes.

Note: It is only necessary to configure the specific ULPs that you intend to 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 14. 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 `o2iblnd`.
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.

12.4 InfiniBand Configuration

Procedure 78. Configuring 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. Change the value for `IPOIB_LOAD` to **`no`** to disable IPoIB services.

      ```
      IPOIB_LOAD=no
      ```

   c. The SCSI RDMA Protocol (SRP) driver loads by default on all nodes where IB services are configured to load at boot time. If a node does not need SRP services, change the value for `SRP_LOAD` to **`no`** to disable SRP.

      ```
      SRP_LOAD=no
      ```

5. Exit `xtopview`.

   ```
   node/27:/ # exit
   boot:~ #
   ```

   Note: You are prompted to type **`c`** and enter a brief comment describing the changes you made. To complete your comment, type **`Ctrl-d`** or a period on a line by itself. Do this each time you exit `xtopview` to log a record of revisions into an RCS system.

6. Proper IPoIB operation requires additional configuration. See Procedure 80 on page 315.

12.5 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 blade 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.

12.5.1 Starting OpenSM at Boot Time

Procedure 79. Starting a single instance of OpenSM on a service node at boot time

Note: This procedure assumes that the IB HCA is in node 8.

1. Use `xtopview` to access the service node that will host your 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
   ```

12.6 Internet Protocol over InfiniBand (IPoIB) Configuration

Procedure 80. Configuring 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) If you would like to configure IPoIB on both ports of a two port IB HCA, repeat step 2 and step 3 for `/etc/sysconfig/network/ifcfg-ib1`. Use a unique IP address from separate networks for each port.

12.7 Configuring SCSI RDMA Protocol (SRP) on Cray Systems

Procedure 81. Configuring and enabling SRP on Cray Systems

While in `xtopview` on the boot node, perform the following steps:

1. Edit `/etc/sysconfig/infiniband`

   ```
   default/:/ # vi /etc/sysconfig/infiniband
   ```

   and change the value of `SRP_DAEMON_ENABLE` to `yes`:

   ```
   SRP_DAEMON_ENABLE=yes
   ```

2. Edit `srp_daemon.conf` to increase the maximum sector size for SRP.

   ```
   default/:/ # vi /etc/srp_daemon.conf

   a       max_sect=8192
   ```

3. Edit `/etc/modprobe.conf.local` to increase the maximum number of gather-scatter entries per SRP I/O transaction.

   ```
   default/:/ # vi /etc/modprobe.conf.local

   options ib_srp srp_sg_tablesize=255
   ```

4. Exit from `xtopview`.

   ```
   default/:/ # exit
   boot:~ #
   ```

12.8 Lustre Networking (LNET) Router

Oracle provides the LNET layer as a separate transport for communication between the Lustre client and server. 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 *Lustre Operations Manual*.

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 XE (Gemini) systems and the OpenFabrics InfiniBand stack. Cray builds and distributes the OFED LND (`o2iblnd`) and `gnilnd` as part of its Lustre distribution.

Note: LNET routing is also available over GigE and 10GigE networks with `socklnd`, although this configuration does not support RDMA.

Routing Lustre requires that three types of nodes be configured: the router, the client, and the InfiniBand server. LNET uses IP addresses to identify LND ports. While `gnilnd` uses node IDs to enumerate its ports, `o2iblnd` uses IP addresses. As a result, IPoIB must be configured on each IB port. See Subnet Manager (OpenSM) Configuration on page 314 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. The network configuration is shown in Internet Protocol over InfiniBand (IPoIB) Configuration on page 315.

Table 15. LNET Network Address Configuration for Cray XE 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

12.8.1 Configuring the LNET Router

Procedure 82. Configuring 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 your `/etc/init.d/lnet` router controller (RC) script. An RC script is necessary to start LNET in the absence of any Lustre file services. A sample RC script is available in Appendix I, Sample LNET Router Controller Script on page 369.

   Note: Cray does not provide an RC script with its release packages. You must verify that this script will work for your configuration or contact your Cray service representative for more information.

   a. Create the `/etc/init.d/lnet` file in the default view.

      ```
      default/:/ # vi /etc/init.d/lnet
      ```

   b. Copy the sample script into your new file and write it.

3. Use `chkconfig` to enable LNET since 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"
   ```

   `o2ib` is the LNET name for the OFED LND and `gni` is the LNET name for the Gemini 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`.

   Note: Each Cray system sharing the external Lustre file system must have a unique `gni` identifier for the LNET options. In this case, the Cray XE is using `gni0`. Other systems would use other numbers to identify their Gemini 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 lnet check_routers_before_use=1
   options lnet router_ping_timeout=5
   options lnet dead_router_check_interval=60
   options lnet live_router_check_interval=60
   ```

   Note: You will need to configure these values for your specific system size.

6. Exit from `xtopview`.

You are prompted to add a comment about the operations you have performed. Enter **c**, and then enter a brief comment about the changes you made to the file.

Procedure 83. Manually controlling LNET routers

- If `/etc/init.d/lnet` is not provided, send the following commands to each LNET router node to control them manually:
  - Startup:

    ```
    modprobe lnet
    lctl net up
    ```

  - Shutdown:

    ```
    lctl net down
    lustre_rmmod
    ```

12.8.2 Configuring the InfiniBand Lustre Server

Procedure 84. Configuring the InfiniBand Lustre Server

The host on the other edge of the IB fabric must be configured to use the router nodes. Add an `lnet routes` directive for each Cray system sharing the external file system. Ensure that the LNET identifier is unique for each system (for example, `gni0`, `gni1`) and maps to the correct IP address for the router. Perform these steps on the remote host:

1. Edit `/etc/modprobe.conf` on the remote host to include the route to the LNET network.

   ```
   options lnet networks=o2ib(ib0)
   options lnet routes="gni0 10.10.10.[28,32]@o2ib"
   ```

   (Optional) 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.

2. Make `/etc/modprobe.conf` consistent with the changes made in Procedure 82 on page 317 by adding the following LNET directives:

   ```
   options lnet check_routers_before_use=1
   options lnet router_ping_timeout=5
   options lnet dead_router_check_interval=60
   options lnet live_router_check_interval=60
   ```

   Note: You will need to configure these values for your specific system size.

Because Lustre is running on the external host, there is no need to start LNET explicitly.

12.8.3 Configuring the LNET Clients

Procedure 85. Configuring the LNET clients

Since compute nodes are running the Lustre client, they do not need explicit commands to start LNET. There is, however, additional configuration required to get LNET to use the routers on the service nodes to reach the external servers. 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 networks` 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 networks=gni0
   options lnet routes="o2ib [27,31]@gni0"
   ```

2. Modify `/etc/fstab` in the compute node boot image to identify the external server. The format indicates the IP address of the external server and the LNET network used to reach it.

   ```
   10.10.10.1@o2ib:/boss1  /mnt/boss1      lustre  rw,flock
   ```

   Here, the `fstab` mount option `rw` gives read/write access to the client node. The additional `flock` option is to allow Lustre's client node to have exclusive access to the file lock.

   In this example, the Lustre file system with the `fsname "boss1"` is exported by the Lustre metadata server (MDS) on the InfiniBand fabric at IP address 10.10.10.1. Because both routers have access to this subnet, the Lustre client performs a round-robin with its requests to the routers.

3. Update the boot image to include these changes; follow the steps in Procedure 3 on page 66.

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.

SMW and CLE System Administration Commands [A]

In addition to the SUSE Linux Enterprise Server (SLES) commands available to you, this appendix lists the Cray developed commands for administering CLE on your 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 Management Services (CMS) commands invoked from the SMW for CMS administration. CMS commands are provided with SMW release packages. For more information about CMS commands, see *Using Cray Management Services (CMS)* provided with SMW release packages.

- Cray Linux Environment (CLE) commands invoked from a node to control the service and compute partitions; CLE commands are provided with CLE release packages.

A.1 HSS Commands

Table 16 shows the HSS commands and their functions.

Table 16. HSS Commands

Command	Description
`dbMonitor`	Controls the monitor process script that starts during system boot to watch `mysqld` and restart `mysqld` if it should crash
`getSedcLogValues`	Displays specified `sedc_manager` log file records
`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.
`nid2nic`	Prints all *nid*-to-*nic* address mappings.
`rtr`	Routes the Cray network

Command	Description
`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 CMS software on `bootroot` and `sharedroot` for system sets with CLE software installed
`xtalive`	Gets a response from an HSS daemon
`xtbootdump`	Parses a *bootinfo-file* to determine if `xtdumpsys` needs to be invoked
`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
`xtcheckmac`	Checks for duplicate MAC addresses among L1 and L0 controllers
`xtclass`	Displays the network topology class for this system
`xtclean_logs`	Removes HSS log files based on age
`xtclear`	Clears component flags in the state manager
`xtclear_link_alerts`	Clears alert flags
`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
`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
`xtdaemonconfig`	Configures HSS daemons dynamically
`xtdimminfo`	Collects and displays summaries from hardware errors reported in the console file
`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 a Cray system

Command	Description
xtgenid	Generates HSS physical IDs
xtgetsyslog	Retrieves the /var/log/messages file from L1 or L0 controllers
xthb	Reports on-chip heartbeats
xthwerrlog	Reports hardware errors
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
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
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 (L1) and/or blade control processors (L0)
xtsedcviewer	Command-line interface for SEDC
xtshow	Shows components with selected characteristics
xtwarmswap	Allows Cray XE modules to be warm swapped
xtwatchsyslog	Shows all log messages for cabinet control processors (L1 controllers) and blade control processors (L0 controllers)

A.2 CLE System Administration Commands

Table 17 shows CLE commands and their functions.

Table 17. CLE Commands

Command	Description
apmgr	Provides interface for ALPS to cancel pending reservations.
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).
generate_config.sh	Generates the Lustre file system configuration file.
lastlogin	Records last date on which each user logged in
lbcd	Invokes the load balancer client daemon.
lcrash	Used to analyze a dump file generated by the ldump command.
lbnamed	Invokes the load balancer service daemon.
ldump	Dumps node memory to a file. This is later analyzed with the lcrash command. ldump may be used to dump service nodes and compute nodes running CNL. The ldump command is run on the SMW.
lustre_control.sh	Manages Lustre file systems using standard Lustre commands and a site specific Lustre file system definition file.
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.
xt-lustre-proxy	Invokes the Lustre startup/shutdown, health monitor, and automatic failover utility.
xtalloc2db	Converts a text file to the alloc_mode table in the Service Database (SDB).
xtattr2db	Converts a text file to the attributes table in the Service Database (SDB).
xtauditctl	Distributes auditctl 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.

Command	Description
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 alloc_mode table in the Service Database (SDB) to a text file.
xtdb2attr	Converts the attributes table in the Service Database (SDB) to a text file.
xtdb2etchosts	Converts service information in the SDB to a text file.
xtdb2filesys	Converts the filesystem table of the SDB to a text file.
xtdb2lustrefailover	Converts the lustre_failover table in the SDB to a text file.
xtdb2lustreserv	Converts the lustre_serv table of the SDB to a text file.
xtdb2nodeclasses	Converts the service_processor table of the SDB to a text file.
xtdb2order	Converts the processor table od_allocator_id field in the Service Database (SDB) to a text file.
xtdb2proc	Converts the processor table of the SDB to a text file.
xtdb2segment	Converts segment table in the Service Database (SDB) to a text file.
xtdb2servcmd	Converts the service_cmd table of the SDB to a text file.
xtdb2servconfig	Converts the service_config table of the SDB to a text file.
xtdbsyncd	Invokes the HSS/SDB synchronization daemon.
xtfilesys2db	Converts a text file to the SDB filesystem table.
xtfsck	Checks file systems on a system set defined in /etc/sysset.conf.
xtgetconfig	Gets configuration information from /etc/sysconfig/xt file.
xthotbackup	Creates a backup copy of a system set on the boot RAID.
xthowspec	Displays file specialization in the shared root directory.
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 lustre_failover table.
xtlustreserv2db	Converts a text file to the SDB lustre_service table.
xtnce	Displays or changes the class of a node.
xtnodeclasses2db	Converts a text file to the service_processor table in the SDB.

Command	Description
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 xtopview 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 od_allocator_id field of the processor table in the Service Database (SDB).
xtpackage	Facilitates creation of boot images.
xtpkgvar	Creates a skeleton structure of /var.
xtproc2db	Converts a text file to the processor 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 module files that are loaded at login.
xtrsipcfg	Generates and optionally installs the necessary RSIP client and server configuration files.
xtsegment2db	Converts a text file to segment table in the Service Database (SDB).
xtservcmd2db	Converts a text file to the service_cmd table of the SDB.
xtservconfig	Adds, removes, or modifies the service_config table of the SDB.
xtservconfig2db	Converts a text file to the service_config table of the SDB.
xtshutdown	Shuts down the service nodes in an orderly fashion.
xtspec	Specializes files for nodes or classes.
xtunspec	Unspecializes files for nodes or classes.
xtverifyconfig	Verifies the coherency of /etc/init.d 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 States [B]

Table 18 defines state definitions for system components. States are designated by uppercase letters. Table 19 shows states that are common to all components.

Note: The state of `off` means that a component is present on the system. If the component is an L0, node, or ASIC, then this will also mean that the component is powered off. If you disable a component, the state shown becomes `disabled`. When you use the `xtcli enable` command 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`.

Table 18. State Definitions

State	L1	L0	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 19. Additional State Definitions

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.

Table 20 shows the states by component for which the `xtcli` commands run.

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 re-enabled.

Note: Administrative states are hierarchal, so disabling or enabling a component has a cascading effect on that components children (a GPU is the child of a node). A component may not be enabled if its parent component is disabled, but a subcomponent may be disabled without affecting its parents.

Table 20. `xtcli` Commands and Allowed States

`xtcli` Command	Subcommand	L1 Controller	L0 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

Error Codes [C]

Table 21 shows the Cray system error codes for the platforms supported with this release. When a Cray system error occurs, the related message is displayed on the SMW.

You can also use the `xterrorcode` command on the SMW to display a single error code or the entire list of error codes. 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 noted in the following table for example:

```
smw:~> xterrorcode 131279
Maximum error code (RS_NUM_ERR_CODE) is 297
code = 207, string = 'node voltage fault'
```

Table 21. System Error Codes

Code	Meaning
0	Success
1	L1 event aborted
2	Cabinet Power Up Fault
3	Cabinet Power Down Fault
4	Cabinet Emergency Power Off Fault
5	Cabinet Systems Daemon Aborted
6	Cabinet MicroController Communications Fault
7	Cabinet Uplink Fault
8	L1 to Event Router Communications Fault
9	Cabinet Cage Communications Fault
10	Cabinet Fan VFD Communications Fault
11	Cabinet Exhaust Pod Communications Fault
12	Cabinet Fan Pod Communications Fault
13	Cabinet Valere Communications Fault
14	Cabinet Received Invalid L0 ID
15	Cabinet Received Unexpected Event
16	Cabinet Blocked Inlet Fault

Code	Meaning
17	Cabinet Fan Overload Fault
18	Cabinet L1 Micro Initial Test Fault
19	Cabinet Set Thermal Mode Fault
20	Cabinet Calibration Valve Fault
21	Cabinet Fan RPM Fault
22	Cabinet (L1) Controller Core Temp Fault
23	Cabinet Exhaust Air Over Temp Fault
24	Cabinet Inlet Air Over Temp Fault
25	Fan Pod Temperature Fault
26	Cabinet 5V Fault
27	Cabinet 3.3V Fault
28	Cabinet 2.5V Fault
29	Cabinet Sensor Check Fault
30	Cabinet Low Air Pressure Fault
31	L0 Heartbeat Fault
32	Cabinet Received Unexpected L0 Heartbeat
33	Cabinet Slot Up Fault
34	Cabinet Slot Down Fault
35	Invalid Fan Pod ID
36	Cabinet Power Up Warning
37	Cabinet Power Down Warning
38	Cabinet Slot Up Warning
39	Cabinet Slot Down Warning
40	Cabinet Power Transition Time-out
41	Cabinet Power Down Time-out
42	Cabinet Slot Up Time-out
43	Cabinet Slot Down Time-out
44	Cabinet Fan Speed Warning
45	Cabinet Controller Temp Warning
46	Cabinet Exhaust Temp Warning
47	Cabinet Inlet Temp Warning
48	Fan Pod Temperature Warning
49	Cabinet 5V Warning

Code	Meaning
50	Cabinet 3.3V Warning
51	Cabinet 2.5V Warning
52	Cabinet Sensor Check Warning
53	Cabinet Calibration Valve Warning
54	Cabinet Low Air Pressure Warning
55	L0 Not Responding Warning
56	L1 Received Invalid State Response
57	Too Many Faults to Route
58	`l0rtrd`: Out of Memory
59	`l0rtrd`: Software Error
60	`l0rtrd`: MMR Access Failed
61	Routing Configuration Error
62	LCB PLL Lock Timeout
63	LCB Write Failed
64	LCB Initialization Failed
65	LCB/routing Phase 1 Timeout
66	LCB/routing Phase 2 Timeout
67	LCB/routing Phase 3 Timeout
68	Router Initialization Timeout
69	LCB link inactive
70	LCB Read Failed
71	L1 Heartbeat Failed
72	Request format error, or invalid flash partition/PIC device
73	System call on target L0 or L1 failed
74	Image file was inaccessible or bad checksum
75	An operation has timed out
76	No space on L0/L1 ramdisk for images
77	Burn/erase of flash/PIC failed
78	Internal inconsistency
79	`ev_len` is not big enough
80	There are no svc ids
81	There are duplicate svc ids
82	Only L0 svc ids are allowed

Code	Meaning
83	Only one bulk transfer context is allowed
84	Badly formed bulk transfer context
85	No filename in the bulk transfer context
86	No valid databuf pointer in the bulk transfer context
87	Datalen is zero in the bulk transfer context
88	`BULK_FLAG_WRITE` must be set in the bulk transfer context
89	`bulk_data_compute_bytes_needed()` failed
90	For L0 svc id, file transfers only
91	For node svc id, memory transfers only
92	Invalid characters found in filename
93	Unable to setup L0 <-> RCA config area
94	Unable to un-setup L0 <-> RCA config area
95	MMR or Memory write failed
96	MMR or Memory read failed
99	`ioctl()` to `/dev/l0sys` failed
100	Unable to load program into processor memory
101	Unable to load dram config info
102	Heartbeat check failed
103	Coldstart failed to complete
104	Shell parsing routine failed
105	Unable to create file
106	Unable to open file
107	Unable to write file
108	Unable to read file
109	`fstat()` failed
110	Unable to spawn process
111	`i2c` write failed
112	`i2c` read failed
113	Invalid L0 board type
114	Unable to reset jtag
117	RCA channel not open
118	Write to RCA channel failed
119	Event is too big to go across RCA channel

Code	Meaning
120	Bulk transfer request is too big
121	Unsupported ui event
122	Internal power manager failure
123	Power sequence is already in progress
124	Power sequence aborted
125	Power sequence timeout
126	Bulk transfer address must be aligned on a 32-bit boundary
127	Bulk transfer datalen must be a multiple of 4
130	Diag generic error
131	Software error
132	Diag hw error
133	Diag init error
134	Diag system call failure
135	Failed to allocate memory
136	Invalid diag option or argument
137	Invalid NULL pointer
138	Diag linked function error
139	Unable to remove file
140	Unable to close file
145	`memtest` init error
146	`memtest` sw error
147	`memtest` hw error
148	`cpuburn` init error
149	`cpuburn` sw error
150	`cpuburn` hw error
151	Invalid state for requested power command
152	Item removed from power sequence due to upstream fault or state
153	SM failed clear flag request by UI
154	SM failed set flag request by UI
155	SM failed set enable state request by UI
156	SM failed set empty state request by UI
157	SM failed set disable state request by UI

Code	Meaning
159	HSN link invalid state for routing
160	No matching ID found in State or Trans request
164	Request list id has invalid type
165	Power sequence does not apply to this item
166	HSN boot failed
167	Boot manager timeout
168	Node is not in bootable state
169	Node is not in haltable state
170	Boot manager internal error
171	Unable to seek to specified file position
172	Boot manager memory error
173	Boot manager CPI/O package error
174	Boot node daemon initialization error
175	Boot node daemon unknown error
176	Item state 'error' is invalid for requested command
177	Item state 'off' is invalid for requested command
178	Item state 'on' is invalid for requested command
179	Item state 'standby' is invalid for requested command
180	Item state 'disable' is invalid for requested command
181	Item state 'ready' is invalid for requested command
182	Item state 'diag' is invalid for requested command
183	Item state 'halt' is invalid for requested command
184	Item state 'NA' is invalid for requested command
185	Item state 'empty' is invalid for requested command
186	Item state 'unknown' is invalid for requested command
188	Invalid request
189	Alert flag is set
190	Cabinet Inlet Air Under Temp Fault
191	The Event has too many ids
203	Memory comparison failed
204	Node heartbeat fault
206	SM was not the target of the event
207	Node voltage fault

Code	Meaning
208	Node temperature fault
209	Node health check fault
210	Mezzanine voltage fault
211	Mezzanine temperature fault
212	Mezzanine health check fault
216	VERTY health check fault
217	Invalid partition id in id list
218	Invalid state for this operation
219	Component not found
220	Invalid or missing parameter or id
221	Generating SM data failed
222	Id is not in the partition
223	SM partition config update failed
224	Unable to obtain state report from this target
225	Received state report from a previously failed target
226	Mismatch in the target state report and our state information
227	Generating partition data failed
228	Locking components failed
229	Failed to process received event
231	Alert set on component from UI
232	Alert on component as SS is in alert
233	Alert received from Valere module
234	Cabinet door security breach
235	Cabinet doors secure
236	Item state 'enable' is invalid for requested command
237	Power manager failed to get the lock for the components requested for this session
238	Session aborted by user request
239	Opteron Built-In Self Test failed
240	Boot/SDB node lookup failed
241	Boot/SDB node has invalid state
242	Boot/SDB node is not a member of this partition
243	One of the partition member id is already in other partition

Code	Meaning
244	Invalid partition state
245	Partition is not configured
246	Invalid id found in Boot/SDB list
247	Partition member lookup failed
248	Invalid partition member state found
249	Invalid id found in partition request id list
250	Invalid partition state: partition is not active
251	Requested ids spanning multiple partitions
252	L1 cage VRM fault
253	Flash operation already in progress
254	Flash attempted with write protect enabled
255	L0 hotswap lever activated
256	Loadable library not found
257	Symbol not found
259	Invalid command in attribute event
260	Invalid module type
261	Failed to set an attribute
262	`i2c` open failed
263	Sending `ec_ssdc_req` to L0 failed
264	Parsing ssdc config file failed
265	Parsing `ec_ui_ssdc_req` failed
266	Invalid ssdc config in L0 request
267	No register config found in SSDC conf
268	Sending `ec_sedc_req` to L0 failed
269	Parsing SEDC config file failed
270	Parsing `ec_ui_sedc_req` failed
271	Invalid SEDC config in L0/L1 request
272	No register config found in SEDC conf
273	Generating SEDC data event failed
274	SEDC UI registration failed
277	Error in component name
278	Daemon(s) not running on target controller
279	Error in allocation of memory

Code	Meaning
287	Module power up failed on blades
288	Module power down failed on blades
295	`ec_route_phase3` failed
296	Invalid configuration specification
297	JTAG write failed
298	JTAG read failed
299	JTAG lock failed
300	JTAG failed
301	LCB Link active, is this a live system?
305	Invalid subcommand
306	Lock operation failed
307	Operation not permitted, lock in effect
308	PCI voltage fault
309	PCI health check fault
310	L1 micro flash read failed
311	L1 micro flash write failed
312	L1 micro flash erase failed
313	L1 micro flash verification failed
314	L1 micro in a bad state for flash operation
315	L1 micro in unknown state
316	L1 micro cannot be flashed, no bootloader present
317	Cabinet fan VFD failure
318	Cabinet fan underload
319	Cabinet over-current condition
320	Cabinet under-voltage condition
321	Node does not accept shutdown events
322	L0 quiesce is already active
323	L0 quiesce is not active
324	Node does not accept quiesce events
325	Node not running for quiesce/unquiesce
326	Node timed out waiting for quiesce
327	Node timed out waiting for unquiesce
328	Cabinet power controller communication fault

Code	Meaning
329	Gemini BIST failure
330	Gemini CCLK/LCLK PLL lock timeout
331	Gemini SerDes PLL lock timeout
332	Gemini SMS timeout
333	Gemini SMS failure
334	CPU Model Mismatch
335	CPU Speed Mismatch
336	Memory Speed Mismatch
337	Memory Configuration Mismatch
338	Mezzanine Configuration Mismatch
339	Role not changed: node on service blade (HW)
340	Role not changed: invalid node state
341	Role not changed: DB node not updated
342	Database update failed
343	Database open failed
344	Memory G34 DIMM Mismatch
345	Memory G34 DIMM SPD Data Mismatch
346	SM failed set enable. Parent(s) not enabled.
347	Warning: SM state change successful, but forced (`-f`).
348	Component state not disabled
349	Component state not enabled
350	Component state not empty
351	Gemini: state not enabled
353	Gemini LCB lane(s) reinit failed
354	Invalid `cpio` path length
355	Cabinet accelerometer failure
356	Accelerometer not supported in L1 Micro Firmware Ver
357	Partition node not child of partition member
358	Incomplete partition definition
359	Invalid cpio path
360	Duplicate partition
361	No service nodes in id list
362	No compute nodes in id list

Code	Meaning
364	CPU core count Mismatch
365	Invalid blade state for operation
366	PXE config file generation failed
367	Module VERTY type mismatch
368	SXM (GPU) module enabled but not present
369	SXM (GPU) PCIe link training failed
370	SXM (GPU) PCIe link speed mismatch
371	SXM (GPU) PCIe link width mismatch
372	Node BIOS communication error
373	Node BIOS synchronization error
374	SerDes NP startup failure
375	SerDes NP in unexpected or invalid state

Remote Access to the SMW [D]

Virtual Network Computing (VNC) software enables you to view and interact with the SMW from another computer. The Cray system provides a VNC server, `Xvnc`; you must download a VNC client to connect to it. See RealVNC (http://www.realvnc.com/) or TightVNC (http://www.tightvnc.com/) for more information.

Note: The VNC software requires a TCP/IP connection between the server and the viewer. Some firewalls and site security do not allow this connection.

Cray supplies a VNC account `cray-vnc`.

Procedure 86. Starting 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. Disable `xinetd` startup of `Xvnc`.

   If the `chkconfig` command you executed in step 2 reports that `Xvnc` was started by INET services (`xinetd`):

   ```
   smw:~ # chkconfig vnc
   vnc xinetd
   ```

   Execute the following commands to disable `xinetd` startup of `Xvnc` (`xinetd` startup of `Xvnc` 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 `failed`, this is normal and you can ignore the `failed` notification.

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 you for one. You must 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-xt: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-xt:1.log

smw:~ # ps -eda | grep vnc
1839 pts/0    00:00:00 Xvnc
```

Note: The startup script starts the `Xvnc` server for display `:1`.

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`. Many clients allow you to specify whether you want to connect in view-only or in an active mode. If you choose active participation, every mouse movement and keystroke made in your client is sent to the server. If more than one client is active at the same time, your typing and mouse movements are intermixed.

Note: 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.

Procedure 87. 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 you have the `vncviewer -via` option available. If you do not, use Procedure 88 on page 343.

- If you are connecting from a workstation or laptop running Linux, enter the `vncviewer` command shown below.

  The first password you enter is for `crayadm` on the SMW. The second password you enter is for the VNC server on the SMW, which was set when the VNC server was started for the first time using `/etc/init.d/vnc start` on the SMW.

```
/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
```

Procedure 88. 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. You will be prompted 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. You will be prompted 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.

Procedure 89. 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. You will be prompted 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. You will be prompted 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.

   If you type this on the Mac OS X command line after having prepared the SSH tunnel, the vncviewer will pop up:

   ```
   local_mac_prompt% open vnc://localhost:5902
   ```

The VNC window from the SMW appears. All traffic between the `vncviewer` on the local Mac OS X computer and the VNC server on the SMW is now encrypted through the SSH tunnel.

Procedure 90. For workstation or laptop running Windows: Connecting to the VNC server through an `ssh` tunnel

Note: If you are connecting from a computer running Windows, then 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 Procedure 88 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: Details about how to create the SSH tunnel vary amongst the different SSH programs for Windows computers.

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 click on the **Connect** button.

Updating the Time Zone [E]

When you install the Cray Linux Environment (CLE) operating system, the Cray system time is set at US/Central Standard Time (CST), which is six hours behind Greenwich Mean Time (GMT). You can change this time.

Note: 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 a CNL image, follow the appropriate procedure below.

Procedure 91. Changing the time zone for the SMW and the L1 and L0 controllers

Warning: Perform this procedure while the Cray system is shut down; do not flash L0 and L1 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 L0 and L1 controllers are responding.

```
smw:~ # xtalive -a l0sysd s0
```

2. Check the current time zone setting for the SMW and controllers.

```
smw:~ # date
Wed Aug 25 21:30:06 CDT 2010

smw:~ # xtrsh -l root -s /bin/date s0
c0-0c0s2 : Wed Aug 25 21:30:51 CDT 2010
c0-0c0s5 : Wed Aug 25 21:30:51 CDT 2010
c0-0c0s7 : Wed Aug 25 21:30:51 CDT 2010
c0-0c1s1 : Wed Aug 25 21:30:51 CDT 2010
.
.
.
c0-0 : Wed Aug 25 21:30:52 CDT 2010
```

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 25 21:32:52 CDT 2010
smw:~ # /usr/sbin/zic -l America/New_York
smw:~ # date
Wed Aug 25 22:33:05 EDT 2010
```

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` directory to `/opt/tftpboot` and restart `rsms`.

```
smw:~ # cp /etc/localtime /opt/tftpboot
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 perform step 8.

   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 l1
crayadm@smw:~> xtflash -t l1
```

   c. Erase the flash memory of the L0s and flash the updated time zone.

```
crayadm@smw:~> fm -w -t l0
crayadm@smw:~> xtflash -t l0
```

   d. Check the current time zone setting for the SMW and controllers.

```
crayadm@smw:~> date
Wed Aug 25 23:07:07 EDT 2010
crayadm@smw:~> xtrsh -l root -s /bin/date s0
c0-0c1s1 : Wed Aug 25 23:07:16 EDT 2010
c0-0c0s7 : Wed Aug 25 23:07:16 EDT 2010
c0-0c1s3 : Wed Aug 25 23:07:16 EDT 2010
.
.
.
c0-0 : Wed Aug 25 23:07:17 EDT 2010
```

8. If the time zone has been changed already, complete this step. If this is the first time the time zone has been modified, perform step 7.

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
```

Procedure 92. Changing the time zone on the boot root and shared root

Perform the following steps to change the time zone. You must be logged on as `root`. In this example, the time zone is changed from `"America/Chicago"` to `"Europe/London"`.

1. Confirm the time zone setting on the SMW.

```
smw:~ # cd /etc/sysconfig
smw:~ # grep TIMEZONE clock
TIMEZONE="Europe/London"
DEFAULT_TIMEZONE="Europe/London"
```

2. Log on to the boot node.

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

3. Verify that the `zone.tab` file in the `/user/share/zoneinfo` directory contains the time zone you want to set.

```
boot:~ # cd /usr/share/zoneinfo
boot:~ # grep Europe/London zone.tab
GB +512830-0001845 Europe/London Great Britain
```

4. Create the time conversion information files.

```
boot:~ # date
Fri Mar 10 05:19:38 CST 2007
boot:~ # /usr/sbin/zic -l Europe/London
boot:~ # date
Fri Mar 10 11:21:31 GMT 2007
```

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:~ # cd /etc/sysconfig
boot:~ # grep TIMEZONE clock
TIMEZONE="America/Chicago"
DEFAULT_TIMEZONE="America/Chicago"
boot:~ # vi clock
make changes
boot:~ # grep TIMEZONE clock
TIMEZONE="Europe/London"
DEFAULT_TIMEZONE="Europe/London"
```

6. Switch to the default view by using `xtopview`.

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

```
boot:~ # xtopview
default/:/ #
```

7. Verify that the `zone.tab` file in the `/user/share/zoneinfo` directory contains the time zone you want to set.

```
default/:/ # cd /usr/share/zoneinfo
default/:/ # grep Europe/London zone.tab
GB +512830-0001845 Europe/London Great Britain
```

8. Create the time conversion information files.

```
default/:/ # date
Fri Mar 10 05:22:38 CST 2007
default/:/ # /usr/sbin/zic -l Europe/London
default/:/ # date
Fri Mar 10 11:24:31 GMT 2007
```

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/:/ # grep TIMEZONE clock
TIMEZONE="America/Chicago"
DEFAULT_TIMEZONE="America/Chicago"
default/:/ # vi clock
make changes
default/:/ # grep TIMEZONE clock
TIMEZONE="Europe/London"
DEFAULT_TIMEZONE="Europe/London"
```

10. Exit `xtopview`.

```
default/:/ # exit
boot:~ #
```

Procedure 93. Changing the time zone for compute nodes

1. Confirm the time zone setting on the SMW.

```
smw:~ # cd /etc/sysconfig
smw:~ # grep TIMEZONE clock
TIMEZONE="America/Chicago"
DEFAULT_TIMEZONE="America/Chicago"
```

2. Copy the new `/etc/localtime` file from the SMW to the bootimage template directory.

```
smw:~# cp -p /etc/localtime
/opt/xt-images/templates/default/etc/localtime
```

3. Update the boot image to include these changes; follow the steps in Procedure 3 on page 66.

   The time zone is not changed until you boot the compute nodes with the updated boot image.

Creating Modulefiles [F]

This appendix provides a template and an example of a modulefile that you can use as you construct modulefiles for your site.

F.1 Modulefile Template

The following listing provides a template of the elements required in a modulefile. Use this as your guide when creating your own modulefiles.

```
#%Module#####################################################################
#
# Generic modulefile template
#

###
# Add your verbiage into ModulesHelp area. This information
# will be seen by those invoking
# module help [my_product]
###

proc ModulesHelp { } {
puts stderr "This modulefile defines the library paths and"
puts stderr "include paths needed to use "
puts stderr "[my_product]."
puts stderr ""
}

###
# [my_product] is the name consistently used in the modulefile
# to set environment variables. It may be the same name as
# the modulefile and the rpm, however the modulefile and rpm
# will be named in a lower case name while [my_product] should
# be upper case, i.e. "module load acml" and ACML_DIR.
###

set is_module_rm [module-info mode remove]

###
# If [my_product] will not be versioned, then set
# [my_product]_CURPATH to the location of [my_product].
# If you use versions, then you only need to change one
# number as you create a module for another product version.
###

set [my_product]_LEVEL [product-version]
set [my_product]_CURPATH /opt/[installed-product-name]/$[my_product]_LEVEL
```

```
setenv [my_product]_DIR $[my_product]_CURPATH

###
# Add your executable to PATH.
###
#prepend-path PATH       $[my_product]_CURPATH/bin

###
# Add your dynamic library path. This is *NOT* for statically built
# libraries. For those use [my_product]_POST_LINK_OPTS below.
###
#prepend-path LD_LIBRARY_PATH  $[my_product]_CURPATH/lib

###
# Add MANPATH and INFOPATH
###
if { [file isdirectory $[my_product]_CURPATH/info] == 1} {
 prepend-path INFOPATH $[my_product]_CURPATH/info
}

if { [file isdirectory $[my_product]_CURPATH/man] == 1} {
 prepend-path MANPATH $[my_product]_CURPATH/man]
}

###
# To make our product work in commandline generation, you must
# add [my_product] to the PE_PRODUCT_LIST.
###
append-path  PE_PRODUCT_LIST  [my_product]

###
# The following 5 *_OPTS environment variables allow placement of compiler
# commandline options. The PRE and POST in the names refers to
# the location before or after the user-specified arguments.
# Remember that, in general, the linker evaluates its commandline from left
# to right, but the compiler generally uses the last argument in the list.
# The commandline is created for you in this order:
# cc [PRE_COMPILE_OPTS] [PRE_LINK_OPTS] user_args [POST_COMPILE_OPTS] \
#   [POST_LINK_OPTS] [INCLUDE_OPTS]
###

###
# Compiler options. The first character in this list must be a
# space and the list must be double quoted.
#
# You can define a fortran modules path for pgi-compiled files by adding it
# as a "-I' options to [my_product]_PRE_COMPILE_OPTS.

#setenv [my_product]_PRE_COMPILE_OPTS
#setenv [my_product]_POST_LINK_OPTS

###
# Options passed to the linker, including
# -L paths and -l library names. The -L and -l are used for statically built
# libraries. The first character in the list must be a space and the list
# must be double quoted. The -L and -l arguments should be added to
# [my_product]_POST_LINK_OPTS.
```

```
#setenv [my_product]_PRE_LINK_OPTS
#setenv [my_product]_POST_COMPILE_OPTS

#
# Include search path
#

#setenv [my_product]_INCLUDE_OPTS
```

Example 115. Modulefile example

This example shows a product, `kate`, with library files `libkate.a` and `libkit.a`, which were built with 64-bit PGI. Naming directories `pgi64` helps keep track of library formats. You can create whatever directory structure works for you. Likewise, naming the modulefile `kate-pgi` tells a potential user that this would be loaded when compiling using PGI.

```
#%Module#####################################################################
#
# kate-pgi modulefile
#

proc ModulesHelp { } {
puts stderr "This modulefile defines the library paths and"
puts stderr "include paths needed to use the pgi-compiled kate."
puts stderr "Libraries -libkate.a, libkit.a, libkate.so and compiler"
puts stderr "option, -Mprof=mpi, are added. The utility run-kate"
puts stderr "is added to PATH."
}

###
# The modulefile kate-gnu could load gnu-built kate libraries,
# which are defined at $KATE_CURPATH/gnu64/lib
###

set is_module_rm [module-info mode remove]

set KATE_LEVEL 2.0
set KATE_CURPATH /opt/kate/$KATE_LEVEL

prepend-path PATH       $KATE_CURPATH/bin
prepend-path LD_LIBRARY_PATH  $KATE_CURPATH/pgi64/lib
prepend-path MANPATH      $KATE_CURPATH/man

append-path  PE_PRODUCT_LIST  KATE

###
# Definitions for these must begin with a space.
# Remember that in general the linker evaluates its command-line
# options left to right, while the compiler takes the last one
# it detects. You can define a Fortran modules path for pgi compiler
# by adding it as a "-I" option to *_POST_COMPILE_OPTS.
###
setenv KATE_PRE_COMPILE_OPTS  " -Mprof=mpi"
setenv KATE_POST_LINK_OPTS  " -L $KATE_CURPATH/lib -lkate -lkit"
setenv KATE_POST_COMPILE_OPTS  " -I $KATE_CURPATH/fortran_modules_dir"
setenv KATE_INCLUDE_OPTS  " -I $KATE_CURPATH/include"
```

F.2 Sharing Your Modulefile

Add your modulefile to `/opt/modulefiles` or to another directory. If you use another directory, you must add the path to your environment by using a `module use` command; for example, `module use /my/module/path`. To make the new modulefile path available to all users, edit the file `/opt/modules/init/.modulespath`.

F.3 Modulefile Help

Using the `module` command, you can get online help about any module in your system:

```
# module help modulefile
```

PBS Professional Licensing for Cray Systems [G]

G.1 Introduction

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 you run 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 your needs and implement that solution prior to installing the PBS Professional software from Altair.

Note: Regardless of the option chosen, you must run a PBS Professional MOM daemon 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 you choose to run the PBS Professional scheduler and server on a login node, you should 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 that you consider them before selecting this option. Refer to Migrating the PBS Professional Server and Scheduler on page 356 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 you run 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 your system has larger node counts (> 8192), you may want to avoid this option. To configure PBS Professional for this strategy, refer to Migrating the PBS Professional Server and Scheduler on page 356.

- **Configuring the SDB node as an RSIP client.** This options allows you to leave the PBS Professional scheduler and server on the SDB node. If you are already running RSIP, this may be an attractive option. Cray recommends a dedicated network node for the RSIP server, which may not be desirable if you are not already running RSIP. Follow the appropriate procedure in Configuring RSIP to the SDB Node on page 358 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 you intend 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 Network Address Translation (NAT) IP Forwarding on page 361.
- **Installing a network card in the SDB node to connect it to the external network.** With this option you do not need to configure RSIP or NAT, but you must purchase a PCIe network interface card (NIC) for a modest cost. This is an attractive option if you want to access the SDB node directly from your external network. This procedure does not require connection via another node on the Cray system. The steps to configure this option are covered in Installing and Configuring a NIC on page 363.

Cray recommends that system administrators consult their local networking and security staff prior to selecting one of these options. Once you have chosen and configured a method for accessing the license server, 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.

G.2 Migrating the PBS Professional Server and Scheduler

Before migrating the PBS Professional server and scheduler off of the SDB node you must first 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 you already have a central PBS Professional server and scheduler, 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. You must modify the default configuration to ensure that the PBS Professional scheduler and server do not start automatically on the SDB node.

Procedure 94. Migrating PBS off 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, your 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 your 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
   boot:~#
   ```

4. 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
   ```

5. 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
   boot:~ #
   ```

6. 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 your server is named *myserver*, type the following commands:

   ```
   login2:~ # vi /var/spool/PBS/mom_priv/config
   $clienthost myserver.domain.com
   ```

7. 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 you have migrated them to a login node.

8. 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=xthostname
   Qmgr: create node mycrayxt2
   Qmgr: set node mycrayxt2 resources_available.mpphost=xthostname
   Qmgr: exit
   myserver:~
   ```

   At this point, the login nodes should be visible to the PBS Professional server.

G.3 Configuring RSIP to the SDB Node

Follow the instructions in this section to configure the SDB node as an RSIP client. Once 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 you have not configured RSIP on your system, follow Procedure 95 on page 359 to generate a simple RSIP configuration with a single server and only the SDB node as a client.

Using the `CLEinstall` Program to Install and Configure RSIP on page 207 includes procedures to configure RSIP on a Cray system using the `CLEinstall` installation program. If you have already configured RSIP using these procedures during your Cray Linux Environment (CLE) installation or upgrade, follow Procedure 96 on page 360 to add the SDB node as an RSIP client for one of your existing RSIP servers.

For additional information on configuring RSIP services, see Configuring Realm-specific IP Addressing (RSIP) on page 206.

Procedure 95. Creating a simple RSIP configuration with the SDB node as a client

Important: Cray strongly recommends Procedure 48 on page 208 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
      node/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:

      ```
      node/c0-0c0s6n1:/ # vi /etc/opt/cray/rsipd/rsipd.conf
      port_range 8192-60000
      max_clients 2
      Uncomment:
      ext_if eth0
      ```

      Note: If your external Ethernet interface is not `eth0`, modify `ext_if` accordingly. For example,

      ```
      ext_if eth1
      ```

   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*|ippp*|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 `krsip` module with the IP argument pointing to the HSN IP address of the RSIP server node; 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`, 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" }
```

Procedure 96. Adding the SDB node as an RSIP client to an existing RSIP configuration

1. Select one of your RSIP servers to provide RSIP access for the SDB node. In this example, we have chosen the RSIP server with the physical ID `c0-0c0s6n1`.

2. Log on to the boot node and invoke `xtopview` in the node view for the RSIP server you have selected.

```
boot:~ # xtopview -n c0-0c0s6n1
node/c0-0c0s6n1:/ #
```

3. Modify the `max_clients` parameters in the `rsipd.conf` file; Add 2 more clients to make room for the new SDB node. For example, if you configured 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" }
```

G.4 Network Address Translation (NAT) IP Forwarding

Follow Procedure 97 on page 361 to configure NAT IP forwarding for the SDB node.

Procedure 97. Configuring NAT IP forwarding for the SDB node

1. Select a login node to act as the NAT router. Cray recommends that you select the node with the lowest load or network latency. For this example the login node is named `login2`.

2. Log on to the node you have selected and invoke the `ifconfig` command to obtain the IP address of the routing node.

   If you have a Cray XE system with 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
boot:~#
```

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. Exit the `xtopview` session.

```
default/:/ # exit
boot:~#
```

8. 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
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state 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:~ #
```

9. 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 you identified in step 2 is *10.128.0.3*, type this command:

   ```
   login2:~ # ssh sdb /sbin/route add default gw 10.128.0.3
   ```

10. Test the new route by invoking the `ping` command and ensuring the service node can access external servers by name.

11. Edit the boot automation script to Configure NAT services. For example, if the IP address you identified in step 2 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.

G.5 Installing and Configuring a NIC

Obtain a PCIe compliant NIC. Intel 82546 based cards have been verified with Cray system SDB nodes. Follow Procedure 98 on page 363 to install the network card in the SDB node and connect it to the external network. Note that you are required to reboot your system as part of this procedure.

Procedure 98. Installing and configuring a NIC on the SDB node

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 your 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 to the ifcfg-eth0 file:
```
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
boot:~ #
```

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; that is, before the 192.168.x.y entries.
```
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
boot:~# 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. Confirm that you can ping the license server from the SDB node.

```
nid00003:~ #  ping tic.domain.com
```

Installing RPMs [H]

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.

H.1 Generic RPM Usage

To install RPMs on a Cray system, you must 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 you are installing modifies files in `/etc`, you must 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, you can set the `CRAY_INSTALL_DEFAULT` environment variable to configure the new version as the default. Set this variable before you install 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 you are installing.

Example 116. Installing an RPM on the SMW

As `root`, use the following command:

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

Example 117. Installing an RPM on the boot node root

As `root`, use the following command:

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

Example 118. Installing an RPM on the shared root

As `root`, use the following commands:

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

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

Sample LNET Router Controller Script [I]

```
#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
         ;;
    restart)
         $0 stop
         $0 start
         rc_status
         ;;
    *)
         echo "Usage: $0 {start|stop|restart}"
         exit 1
         ;;
esac
rc_exit
```