DirectMon™

Administrator Manual

Revision: 7610

Date: Fri, 16 Sep 2016

DataDirect™
NETWORKS

Table of Contents

Preface

Welcome to the *Administrator Manual* for the DirectMon™ cluster environment.

0.1 Quickstart

For readers who want to get a cluster up and running as quickly as possible with DirectMon, Appendix F is a quickstart installation guide.

0.2 About This Manual

The rest of this manual is aimed at helping system administrators install, understand, and manage a cluster running DirectMon so as to get the best out of it.

The *Administrator Manual* covers administration topics which are specific to the DirectMon environment. Readers should already be familiar with basic Linux system administration, which the manual does not generally cover. Aspects of system administration that require a more advanced understanding of Linux concepts for clusters are explained appropriately.

This manual is not intended for users interested only in interacting with the cluster to run compute jobs. The *User Manual* is intended to get such users up to speed with the user environment and workload management system.

Updated versions of the *Administrator Manual*, as well as the *User Manual*, are always available on the cluster at `/cm/shared/docs/cm`.

The manuals constantly evolve to keep up with the development of the DirectMon environment and the addition of new hardware and/or applications.

The manuals also regularly incorporate customer feedback. Administrator and user input is is greatly valued at DDN, so that any comments, suggestions or corrections will be very gratefully accepted at `manuals@ddn.com`.

0.3 Getting Administrator-Level Support

Unless the DirectMon reseller offers support, support is provided by DDN over e-mail via `support@ddn.com`. Section 12.2 has more details on working with support.

1 Introduction

1.1 What Is DirectMon?

DirectMon™ is a cluster management application built on top of a major Linux distribution. It is available for:

- Scientific Linux 5 and 6 (`x86_64` only)
- Red Hat Enterprise Linux Server 5 and 6 (`x86_64` only)
- CentOS 5 and 6 (`x86_64` only)
- SUSE Enterprise Server 11 (`x86_64` only)

This chapter introduces some basic features of DirectMon and describes a basic cluster in terms of its hardware.

1.2 Cluster Structure

In its most basic form, a cluster running DirectMon contains:

- One machine designated as the *head node*
- Several machines designated as *compute nodes*
- One or more (possibly managed) *Ethernet switches*
- One or more *power distribution units* (Optional)

The head node is the most important machine within a cluster because it controls all other devices, such as compute nodes, switches and power distribution units. Furthermore, the head node is also the host that all users (including the administrator) log in to. The head node is the only machine that is connected directly to the external network and is usually the only machine in a cluster that is equipped with a monitor and keyboard. The head node provides several vital services to the rest of the cluster, such as central data storage, workload management, user management, DNS and DHCP service. The head node in a cluster is also frequently referred to as the *master node*.

Often, the head node is replicated to a second head node, frequently called a passive head node. If the active head node fails, the passive head node can become active and take over. This is known as a high availability setup, and is a typical configuration (chapter 15) in DirectMon.

A cluster normally contains a considerable number of non-head, or *regular nodes*, also referred to simply as nodes.

Most of these nodes are *compute nodes*. Compute nodes are the machines that will do the heavy work when a cluster is being used for large computations. In addition to compute nodes, larger clusters may have other types of nodes as well (e.g. storage nodes and login nodes). Nodes can be easily installed through the (network bootable) node provisioning system that is included with DirectMon. Every time a compute node is started, the software installed on its local hard drive is synchronized automatically against a software image which resides on the head node. This ensures that a node can always be brought back to a "known state". The node provisioning system greatly eases compute node administration and makes it trivial to replace an entire node in the event of hardware failure. Software changes need to be carried out only once (in the software image), and can easily be undone. In general, there will rarely be a need to log on to a compute node directly.

In most cases, a cluster has a private internal network, which is usually built from one or multiple managed Gigabit Ethernet switches, or made up of an InfiniBand fabric. The internal network connects all nodes to the head node and to each other. Compute nodes use the internal network for booting, data storage and interprocess communication. In more advanced cluster setups, there may be several dedicated networks. Note that the external network (which could be a university campus network, company network or the Internet) is not normally directly connected to the internal network. Instead, only the head node is connected to the external network.

Figure 1.1 illustrates a typical cluster network setup.

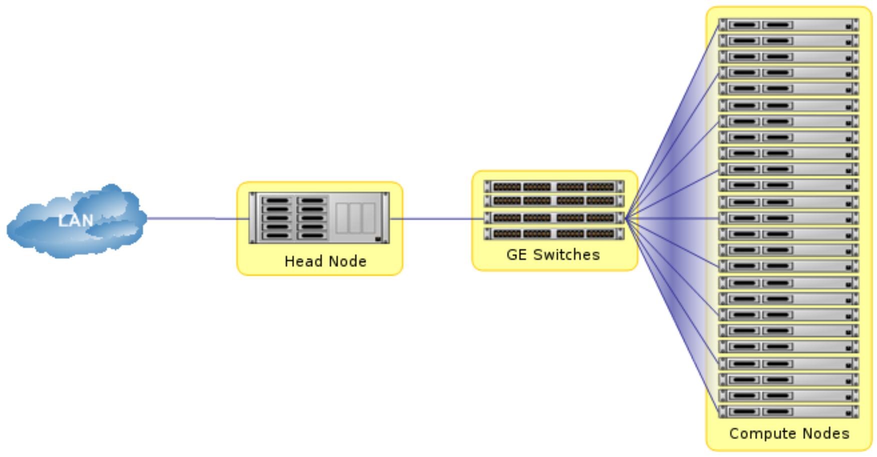


Figure 1.1: Cluster network

Most clusters are equipped with one or more power distribution units. These units supply power to all compute nodes and are also connected to the internal cluster network. The head node in a cluster can use the power control units to switch compute nodes on or off. From the head node, it is straightforward to power on/off a large number of compute nodes with a single command.

1.3 DirectMon Administrator And User Environment

DirectMon contains several tools and applications to facilitate the administration and monitoring of a cluster. In addition, DirectMon aims to provide users with an optimal environment for developing and running applications that require extensive computational resources.

- The administrator normally deals with the cluster software configuration via a front end to the DirectMon. This can be GUI-based (`cmgui`, section 3.4), or shell-based (`cmsh`, section 3.5). Other tasks can be handled via special tools provided with DirectMon, or the usual Linux tools. The use of DirectMon tools is usually recommended over standard Linux tools because cluster administration often has special issues, including that of scale.

  The *Administrator Manual* (this manual) covers how DirectMon is used by the administrator to do all this. Additionally, some more obscure configuration cases can often be found in the DDN knowledge base at `http://kb.brightcomputing.com`, along with some procedures that are not really within the scope of DirectMon itself, but that may come up as part of related general Linux configuration. Online support is also available (section 12.2).

- The user normally interacts with the cluster by logging into a custom Linux user environment to run jobs. Details on how to do this from the perspective of the user are given in the *User Manual*.

1.4 Organization Of This Manual

The following chapters of this manual describe all aspects of DirectMon from the perspective of a cluster administrator.

Chapter 2 gives step-by-step instructions to installing DirectMon on the head node of a cluster. Readers with a cluster that was shipped with DirectMon pre-installed may safely skip this chapter.

Chapter 3 introduces the main concepts and tools that play a central role in DirectMon, laying down groundwork for the remaining chapters.

Chapter 4 explains how to configure and further set up the cluster after software installation of DirectMon on the head node.

Chapter 5 describes how power management within the cluster works.

Chapter 6 explains node provisioning in detail.

Chapter 7 explains how to carry out cloudbursting.

Chapter 8 explains how accounts for users and groups are managed.

Chapter 9 explains how workload management is implemented and used.

Chapter 10 demonstrates a number of techniques and tricks for working with software images and keeping images up to date.

Chapter 11 explains how the monitoring features of DirectMon can be used.

Chapter 12 summarizes several useful tips and tricks for day to day monitoring.

Chapter 13 describes a number of third party software packages that play a role in DirectMon.

Chapter 14 describes how the Intel MIC architecture integrates with DirectMon.

Chapter 15 gives details and setup instructions for high availability features provided by DirectMon. These can be followed to build a cluster with redundant head nodes.

The appendices generally give supplementary details to the main text.

2 Installing DirectMon

This chapter describes the installation of DirectMon onto the head node of a cluster. Sections 2.1 and 2.2 list hardware requirements and supported hardware. Section 2.3 gives step-by-step instructions on installing DirectMon from a DVD onto a head node that has no operating system running on it initially, while section 2.4 gives instructions on installing onto a head node that already has an operating system running on it.

Once the head node is installed, the other, regular, nodes can (PXE) boot off the head node and provision themselves from it with a default image, without requiring a Linux distribution DVD themselves. The regular nodes normally have any existing data wiped during the process of provisioning from the head node. The PXE boot and provisioning process for the regular nodes is described in Chapter 6.

The installation of software on an already-configured cluster running DirectMon is described in Chapter 10.

2.1 Minimal Hardware Requirements

The following are minimal hardware requirements:

2.1.1 Head Node

- Intel Xeon or AMD Opteron CPU (64-bit)
- 2GB RAM
- 80GB diskspace
- 2 Gigabit Ethernet NICs (for the most common Type 1 topology (section 2.3.6))
- DVD drive

2.1.2 Compute Nodes

- Intel Xeon or AMD Opteron CPU (64-bit)
- 1GB RAM (at least 4GB is recommended for diskless nodes)
- 1 Gigabit Ethernet NIC

2.2 Supported Hardware

The following hardware is supported:

2.2.1 Compute Nodes

- SuperMicro
- Cray
- Cisco
- Dell
- IBM
- Asus
- Hewlett Packard

Other brands are also expected to work, even if not explicitly supported.

2.2.2 Ethernet Switches

- HP ProCurve
- Nortel
- Cisco
- Dell
- SuperMicro
- Netgear

Other brands are also expected to work, although not explicitly supported.

2.2.3 Power Distribution Units

- APC (American Power Conversion) Switched Rack PDU

Other brands with the same SNMP MIB mappings are also expected to work, although not explicitly supported.

2.2.4 Management Controllers

- IPMI 1.5/2.0
- HP iLO 1/2/3

2.2.5 InfiniBand

- Mellanox HCAs, and most other InfiniBand HCAs
- Mellanox InfiniBand switches
- QLogic (Intel) InfiniBand switches
- Most other InfiniBand switches

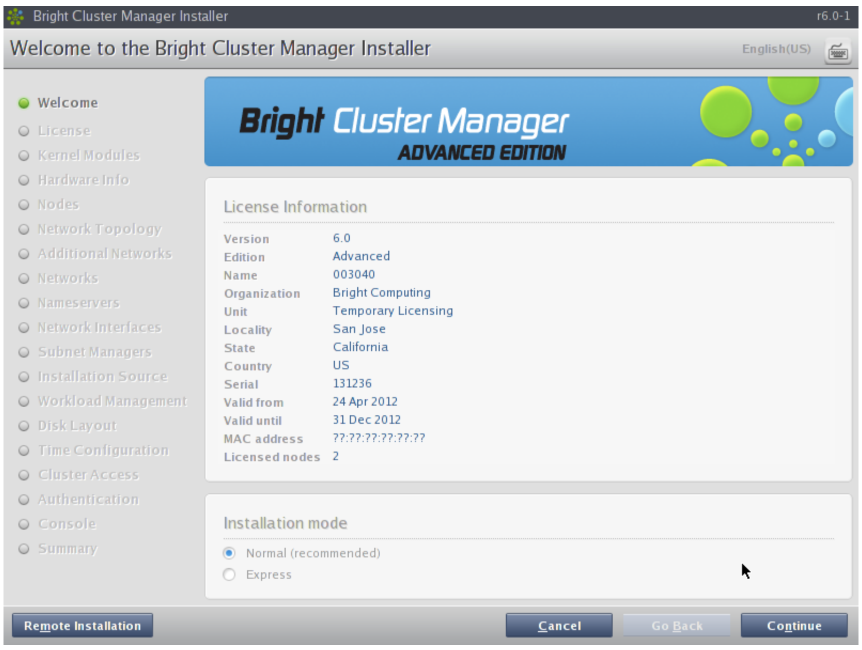
2.3 Head Node Installation: Bare Metal Method

A *bare metal* installation, that is, installing the head node onto a machine with no operating system on it already, is the recommended option because it cannot run into issues from an existing configuration. An operating system from one of the ones listed in section 1.1 is installed during a bare metal installation. The alternative to a bare metal installation is the *add-on* installation of section 2.4.

To start a bare metal installation, the time in the BIOS of the head node is set to local time and the head node is set to boot from DVD. The head node is then booted from the DirectMon DVD.

2.3.1 Welcome Screen

The welcome screen (figure 2.1) displays version and license information. Two installation modes are available, normal mode and express mode. Selecting the express mode installs the head node with the predefined configuration that the DVD was created with. The administrator password automatically set when express mode is selected is: `system`. Clicking on the `Continue` button brings up the DirectMon software license screen, described next.



License Information

Version	6.0
Edition	Advanced
Name	003040
Organization	Bright Computing
Unit	Temporary Licensing
Locality	San Jose
State	California
Country	US
Serial	131236
Valid from	24 Apr 2012
Valid until	31 Dec 2012
MAC address	??:??:??:??:??:??
Licensed nodes	2

Installation mode

[✓ Normal (recommended)]
Express

Figure 2.1: Installation welcome screen for DirectMon

2.3.2 Software License

The "`DDN Software License`" screen (figure 2.2) explains the applicable terms and conditions that apply to use of the DirectMon software.

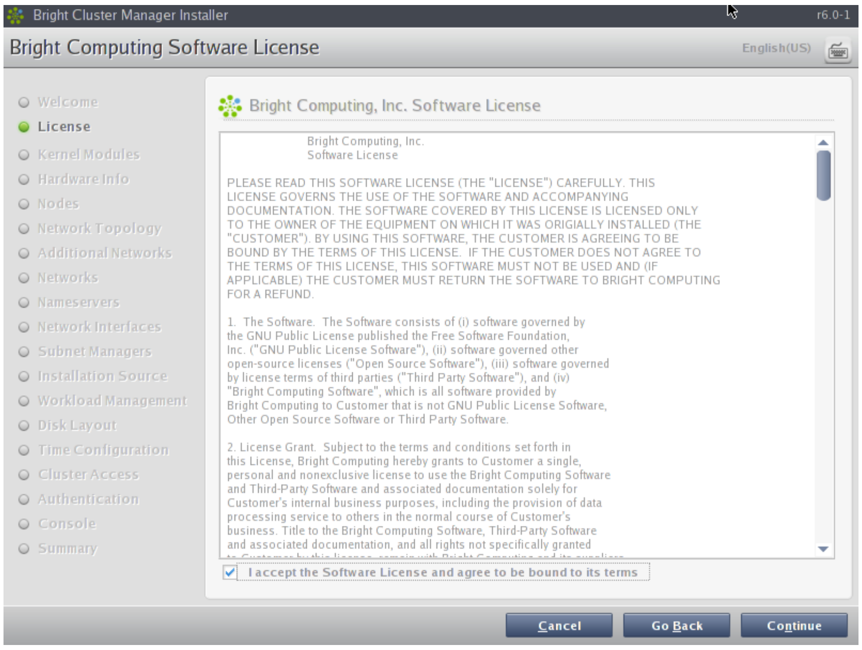
Accepting the terms and conditions, and clicking on the `Continue` button leads to the `Base Distribution EULA` (End User License Agreement) (figure 2.3).

Accepting the terms and conditions of the base distribution EULA, and clicking on the `Continue` button leads to two possibilities.

1. If express mode was selected earlier, then the installer skips ahead

to the `Summary` screen (figure 2.28), where it shows an overview of the predefined installation parameters, and awaits user input to start the install.

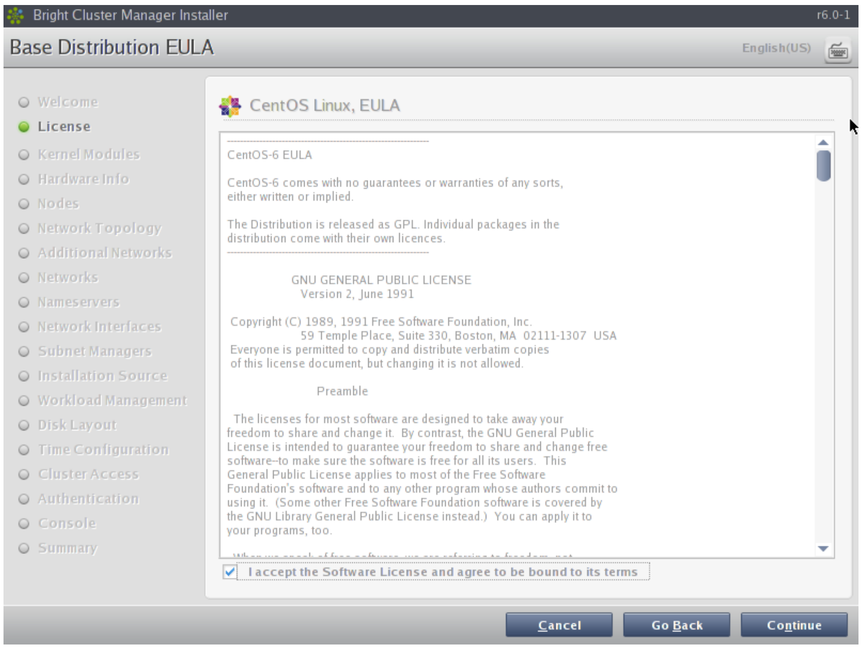
2. Otherwise, if normal installation mode was selected earlier, then the "`Kernel Modules`" configuration screen is displayed, described next.



```
            Bright Computing, Inc.
            Software License

PLEASE READ THIS SOFTWARE LICENSE (THE "LICENSE") CAREFULLY. THIS
LICENSE GOVERNS THE USE OF THE SOFTWARE AND ACCOMPANYING
DOCUMENTATION. THE SOFTWARE COVERED BY THIS LICENSE IS LICENSED ONLY
TO THE OWNER OF THE EQUIPMENT ON WHICH IT WAS ORIGIALLY INSTALLED (THE
"CUSTOMER"). BY USING THIS SOFTWARE, THE CUSTOMER IS AGREEING TO BE
BOUND BY THE TERMS OF THIS LICENSE.  IF THE CUSTOMER DOES NOT AGREE TO
THE TERMS OF THIS LICENSE, THIS SOFTWARE MUST NOT BE USED AND (IF
APPLICABLE) THE CUSTOMER MUST RETURN THE SOFTWARE TO BRIGHT COMPUTING
FOR A REFUND.

1.  The Software.  The Software consists of (i) software governed by
the GNU Public License published the Free Software Foundation,
Inc. ("GNU Public License Software"), (ii) software governed other
open-source licenses ("Open Source Software"), (iii) software governed
by license terms of third parties ("Third Party Software"), and (iv)
"Bright Computing Software", which is all software provided by
Bright Computing to Customer that is not GNU Public License Software,
Other Open Source Software or Third Party Software.

2. License Grant.  Subject to the terms and conditions set forth in
this License, Bright Computing hereby grants to Customer a single,
personal and nonexclusive license to use the Bright Computing Software
and Third-Party Software and associated documentation solely for
Customer's internal business purposes, including the provision of data
processing service to others in the normal course of Customer's
business. Title to the Bright Computing Software, Third-Party Software
and associated documentation, and all rights not specifically granted
```

[✓] I accept the Software License and agree to be bound to its terms

Figure 2.2: DirectMon Software License



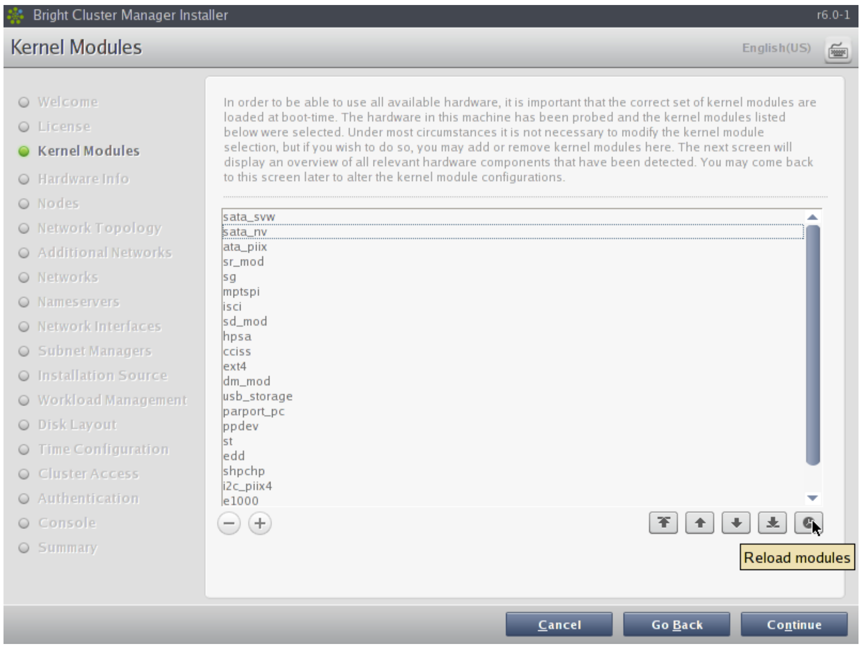
```
-------------------------------------------------
CentOS-6 EULA

CentOS-6 comes with no guarantees or warranties of any sorts,
either written or implied.

The Distribution is released as GPL. Individual packages in the
distribution come with their own licences.
-------------------------------------------------

                GNU GENERAL PUBLIC LICENSE
                   Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
                   59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                        Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.
```

[✓] I accept the Software License and agree to be bound to its terms

Figure 2.3: Base Distribution End User License Agreement

2.3.3 Kernel Modules Configuration

The `Kernel Modules` screen (figure 2.4) shows the kernel modules recommended for loading based on hardware auto-detection.

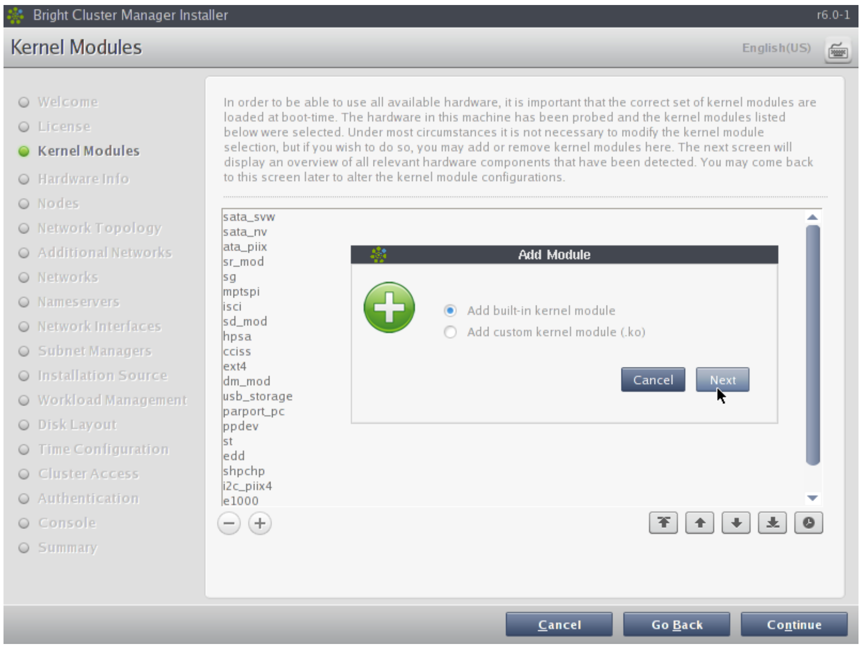


In order to be able to use all available hardware, it is important that the correct set of kernel modules are loaded at boot-time. The hardware in this machine has been probed and the kernel modules listed below were selected. Under most circumstances it is not necessary to modify the kernel module selection, but if you wish to do so, you may add or remove kernel modules here. The next screen will display an overview of all relevant hardware components that have been detected. You may come back to this screen later to alter the kernel module configurations.

```
sata_svw
sata_nv
ata_piix
sr_mod
sg
mptspi
isci
sd_mod
hpsa
cciss
ext4
dm_mod
usb_storage
parport_pc
ppdev
st
edd
shpchp
i2c_piix4
e1000
```

Figure 2.4: Kernel Modules Recommended For Loading After Probing

Clicking the ⊕ button opens an input box for adding a module name and optional module parameters (figure 2.5).



In order to be able to use all available hardware, it is important that the correct set of kernel modules are loaded at boot-time. The hardware in this machine has been probed and the kernel modules listed below were selected. Under most circumstances it is not necessary to modify the kernel module selection, but if you wish to do so, you may add or remove kernel modules here. The next screen will display an overview of all relevant hardware components that have been detected. You may come back to this screen later to alter the kernel module configurations.

```
sata_svw
sata_nv
ata_piix
sr_mod
sg
mptspi
isci
sd_mod
hpsa
cciss
ext4
dm_mod
usb_storage
parport_pc
ppdev
st
edd
shpchp
i2c_piix4
e1000
```

[✓ Add built-in kernel module]
Add custom kernel module (.ko)

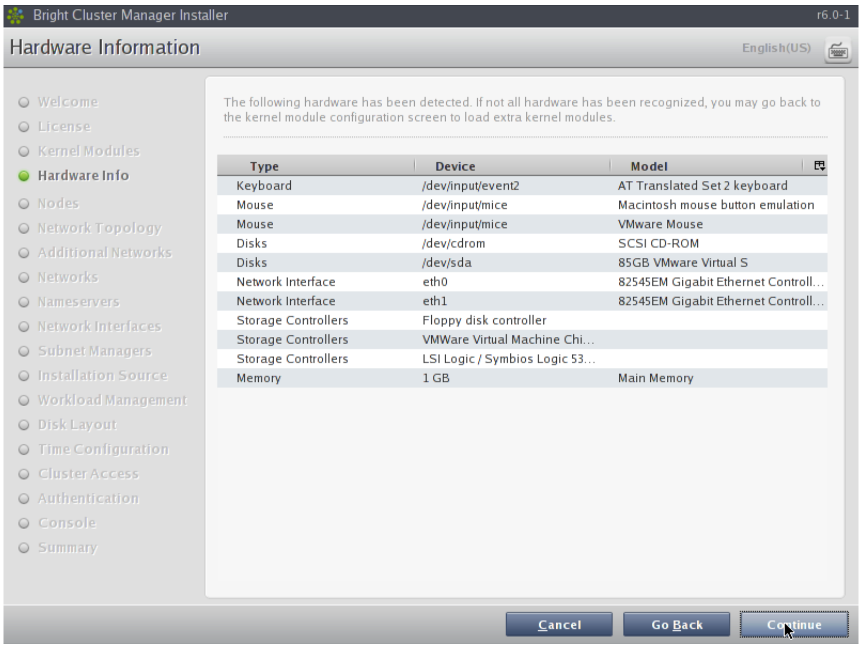
Figure 2.5: Adding Kernel Modules

Similarly, the ⊖ button removes a selected module from the list. The arrow buttons move a kernel module up or down in the list. Kernel module loading order decides the exact name assigned to a device (e.g. `sda`, sdb, eth0, eth1).

After optionally adding or removing kernel modules, clicking `Continue` leads to the “`Hardware Information`” overview screen, described next.

2.3.4 Hardware Overview

The “`Hardware Information`” screen (figure 2.6) provides an overview of detected hardware depending on the kernel modules that have been loaded. If any hardware is not detected at this stage, the “`Go Back`” button is used to go back to the “`Kernel Modules`” screen (figure 2.4) to add the appropriate modules, and then the “`Hardware Information`” screen is returned to, to see if the hardware has been detected. Clicking `Continue` in this screen leads to the `Nodes` configuration screen, described next.



The following hardware has been detected. If not all hardware has been recognized, you may go back to the kernel module configuration screen to load extra kernel modules.

Type	Device	Model
Keyboard	/dev/input/event2	AT Translated Set 2 keyboard
Mouse	/dev/input/mice	Macintosh mouse button emulation
Mouse	/dev/input/mice	VMware Mouse
Disks	/dev/cdrom	SCSI CD-ROM
Disks	/dev/sda	85GB VMware Virtual S
Network Interface	eth0	82545EM Gigabit Ethernet Controll...
Network Interface	eth1	82545EM Gigabit Ethernet Controll...
Storage Controllers	Floppy disk controller	
Storage Controllers	VMWare Virtual Machine Chi...	
Storage Controllers	LSI Logic / Symbios Logic 53...	
Memory	1 GB	Main Memory

Figure 2.6: Hardware Overview Based On Loaded Kernel Modules

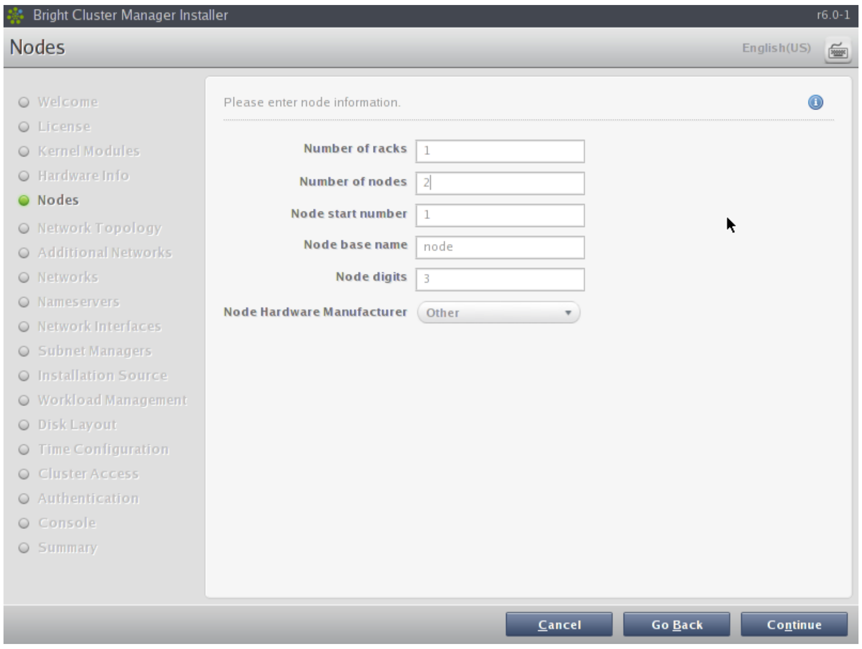
2.3.5 Nodes Configuration

The `Nodes` screen (figure 2.7) configures the number of racks, the number of regular nodes, the node basename, the number of digits for nodes, and the hardware manufacturer.

The maximum number of digits is 5, to keep the hostname reasonably readable.

The “`Node Hardware Manufacturer`” selection option initializes any monitoring parameters relevant for that manufacturer’s hardware. If the manufacturer is not known, then `Other` is selected from the list.

Clicking `Continue` in this screen leads to the “`Network Topology`” selection screen, described next.



Please enter node information.

Number of racks: 1
Number of nodes: 2
Node start number: 1
Node base name: node
Node digits: 3
Node Hardware Manufacturer: Other

Figure 2.7: Nodes Configuration

2.3.6 Network Topology

Regular nodes are always located on an internal network, by default called `Internalnet`.

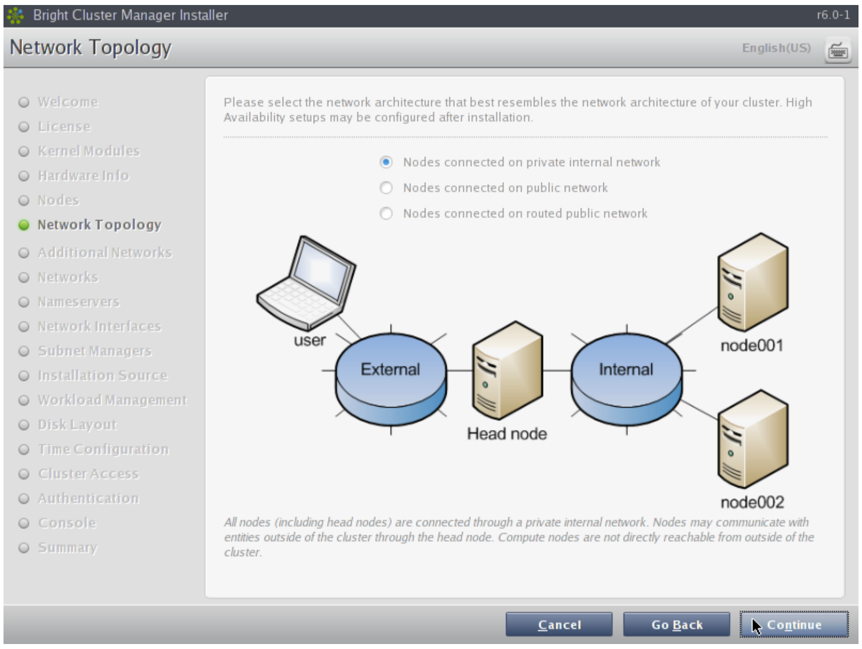
The “`Network Topology`” screen allows selection of one of three different network topologies.

A *type 1* network (figure 2.8), with nodes connected on a private internal network. This is the default network setup. In this topology, a network packet from a head or regular node destined for any external network that the cluster is attached to, by default called `Externalnet`, can only reach the external network by being routed and forwarded at the head node itself. The packet routing for `Externalnet` is configured at the head node.

A *type 2* network (figure 2.9) has its nodes connected via a router to a public network. In this topology, a network packet from a regular node destined for outside the cluster does not go via the head node, but uses the router to reach a public network. Packets destined for the head node however still go directly to the head node. Any routing for beyond the router is configured on the router, and not on the cluster or its parts. Care should be taken to avoid DHCP conflicts between the DHCP server on the head node and any existing DHCP server on the internal network if the cluster is being placed within an existing corporate network that is also part of `Internalnet` (there is no `Externalnet` in this topology). Typically, in the case where the cluster becomes part of an existing network, there is another router configured and placed between the regular corporate machines and the cluster nodes to shield them from effects on each other.

A *type 3* network (figure 2.10), with nodes connected on a routed public network. In this topology, a network packet from a regular node, destined for another network, uses a router to get to it. The head node, being on another network, can only be reached via a router too. The network the regular nodes are on is called `Internalnet` by default, and the network the head node is on is called `Managementnet` by default. Any routing configuration for beyond the routers that are attached to the `Internalnet` and `Managementnet` networks is configured on the routers, and not on the clusters or its parts.

Selecting the network topology helps decide the predefined networks on the `Networks` settings screen later (figure 2.15). Clicking `Continue` here leads to the "`Additional Network Configuration`" screen, described next.

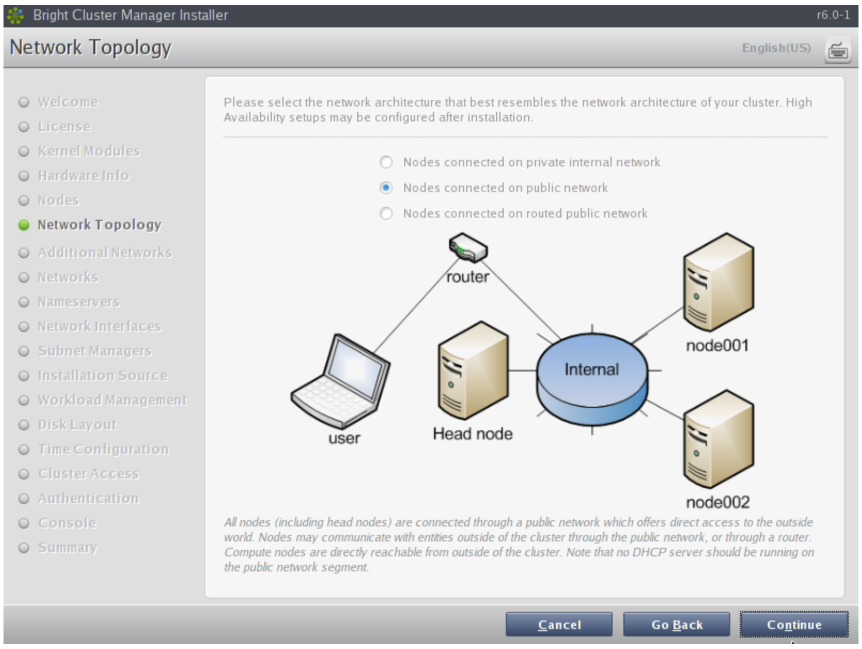


Please select the network architecture that best resembles the network architecture of your cluster. High Availability setups may be configured after installation.

[✓ Nodes connected on private internal network]
Nodes connected on public network
Nodes connected on routed public network

All nodes (including head nodes) are connected through a private internal network. Nodes may communicate with entities outside of the cluster through the head node. Compute nodes are not directly reachable from outside of the cluster.

Figure 2.8: Networks Topology: nodes connected on a private internal network



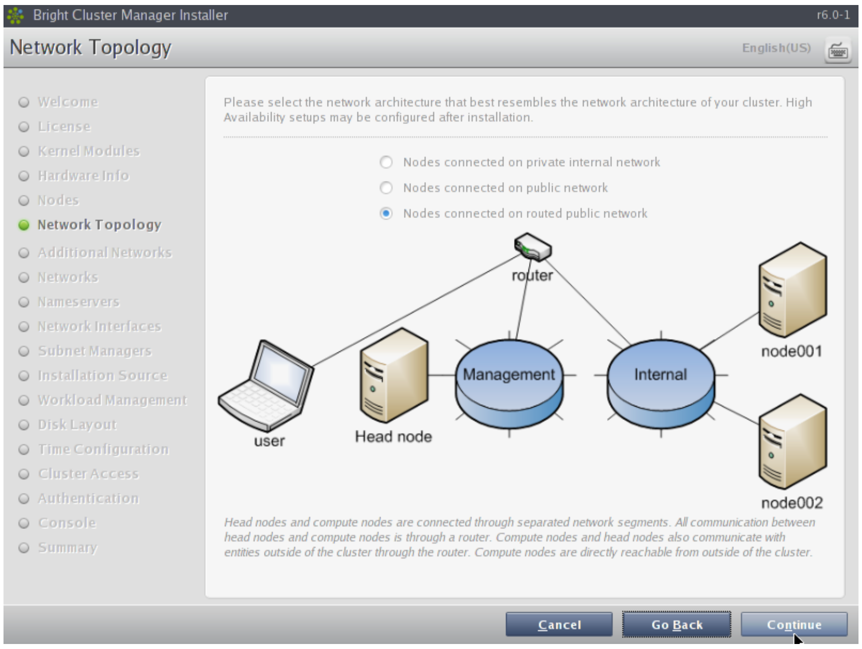
Network Topology

Please select the network architecture that best resembles the network architecture of your cluster. High Availability setups may be configured after installation.

- ○ Nodes connected on private internal network
- ● Nodes connected on public network
- ○ Nodes connected on routed public network

All nodes (including head nodes) are connected through a public network which offers direct access to the outside world. Nodes may communicate with entities outside of the cluster through the public network, or through a router. Compute nodes are directly reachable from outside of the cluster. Note that no DHCP server should be running on the public network segment.

Figure 2.9: Networks Topology: nodes connected via router to a public network



Network Topology

Please select the network architecture that best resembles the network architecture of your cluster. High Availability setups may be configured after installation.

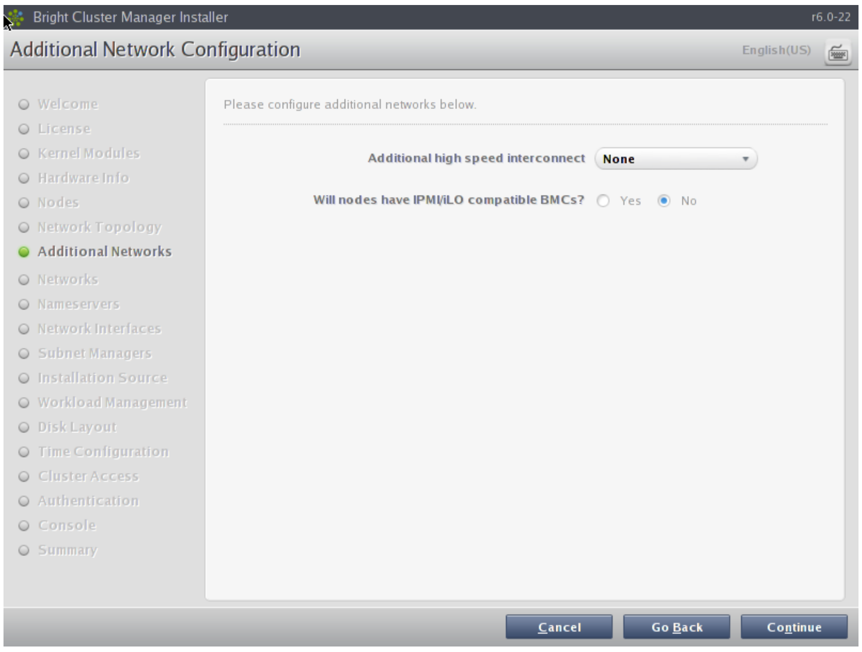
- ○ Nodes connected on private internal network
- ○ Nodes connected on public network
- ● Nodes connected on routed public network

Head nodes and compute nodes are connected through separated network segments. All communication between head nodes and compute nodes is through a router. Compute nodes and head nodes also communicate with entities outside of the cluster through the router. Compute nodes are directly reachable from outside of the cluster.

Figure 2.10: Network Topology: nodes connected on a routed public network

2.3.7 Additional Network Configuration

The "`Additional Network Configuration`" screen allows the configuration of high speed interconnect networks, and also of BMC networks (figure 2.11).



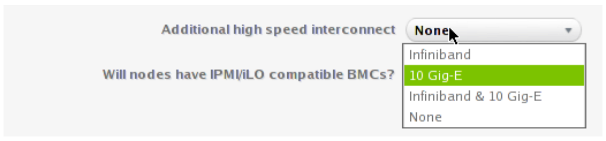
Additional Network Configuration

Please configure additional networks below.

Additional high speed interconnect: None

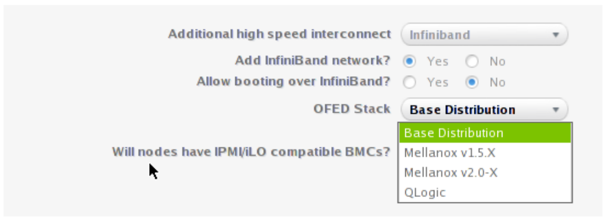
Will nodes have IPMI/iLO compatible BMCs? Yes [✓ No]

Figure 2.11: Additional Network Configuration: OFED networking


Additional high speed interconnect: None

Will nodes have IPMI/iLO compatible BMCs?

Figure 2.12: Additional Network Configuration: Interconnect Interface


Additional high speed interconnect: Infiniband

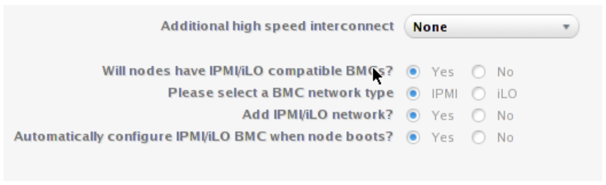
Add InfiniBand network? [✓ Yes] No

Allow booting over InfiniBand? Yes [✓ No]

OFED Stack: Base Distribution

Will nodes have IPMI/iLO compatible BMCs?

Figure 2.13: Additional Network Configuration: OFED stack


Additional high speed interconnect: None

Will nodes have IPMI/iLO compatible BMCs? [✓ Yes] No

Please select a BMC network type [✓ IPMI] iLO

Add IPMI/iLO network? [✓ Yes] No

Automatically configure IPMI/iLO BMC when node boots? [✓ Yes] No

Figure 2.14: Additional Network Configuration: BMC type

- The interconnect selector options are to configure the compute nodes so that they communicate quickly with each other while running computational workload jobs.

  The choices include 10 Gig-E and InfiniBand RDMA OFED (figure 2.12). The regular nodes of a cluster can be set to boot over the chosen option in both cases.

In the case of InfiniBand, there are OFED stack driver options (figure 2.13). The OFED stack used can be the parent distribution packaged stack, or it can be the appropriate (Mellanox 1.5, Mellanox 2.0, or QLogic) InfiniBand hardware vendor stack. Currently, choosing the parent distribution stack is recommended because it tends to be integrated better with the OS. OFED installation is discussed further in section 13.6.

- The BMC (Baseboard Management Controller) selector options configure the BMC network for the regular nodes (figure 2.14). The appropriate BMC network type (IPMI or iLO) should be chosen if a BMC is to be used. The remaining options—adding the network, and automatically configuring the network—can then be set. The BMC configuration is discussed further in section 4.8.

  If a BMC is to be used, the BMC password is set to a random value. Retrieving and changing a BMC password is covered in section 4.8.2.

Clicking `Continue` in figure 2.11 leads to the `Networks` configuration screen, described next.

2.3.8 Networks Configuration

The `Networks` configuration screen (figure 2.15) displays the predefined list of networks, based on the selected network topology. IPMI and InfiniBand networks are defined based on selections made in the "`Additional Network Configuration`" screen earlier (figure 2.11).

The parameters of the network interfaces can be configured in this screen.

For a *type 1* setup, an external network and an internal network are always defined.

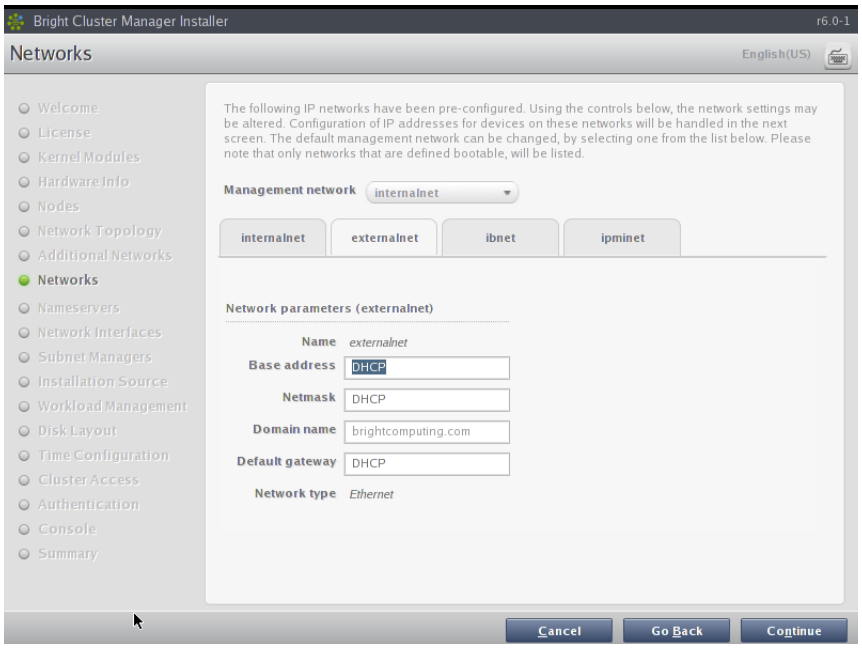
For a *type 2* setup only an internal network is defined and no external network is defined.

For a *type 3* setup, an internal network and a management network are defined.

A pop-up screen is used to help fill these values in for a type 1 network. The values can be provided via DHCP, but usually static values are used in production systems to avoid confusion. The pop-up screen asks for IP address details for the external network, where the network `externalnet` corresponds to the site network that the cluster resides in (e.g. a corporate or campus network). The IP address details are therefore the details of the head node for a type 1 externalnet network (figure 2.8).

Clicking `Continue` in this screen validates all network settings. Invalid settings for any of the defined networks cause an alert to be displayed, explaining the error. A correction is then needed to proceed further.

If all settings are valid, the installation proceeds on to the `Nameservers` screen, described in the next section.


Bright Cluster Manager Installer r6.0-1

Networks

The following IP networks have been pre-configured. Using the controls below, the network settings may be altered. Configuration of IP addresses for devices on these networks will be handled in the next screen. The default management network can be changed, by selecting one from the list below. Please note that only networks that are defined bootable, will be listed.

Management network: internalnet

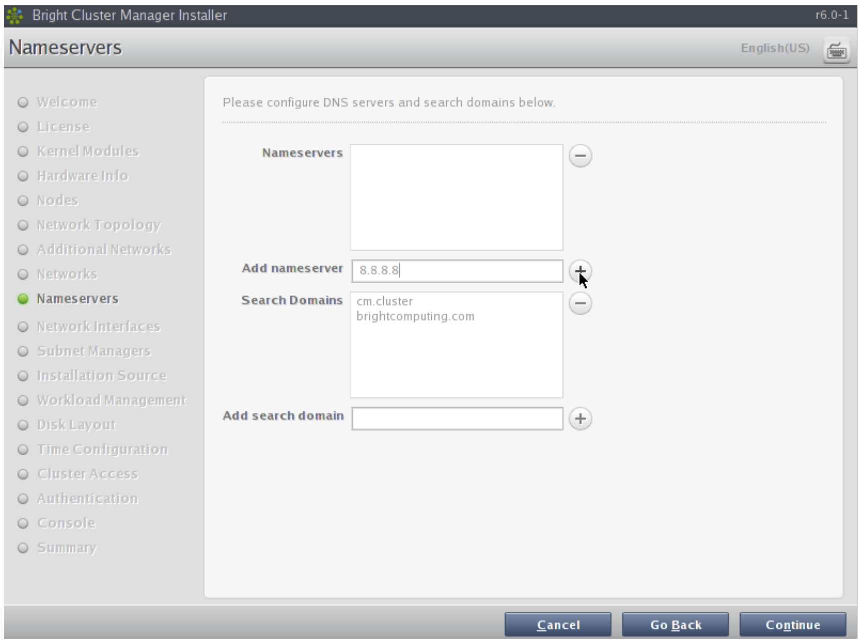
Network parameters (externalnet)

Name	externalnet
Base address	DHCP
Netmask	DHCP
Domain name	brightcomputing.com
Default gateway	DHCP
Network type	Ethernet

Figure 2.15: Networks Configuration

2.3.9 Nameservers And Search Domains

Search domains and external name servers can be added or removed using the `Nameservers` screen (figure 2.16). Using an external name server is recommended. Clicking on `Continue` leads to the "`Network Interfaces`" configuration screen, described next.


Bright Cluster Manager Installer r6.0-1

Nameservers

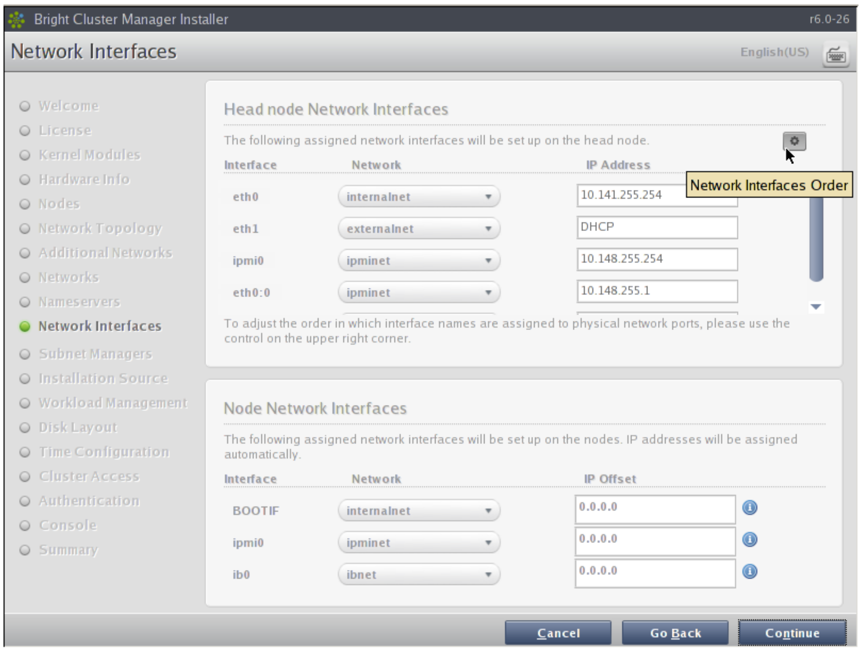
Please configure DNS servers and search domains below.

Nameservers	
Add nameserver	8.8.8.8
Search Domains	cm.cluster brightcomputing.com
Add search domain	

Figure 2.16: Nameservers and search domains

2.3.10 Network Interfaces Configuration

The "`Network Interfaces`" screen (figure 2.17) allows a review of the list of network interfaces with their proposed settings. The head node and regular nodes each have a settings pane for their network configurations.



Head node Network Interfaces

The following assigned network interfaces will be set up on the head node.

Interface	Network	IP Address
eth0	internalnet	10.141.255.254
eth1	externalnet	DHCP
ipmi0	ipminet	10.148.255.254
eth0:0	ipminet	10.148.255.1

To adjust the order in which interface names are assigned to physical network ports, please use the control on the upper right corner.

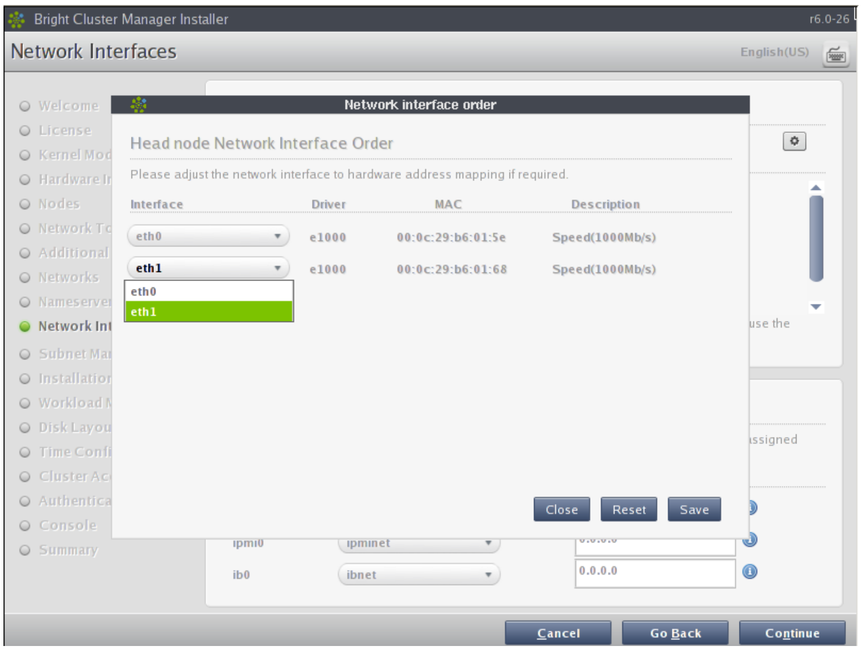
Node Network Interfaces

The following assigned network interfaces will be set up on the nodes. IP addresses will be assigned automatically.

Interface	Network	IP Offset
BOOTIF	internalnet	0.0.0.0
ipmi0	ipminet	0.0.0.0
ib0	ibnet	0.0.0.0

Figure 2.17: Network Interface Configuration

An icon in the `Head Node Interfaces` section, where the hover-text is showing in the figure, allows the Ethernet network interface order to be changed on the head node. For example, if the interfaces with the names eth0 and eth1 need to be swapped around, clicking on the icon brings up a screen allowing the names to be associated with specific MAC addresses (figure 2.18).



Head node Network Interface Order

Please adjust the network interface to hardware address mapping if required.

Interface	Driver	MAC	Description
eth0	e1000	00:0c:29:b6:01:5e	Speed(1000Mb/s)
eth1	e1000	00:0c:29:b6:01:68	Speed(1000Mb/s)

Figure 2.18: Network Interface Configuration Order Changing

For node network interfaces, the *IP offset* can be modified. [1]

A different network can be selected for each interface using the dropdown box in the `Network` column. Selecting `Unassigned` disables a network interface.

If the corresponding network settings are changed (e.g., base address of the network) the IP address of the head node interface needs to be modified accordingly. If IP address settings are invalid, an alert is displayed, explaining the error.

Clicking `Continue` on a "`Network Interfaces`" screen validates IP address settings for all node interfaces.

If all setting are correct, and if InfiniBand networks have been defined, clicking on `Continue` leads to the "`Subnet Managers`" screen (figure 2.19), described in the next section.

If no InfiniBand networks are defined, or if InfiniBand networks have not been enabled on the networks settings screen, then clicking `Continue` instead leads to the `CD/DVD ROMs` selection screen (figure 2.20).

2.3.11 Select Subnet Managers

The "`Subnet Managers`" screen in figure 2.19 is only displayed if an InfiniBand network was defined, and lists all the nodes that can run the InfiniBand subnet manager. The nodes assigned the role of a subnet manager are ticked, and the `Continue` button is clicked to go on to the "`CD/DVD ROMs`" selection screen, described next.

[1] The IP offset is used to calculate the IP address assigned to a regular node interface. The nodes are conveniently numbered in a sequence, so their interfaces are typically also given a network IP address that is in a sequence on a selected network. In DirectMon, interfaces by default have their IP addresses assigned to them sequentially, in steps of 1, starting after the network base address.

The default IP offset is 0.0.0.0, which means that the node interfaces by default start their range at the usual default values in their network.

With a modified IP offset, the point at which addressing starts is altered. For example, a different offset might be desirable when no IPMI network has been defined, but the nodes of the cluster do have IPMI interfaces in addition to the regular network interfaces. If a modified IP offset is not set for one of the interfaces, then the `BOOTIF` and `ipmi0` interfaces get IP addresses assigned on the same network by default, which could be confusing.

However, if an offset is entered for the `ipmi0` interface, then the assigned IPMI IP addresses start from the IP address specified by the offset. That is, each modified IPMI address takes the value:

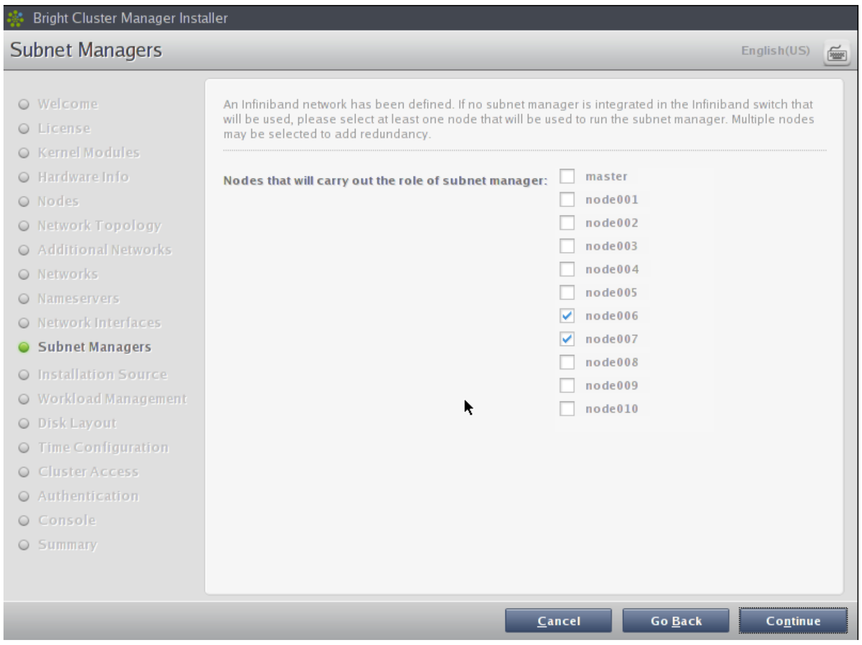
```
address that would be assigned by default + IP offset
```

Example

Taking the case where BOOTIF and IPMI interfaces would have IP addresses on the same network with the default IP offset:

Then, on a cluster of 10 nodes, a modified IPMI IP offset of 0.0.0.20 means:

- the BOOTIF interfaces stay on 10.141.0.1,...,10.141.0.10 while
- the IPMI interfaces range from 10.141.0.21,...,10.141.0.30



An Infiniband network has been defined. If no subnet manager is integrated in the Infiniband switch that will be used, please select at least one node that will be used to run the subnet manager. Multiple nodes may be selected to add redundancy.

Nodes that will carry out the role of subnet manager:
☐ master
☐ node001
☐ node002
☐ node003
☐ node004
☐ node005
☑ node006
☑ node007
☐ node008
☐ node009
☐ node010

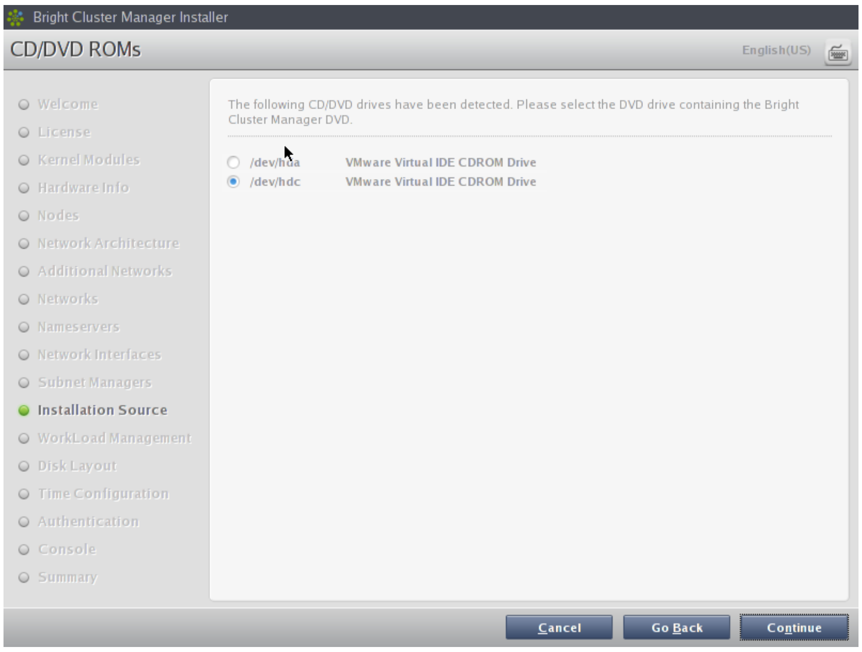
Figure 2.19: Subnet Manager Nodes

2.3.12 Select CD/DVD ROM

The “`CD/DVD ROMs`” screen in figure 2.20 lists all detected CD/DVD-ROM devices. If multiple drives are found, then the drive with the DirectMon DVD needs to be selected by the administrator. If the installation source is not detected, it can be added manually.

Optionally, a media integrity check can be set.

Clicking on the `Continue` button starts the media integrity check, if it was ordered. The media integrity check can take about a minute to run. If all is well, then the “`Workload Management`” setup screen is displayed, as described next.



The following CD/DVD drives have been detected. Please select the DVD drive containing the Bright Cluster Manager DVD.

○	/dev/hda	VMware Virtual IDE CDROM Drive
●	/dev/hdc	VMware Virtual IDE CDROM Drive

Figure 2.20: DVD Selection

2.3.13 Workload Management Configuration

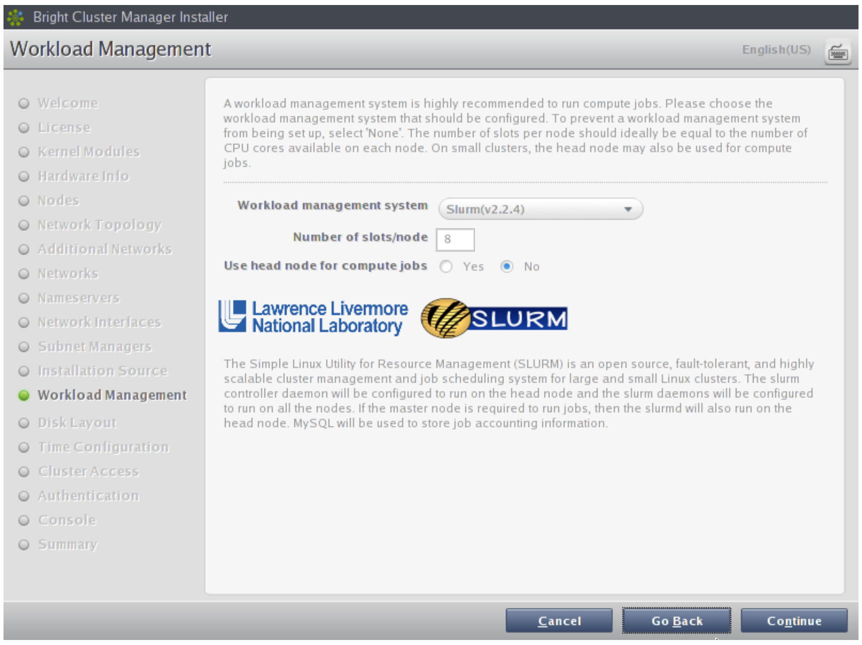
The "`Workload Management`" configuration screen (figure 2.21) allows selection from a list of supported workload managers. A workload management system is highly recommended to run multiple compute jobs on a cluster.

The Maui and Moab scheduler can be configured to run on the cluster if selected. However, these are not installed by the DirectMon installer because Adaptive Computing prefers to distribute them directly. Details on installing the packages after the cluster has been installed are given in Chapter 9 on workload management.

To prevent a workload management system from being set up, select `None`. If a workload management system is selected, then the number of slots per node can be set, otherwise the slots setting is ignored. If no changes are made, then the number of slots defaults to the CPU count on the head node.

The head node can also be selected for use as a compute node, which can be a sensible choice on small clusters. The setting is ignored if no workload management system is selected.

Clicking `Continue` on this screen leads to the "`Disk Partitioning and Layouts`" screen, described next.



Workload Management

A workload management system is highly recommended to run compute jobs. Please choose the workload management system that should be configured. To prevent a workload management system from being set up, select 'None'. The number of slots per node should ideally be equal to the number of CPU cores available on each node. On small clusters, the head node may also be used for compute jobs.

Workload management system: Slurm(v2.2.4)

Number of slots/node: 8

Use head node for compute jobs: Yes [✓ No]

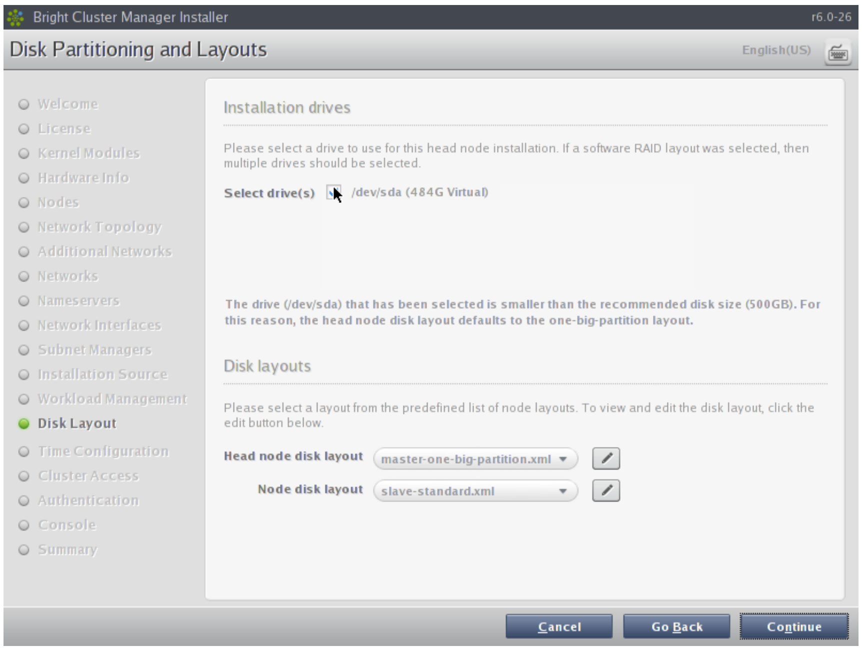
The Simple Linux Utility for Resource Management (SLURM) is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. The slurm controller daemon will be configured to run on the head node and the slurm daemons will be configured to run on all the nodes. If the master node is required to run jobs, then the slurmd will also run on the head node. MySQL will be used to store job accounting information.

Figure 2.21: Workload Management Setup

2.3.14 Disk Partitioning And Layouts

The partitioning layout XML schema is described in detail in Appendix D.

The "`Disk Partitioning and Layouts`" configuration screen (figure 2.22):



Disk Partitioning and Layouts

Installation drives

Please select a drive to use for this head node installation. If a software RAID layout was selected, then multiple drives should be selected.

Select drive(s): /dev/sda (484G Virtual)

The drive (/dev/sda) that has been selected is smaller than the recommended disk size (500GB). For this reason, the head node disk layout defaults to the one-big-partition layout.

Disk layouts

Please select a layout from the predefined list of node layouts. To view and edit the disk layout, click the edit button below.

Head node disk layout: master-one-big-partition.xml

Node disk layout: slave-standard.xml

Figure 2.22: Disk Partitioning And Layouts

- selects the drive that the cluster manager is installed onto on the head node.
- sets the disk partitioning layout for the head node and regular nodes with the two options: "`Head node disk layout`" and

"`Node disk layout`".

- - * The head node by default uses
      - one big partition if it has a drive size smaller than about 500GB
      - several partitions if it has a drive size greater than or equal to about 500GB.
    * The regular node by default uses several partitions.

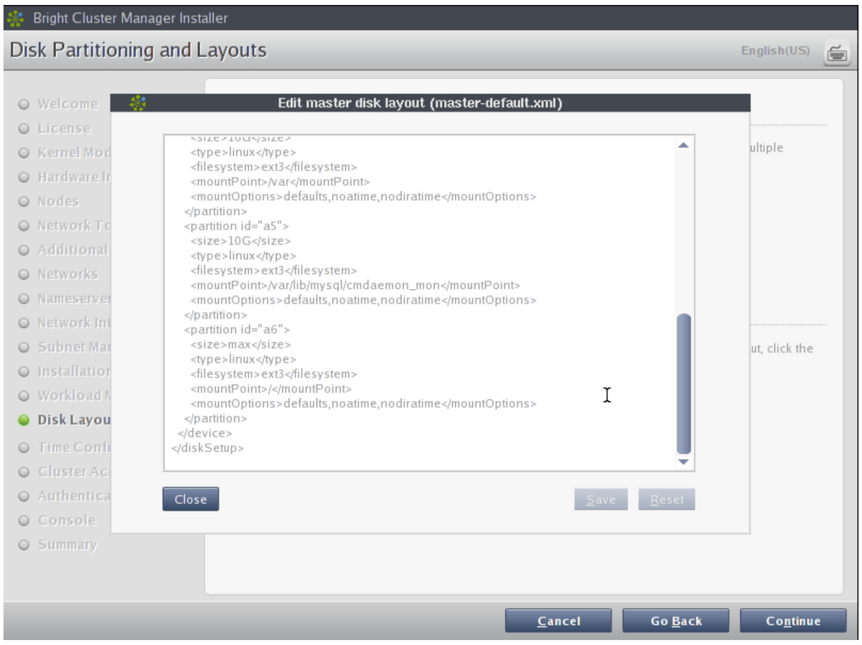
  A partitioning layout other than the default can be selected for installation from the drop-down boxes for the head node and regular nodes.
- A partitioning layout is the only installation setting that cannot easily be changed after the completion (section 2.3.20) of installation. It should therefore be decided upon with care.
- A text editor pops up when the edit button of a partitioning layout is clicked (figure 2.23). This allows the administrator to view and change layout values within the layout's configuration XML file using the schema in Appendix D.1.

  The `Save` and `Reset` buttons are enabled on editing, and save or undo the text editor changes. Once saved, the changes cannot be reverted automatically in the text editor, but must be done manually.

The XML schema allows the definition of a great variety of layouts in the layout's configuration XML file. For example:

1. for a large cluster or for a cluster that is generating a lot of monitoring or burn data, the default partition layout partition size for `/var` may fill up with log messages because log messages are usually stored under `/var/log/`. If `/var` is in a partition of its own, as in the default partitioning layout presented when the hard drive is about 500GB or more, then providing a larger size of partition than the default for `/var` allows more logging to take place before `/var` is full. Modifying the value found within the `<size></size>` tags associated with that partition in the XML file (figure 2.23) modifies the size of the partition that is to be installed.

2. the administrator could specify the layout for multiple non-RAID drives on the head node using one `<blockdev></blockdev>` tag pair within an enclosing `<device></device>` tag pair for each drive.



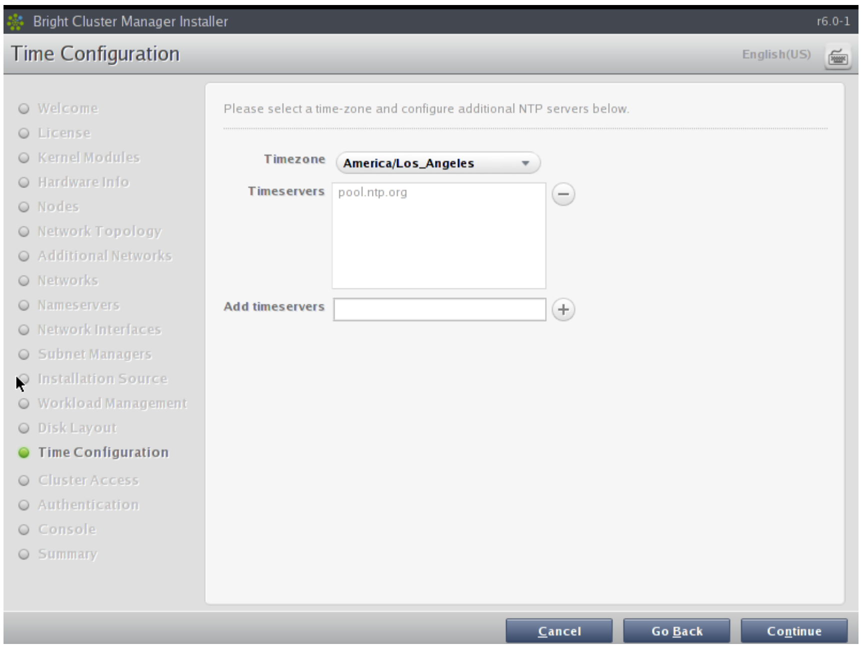
```
    <size>10G</size>
    <type>linux</type>
    <filesystem>ext3</filesystem>
    <mountPoint>/var</mountPoint>
    <mountOptions>defaults,noatime,nodiratime</mountOptions>
  </partition>
  <partition id="a5">
    <size>10G</size>
    <type>linux</type>
    <filesystem>ext3</filesystem>
    <mountPoint>/var/lib/mysql/cmdaemon_mon</mountPoint>
    <mountOptions>defaults,noatime,nodiratime</mountOptions>
  </partition>
  <partition id="a6">
    <size>max</size>
    <type>linux</type>
    <filesystem>ext3</filesystem>
    <mountPoint>/</mountPoint>
    <mountOptions>defaults,noatime,nodiratime</mountOptions>
  </partition>
 </device>
</diskSetup>
```

Figure 2.23: Edit Head Node Disk Partitioning

Clicking `Continue` on the “`Disk Partitioning and Layouts`” screen leads to the “`Time Configuration`” screen, described next.

2.3.15 Time Configuration

The “`Time Configuration`” screen (figure 2.24) displays a predefined list of time servers.



Please select a time-zone and configure additional NTP servers below.

Timezone: America/Los_Angeles

Timeservers: pool.ntp.org

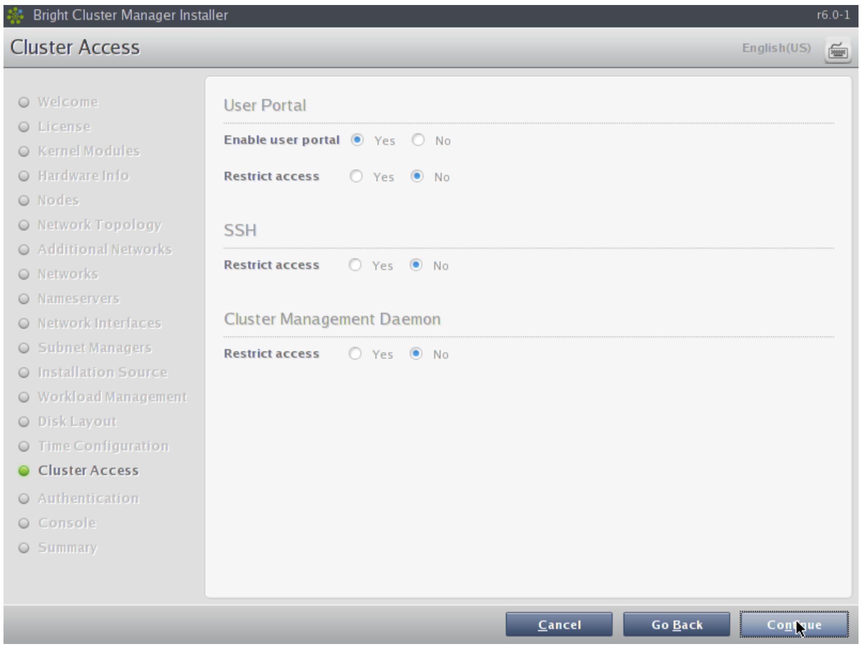
Add timeservers:

Figure 2.24: Time Configuration

Timeservers can be removed by selecting a time server from the list and clicking the ⊖ button. Additional time servers can be added by entering the name of the time server and clicking the ⊕ button. A timezone can be selected from the drop-down box if the default is incorrect. Clicking `Continue` leads to the “`Cluster Access`” screen, described next.

2.3.16 Cluster Access

The “`Cluster Access`” screen (figure 2.25) sets the existence of a cluster management web portal service, and also sets network access to several services.



User Portal

Enable user portal [✓ Yes] No

Restrict access Yes [✓ No]

SSH

Restrict access Yes [✓ No]

Cluster Management Daemon

Restrict access Yes [✓ No]

Figure 2.25: Cluster Access

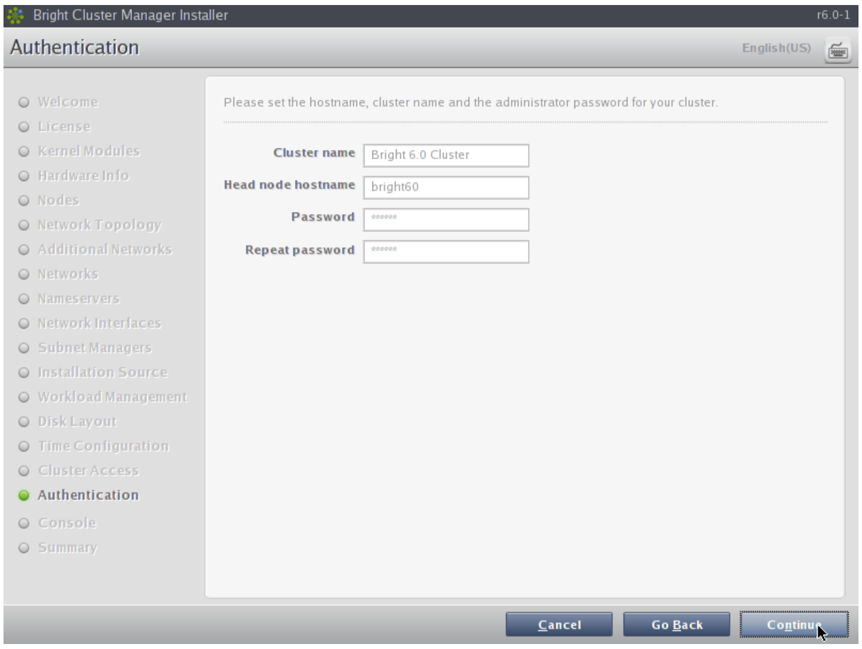
These services are the web portal, ssh, and the cluster management daemon.

If restricting network access for a service is chosen, then an editable list of networks that may access the service is displayed. By default the list has no members. The screen will not move on to the next screen until the list contains at least one CIDR-format network IP address.

If the conditions for this screen are satisfied, then clicking `Continue` leads to the `Authentication` screen, described next.

2.3.17 Authentication

The `Authentication` screen (figure 2.26) requires the password to be set twice for the cluster administrator. The cluster name and the head node hostname can also be set in this screen. Clicking `Continue` validates the passwords that have been entered, and if successful, leads to the `Console` screen, described next.



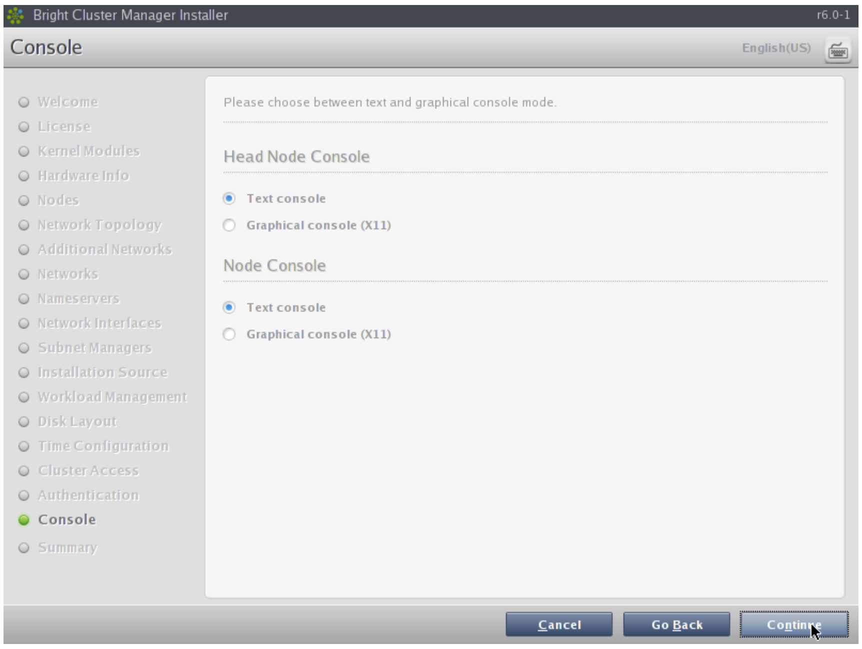
Please set the hostname, cluster name and the administrator password for your cluster.

Cluster name	Bright 6.0 Cluster
Head node hostname	bright60
Password	******
Repeat password	******

Figure 2.26: Authentication

2.3.18 Console

The `Console` screen (figure 2.27) allows selection of a graphical mode or a text console mode for when the head node or regular nodes boot. Clicking `Continue` leads to the `Summary` screen, described next.



Please choose between text and graphical console mode.

Head Node Console

[✓ Text console] Graphical console (X11)

Node Console

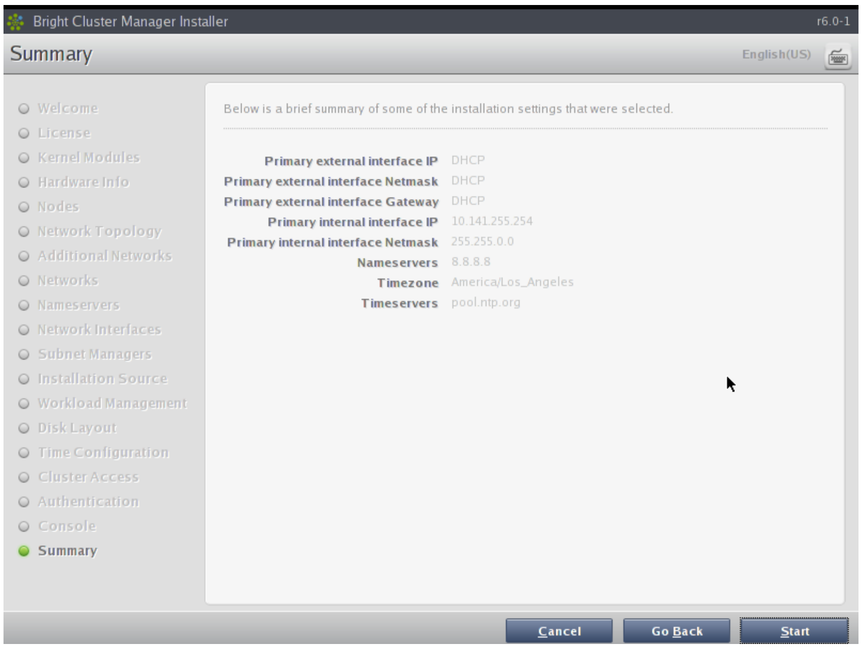
[✓ Text console] Graphical console (X11)

Figure 2.27: Console

2.3.19 Summary

The `Summary` screen (figure 2.28), summarizes some of the installation settings and parameters configured during the previous stages. If the express mode installation was chosen, then it summarizes the predefined settings and parameters. Changes to the values on this screen are made by navigating to previous screens and correcting the values there.

When the summary screen displays the right values, clicking on the `Start` button leads to the "`Installation Progress`" screen, described next.



Below is a brief summary of some of the installation settings that were selected.

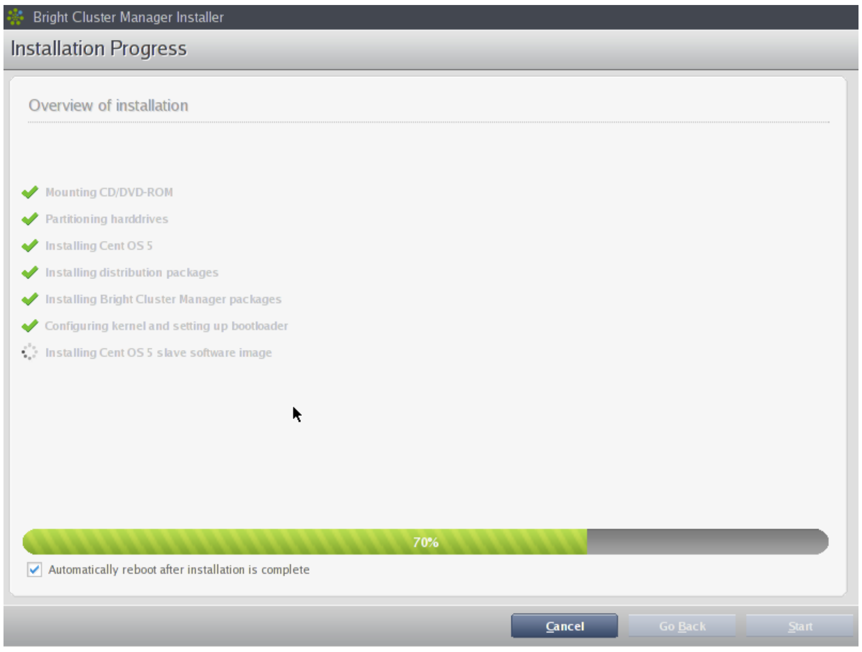
Primary external interface IP	DHCP
Primary external interface Netmask	DHCP
Primary external interface Gateway	DHCP
Primary internal interface IP	10.141.255.254
Primary internal interface Netmask	255.255.0.0
Nameservers	8.8.8.8
Timezone	America/Los_Angeles
Timeservers	pool.ntp.org

Figure 2.28: Summary of Installation Settings

2.3.20 Installation

The "`Installation Progress`" screen (figure 2.29) shows the progress of the installation. It is not possible to navigate back to previous screens once the installation has begun. When the installation is complete (figure 2.30), the installation log can be viewed in detail by clicking on "`Install Log`".

The `Reboot` button restarts the machine. The BIOS boot order may need changing or the DVD should be removed, in order to boot from the hard drive on which DirectMon has been installed.


Bright Cluster Manager Installer

Installation Progress

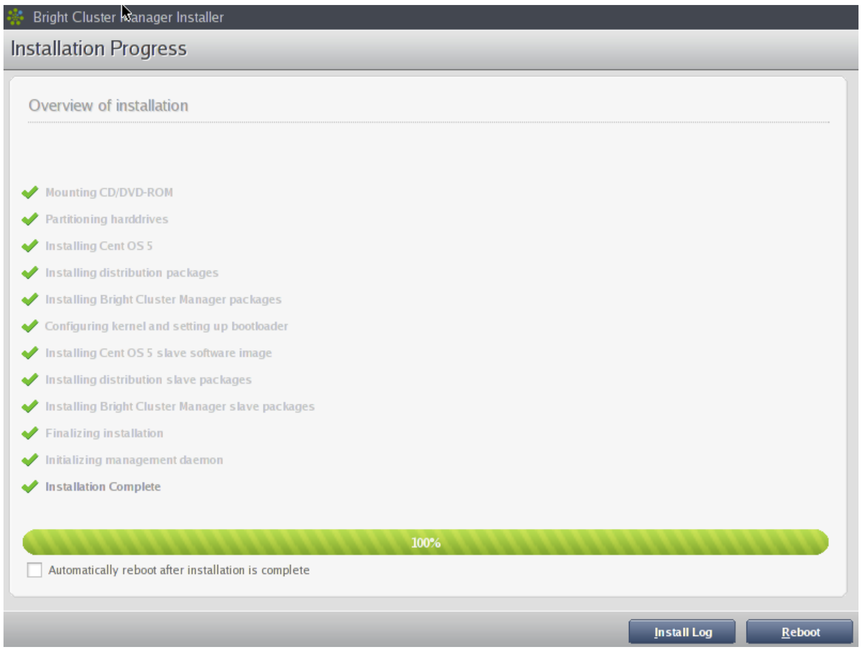
Overview of installation

✓ Mounting CD/DVD-ROM
✓ Partitioning harddrives
✓ Installing Cent OS 5
✓ Installing distribution packages
✓ Installing Bright Cluster Manager packages
✓ Configuring kernel and setting up bootloader
Installing Cent OS 5 slave software image

70%

[✓] Automatically reboot after installation is complete

Figure 2.29: Installation Progress


Bright Cluster Manager Installer

Installation Progress

Overview of installation

✓ Mounting CD/DVD-ROM
✓ Partitioning harddrives
✓ Installing Cent OS 5
✓ Installing distribution packages
✓ Installing Bright Cluster Manager packages
✓ Configuring kernel and setting up bootloader
✓ Installing Cent OS 5 slave software image
✓ Installing distribution slave packages
✓ Installing Bright Cluster Manager slave packages
✓ Finalizing installation
✓ Initializing management daemon
✓ Installation Complete

100%

[ ] Automatically reboot after installation is complete

Figure 2.30: Installation Completed

After rebooting, the system starts and presents a login prompt. After logging in as `root` using the password that was set during the installation procedure, the system is ready to be configured. If express installation mode was chosen earlier as the install method, then the password is preset to `system`.

At this stage, if the parent distribution is Red Hat or SUSE, an administrator typically registers the head node (Appendix M) in order to allow the cluster to track updates.

The administrator with no interest in the add-on method of installation can skip on to Chapter 3, where some of the tools and concepts that play a central role in DirectMon are introduced. Chapter 4 then explains how to configure and further set up the cluster.

2.4 Head Node Installation: Add-On Method

An *add-on* installation, in contrast to the bare metal installation (section 2.3), is an installation that is done onto a machine that is already running one of the supported distributions of section 1.1. The installation of the distribution can therefore be skipped for this case. However, unlike the bare metal installation, the add-on is not recommended for inexperienced cluster administrators. This is because after an add-on installation has been done to the head node, a software image for the regular nodes must still be installed into a directory on the head node. The software image is what is provisioned to regular nodes when they are powered up. The creation and installation of a software image requires some understanding of the Linux operating system, and is described in section 10.6.

2.4.1 Prerequisites

For the add-on method

- The operating system must obviously follow system administration best practices so that it works properly with the official distribution, when DirectMon is added on
- The items of software that DirectMon adds must be allowed to overrule in any conflict with what is already installed, or the end result of the installation cannot be supported.
- A product key is needed
- There must be repository access to the supported distribution.
  - Internet access makes up-to-date repository access possible. RHEL and SLES repository access requires a subscription from Red Hat or SUSE (Appendix M).
  - For high-security environments without internet access, an alternative is to mount a DVD device or ISO image to the head node containing a local repository snapshot of the parent distribution, and specify the repository location when running the installation command. Configuring a local repository is described in section 10.6.3.

2.4.2 Installing The Installer

To carry out an add-on installation, the `ddn-installer-` package must be installed with a package installer.

The `ddn-installer-` package can be obtained from a DirectMon installation DVD, in the directory `/addon/`. The file name is something like `ddn-installer--129_cmddn.noarch.rpm` (the exact version number may differ).

After obtaining the package, it is installed as root on the node that is to become the head node, as follows:

```
[root@rhel6 ~]# rpm -ivh ddn-installer-ddn-129_cmddn.\
noarch.rpm
```

Because the installation of the package is done using `rpm` directly, and is not using a dependency resolver such as YUM, some packages may still need to be installed first. The administrator is prompted by the installer to install these packages, and they can be installed with YUM as usual. Installation progress is logged in `/var/log/install-ddn.log`.

2.4.3 Running The Installer

The Help Text For `install-ddn`

The installer is run with the command `install-ddn`. Running it without options displays the following help text:

```
[root@rhel6 ~]# install-ddn

  USAGE: install-ddn <OPTIONS1> [OPTIONS2]

  OPTIONS1:
  ---------
  -d | --fromdvd    Path to DVD device
  -i | --fromiso    Path to ISO image
  -n | --network    Install over network
  -l | --localrepo  Install using local repository

  OPTIONS2:
  ---------
  -h | --help       Print this help
  -c | --useconfig  Use predefined config file
  -r | --repodir    Create repository in specified directory
  -o | --overwrite  Overwrite existing repository
```

Usage Examples For `install-ddn`

- Install DirectMon directly from the DDN repositories over the internet:

  ```
  install-ddn -n
  ```

- Install DirectMon using a DDN DVD as the package repository:

  ```
  install-ddn -d /dev/sr0
  ```

- Install DirectMon using a DDN ISO as the package repository:

  ```
  install-ddn -i /tmp/ddn-centos5.iso
  ```

- Install DirectMon using a DDN ISO (`-i` option) as the package repository, and create the repository (`-r` option) in the specified directory:

  ```
  install-ddn -i /tmp/ddn-centos5.iso -r /tmp/repodir
  ```

- Install DirectMon using a DDN ISO as the package repository (`-i` option), and create the repository in the specified directory (`-r` option), but overwrite contents (`-o` option), if the directory already exists:

```
install-ddn -i /tmp/ddn-centos5.iso -r /tmp/repodir -o
```

- Install DirectMon from a local repository which has already been configured. This also assumes that the repository configuration files for zypper/YUM use are already in place:

  ```
  install-ddn -l
  ```

- Install DirectMon from a local repository (`-l` option) which has already been configured, and specify a path to the repository directory with the `-r` option. This can be used when the repository configuration file has not yet been generated and created. A repository configuration file is generated, and placed permanently in the appropriate directories (`/etc/yum.repos.d/` or `/etc/zypp/repos.d/`):

  ```
  install-ddn -l -r /tmp/repodir
  ```

An Installation Run For `install-ddn`
The most common installation option is with an internet connection. Any required software packages are asked for at the start:

Example

```
[root@rhel6 ~]# install-ddn -n

Please install the follow pre-requisites
----------------------------------------
createrepo
[root@rhel6 ~]# yum install createrepo
...
```

After all the packages are installed on the head node, the installer can be run again. It checks for some software conflicts, and warns about the ones it runs into.:

Example

```
[root@rhel6 ~]# install-ddn -n

  INFO/ERROR/WARNING:
  --------------------
  WARNING:
  A DHCP daemon is already running. DirectMon
  provides a customized DHCP server, and will  update the
  'dhcpd' configuration files. It is highly recommended
  that you stop your existing DHCP server, and let DDN
  Cluster Manager configure your dhcp server.

  You can also choose to ignore this message, and proceed
  with the existing DHCP server, which may or may not work.
  --------------------
Continue(c)/Exit(e)? e
[root@rhel6 ~]# /etc/init.d/dhcpd stop
Shutting down dhcpd:                                     [  OK  ]
```

Having resolved potential software conflicts, the product key (supplied by DDN or its vendor) is supplied:

Example

```
[root@rhel6 ~]# install-ddn -n
DirectMon Product Key Activation
---------------------------------------------
Product key [XXXXX-XXXXX-XXXXX-XXXXX-XXXXX]: 001323-134122-134134-\
314384-987986
...
License Parameters
------------------
             Country Name (2 letter code) []: US
       State or Province Name (full name) []: CA
                          Locality (city) []: San Francisco
         Organization Name (e.g. company) []: DDN
      Organization Unit (e.g. department) []: Development
                             Cluster Name []: ddnmon61
             MAC address [??:??:??:??:??:??]: 08:B8:BD:7F:59:4B

Submit certificate request to DDN? [y(yes)/n(no)]: y


Contacting license server ... License granted.
License has been installed in /cm/local/apps/cmd/etc/
```

The software license is displayed, and can be clicked through. Some warning is given about the configuration changes about to take place:

```
Please be aware that the DirectMon will re-write the
following configuration on your system:
- Update network configuration files.
- Start a DHCP server on the management network.
- Update syslog configuration
```

The software configuration sections is reached. Default DirectMon values are provided, but should normally be changed to appropriate values for the cluster. Questions asked are:

```
Management network parameters
-----------------------------
              Network Name [internalnet]:
               Base Address [10.141.0.0]:
                       Netmask Bits [16]:
               Domain Name [eth.cluster]:

Management interface parameters
-------------------------------
                    Interface Name [eth0]:
             IP Address [10.141.255.254]:

External network parameters
---------------------------
              Network Name [externalnet]:
                     Base Address [DHCP]:
                       Netmask Bits [24]:
                         Domain Name []: cm.cluster

External interface parameters
-----------------------------
                   Interface Name [eth1]:
                       IP Address [DHCP]:

External name servers list (space separated)
--------------------------------------------
                   List [10.150.255.254]:

Root password
-------------

Please enter the cluster root password:

MySQL root password
-------------------

Please enter the MySQL root password:
```

The DirectMon packages are then installed and configured. The stages include, towards the end:

Example

```
                  Setting up repositories  .....   [  OK  ]
             Installing required packages  .....   [  OK  ]
         Updating database authentication  .....   [  OK  ]
               Setting up MySQL database   .....   [  OK  ]
                          Starting syslog  .....   [  OK  ]
   Initializing cluster management daemon  .....   [  OK  ]
            Generating admin certificates  .....   [  OK  ]
       Starting cluster management daemon  .....   [  OK  ]
```

If all is well, a congratulatory message then shows up, informing the administrator that DirectMon has been installed successfully, that the host is now a head node.

Installing The Software Image For Regular Nodes After The `install-ddn` Installation Run

A functional cluster needs regular nodes to work with the head node. The regular nodes at this point of the installation still need to be set up. To do that, a software image (section 3.1.2) must now be created for the regular nodes on the head node. The regular nodes, when booting, use such a software image when they boot up to become a part of the cluster. A software image can be created using the base tar image included on the DVD, or as a custom image. The details on how to do this with `cm-create-image` are given in section 10.6.

Once the head node and software image have been built, the head node installation is complete, and the cluster is essentially at the same stage as that at the end of section 2.3.20 of the bare metal installation, except for that the software image is possibly a more customized image than the default image provided with the bare-metal installation.

The Build Configuration Files

This section is mainly intended for administrators who would like to deploy installations that are pre-configured. It can therefore be skipped in a first reading.

The build configuration file of a cluster contains the configuration scheme for a cluster. The bare metal and add-on installations both generate their own, separate build configuration files, stored in separate locations.

Most administrators do not deal with a build configuration file directly, partly because a need to do this arises only in rare and special cases, and partly because it is easy to make mistakes. An overview, omitting details, is given here to indicate how the build configuration file relates to the installations carried out in this chapter and how it may be used.

The bare metal build configuration file: The file at:
`/root/cm/build-config.xml`
on the head node contains cluster configuration settings and the list of distribution packages that are installed during the bare metal installation. Once the installation has completed, this file is static, and does not change as the cluster configuration changes.

The add-on installation build configuration file: Similarly, the file at:
`/root/.ddn/build-config.xml`
contains configuration settings. However, it does not contain a list of distribution packages. The file is created during the add-on installation, and if the installation is interrupted, the installation can be resumed at the point of the last confirmation prior to the interruption. This is done by using the `-c` option to `install-ddn` as follows:

Example

```
install-ddn -c /root/.ddn/build-config.xml
```

Both original "build" configuration XML files can be copied and installed via the `-i|import` option: For example:

```
service cmd stop
cmd -i build-config-copy.xml     #reinitializes CMDaemon from scratch
service cmd start
```

overwrites the old configuration. It means that the new cluster presents the same cluster manager configuration as the old one did initially. This can only be expected to work with identical hardware because of hardware dependency issues.

An XML configuration file can be exported via the `-x` option to `cmd`: For example:

```
service cmd stop
cmd -x myconfig.xml
service cmd start
```

Exporting the configuration is sometimes helpful in examining the XML configuration of an existing cluster after configuration changes have been made to the original installation. This "snapshot" can then, for example, be used to customize a `build-config.xml` file in order to deploy a custom version of DirectMon.

An exported configuration cannot replace the original bare-metal `build-config.xml` during the installation procedure. For example, if the original bare-metal file is replaced by the exported version by opening up another console with `alt-f2`, before the point where the "`Start`" button is clicked (figure 2.28), then the installation will fail. This is because the replacement does not contain the list of packages to be installed.

The exported configuration can however be used after a distribution is already installed. This is true for a head node that has been installed from bare-metal, and is also true for a head node that has undergone or is about to undergo an add-on installation. This is because a head node after a bare-metal installation, and also a head node that is ready for an add-on installation (or already has had an add-on installation done) do not rely on a packages list in the XML file.

These possibilities for an export file are indicated by the following table:

Can the `cmd -x` export file be used as-is for cluster installation?		
install type	**before or after installation**	**cluster installs from export?**
bare metal	before	no
bare metal	after	yes
add-on install	before	yes
add-on install	after	yes

Next, Chapter 3 introduces some of the tools and concepts that play a central role in DirectMon. Chapter 4 then explains how to configure and further set up the cluster.

3 Cluster Management With DirectMon

This chapter introduces cluster management with DirectMon. A cluster running DirectMon exports a cluster management interface to the outside world, which can be used by any application designed to communicate with the cluster.

Section 3.1 introduces a number of concepts which are key to cluster management using DirectMon.

Section 3.2 gives a short introduction on how the modules environment can be used by administrators. The modules environment provides facilities to control aspects of a users' interactive sessions and also the environment used by compute jobs.

Section 3.3 introduces how authentication to the cluster management infrastructure works and how it is used.

Section 3.4 and section 3.5 introduce the cluster management GUI (`cmgui`) and cluster management shell (`cmsh`) respectively. These are the primary applications that interact with the cluster through its management infrastructure.

Section 3.6 describes the basics of the cluster management daemon, CMDaemon, running on all nodes of the cluster.

3.1 Concepts

In this section some concepts central to cluster management with DirectMon are introduced.

3.1.1 Devices

A *device* in the DirectMon cluster management infrastructure represents physical hardware components of a cluster. A device can be any of the following types:

- Head Node
- Node
- Virtual SMP Node[1]

[1]a hardware component because it is fundamentally made up of several physical nodes, even though it is seen by the end user as virtual nodes

- GPU Unit
- Ethernet Switch
- InfiniBand Switch
- Myrinet Switch
- Power Distribution Unit
- Rack Sensor Kit
- Generic Device

A device can have a number of properties (e.g. rack position, hostname, switch port) which can be set in order to configure the device. Using the cluster management infrastructure, operations (e.g. power on) may be performed on a device. The property changes and operations that can be performed on a device depend on the type of device. For example, it is possible to mount a new filesystem to a node, but not to an Ethernet switch.

Every device that is present in the cluster management infrastructure has a device state associated with it. The table below describes the most important states for devices:

state	**device is**	**monitored by DDN?**	**state tracking?**
`UP`	UP	monitored	tracked
`DOWN`	DOWN	monitored	tracked
`UP/CLOSED`	UP	mostly ignored	tracked
`DOWN/CLOSED`	DOWN	mostly ignored	tracked

These, and other other states are described in more detail in section 6.5.

`DOWN` and `DOWN/CLOSED` states have an important difference. In the case of `DOWN`, the device is down, but is typically intended to be available, and thus typically indicates a failure. In the case of `DOWN/CLOSED`, the device is down, but is intended to be unavailable, and thus typically indicates that the administrator would like the device to be ignored.

3.1.2 Software Images

A *software image* is a blueprint for the contents of the local file-systems on a regular node. In practice, a software image is a directory on the head node containing a full Linux file-system.

The software image in a standard DirectMon installation is based on the same parent distribution that the head node uses. A different distribution can also be chosen after installation, from the distributions listed in section 1.1 for the software image. That is, the head node and the regular nodes can run different parent distributions. However, such a "mixed" cluster can be harder to manage and it is easier for problems to arise in such mixtures. Such mixtures, while supported, are therefore not recommended, and should only be administered by system administrators that understand the differences between Linux distributions.

RHEL6/CentOS6/SL6 mixtures are completely compatible with each other on the head and regular nodes. This is similarly true for RHEL5/CentOS5/SL5. On the other hand, RHEL5/CentOS5/SL5 may need some effort to work with RHEL6/CentOS6/SL6 in a mixture. Similarly, SLES may need some effort to work in a mixture with RHEL/CentOS/SL variants.

When a regular node boots, the node provisioning system (Chapter 6) sets up the node with a copy of the software image, which by default is called `default-image`.

Once the node is fully booted, it is possible to instruct the node to re-synchronize its local filesystems with the software image. This procedure can be used to distribute changes to the software image without rebooting nodes (section 6.6.2).

It is also possible to "lock" a software image so that no node is able to pick up the image until the software image is unlocked. (section 6.4.7).

Software images can be changed using regular Linux tools and commands (such as `rpm` and `chroot`). More details on making changes to software images and doing image package management can be found in Chapter 10.

3.1.3 Node Categories

The collection of settings in DirectMon that can apply to a node is called the configuration of the node. The administrator usually configures nodes using the `cmgui` (section 3.4) and `cmsh` (section 3.5) front end tools, and the configurations are managed internally with a database.

A *node category* is a group of regular nodes that share the same configuration. Node categories allow efficiency, allowing an administrator to:

- configure a large group of nodes at once. For example, to set up a group of nodes with a particular disk layout.
- operate on a large group of nodes at once. For example, to carry out a reboot on an entire category.

A node is in exactly one category at all times, which is `default` by default.

Nodes are typically divided into node categories based on the hardware specifications of a node or based on the task that a node is to perform. Whether or not a number of nodes should be placed in a separate category depends mainly on whether the configuration—for example: monitoring setup, disk layout, role assignment—for these nodes differs from the rest of the nodes.

A node inherits values from the category it is in. Each value is treated as the default property value for a node, and is overruled by specifying the node property value for the node.

One configuration property value of a node category is its software image (section 3.1.2). However, there is no requirement for a one-to-one correspondence between node categories and software images. Therefore multiple node categories may use the same software image, and multiple images may be used in the same node category.

By default, all nodes are placed in the `default` category. Alternative categories can be created and used at will, such as:

Example

Node Category	Description
`nodes-ib`	nodes with InfiniBand capabilities
`nodes-highmem`	nodes with extra memory
`login`	login nodes
`storage`	storage nodes

3.1.4 Node Groups

A *node group* consists of nodes that have been grouped together for convenience. The group can consist of any mix of all kinds of nodes, irrespective of whether they are head nodes or regular nodes, and irrespective of what (if any) category they are in. A node may be in 0 or more node groups at one time. I.e.: a node may belong to many node groups.

Node groups are used mainly for carrying out operations on an entire group of nodes at a time. Since the nodes inside a node group do not necessarily share the same configuration, configuration changes cannot be carried out using node groups.

Example

Node Group	Members
`brokenhardware`	`node087,node783,node917`
`headnodes`	`mycluster-m1,mycluster-m2`
`rack5`	`node212..node254`
`top`	`node084,node126,node168,node210`

One important use for node groups is in the `nodegroups` property of the provisioning role configuration (section 6.2.1), where a list of node groups that provisioning nodes provision is specified.

3.1.5 Roles

A *role* is a task that can be performed by a node. By assigning a certain role to a node, an administrator activates the functionality that the role represents on this node. For example, a node can be turned into provisioning node, or a storage node by assigning the corresponding roles to the node.

Roles can be assigned to individual nodes or to node categories. When a role has been assigned to a node category, it is implicitly assigned to all nodes inside of the category.

Some roles allow parameters to be set that influence the behavior of the role. For example, the `Slurm Client Role` (which turns a node into a Slurm client) uses parameters to control how the node is configured within Slurm in terms of queues and the number of GPUs.

When a role has been assigned to a node category with a certain set of parameters, it is possible to override the parameters for a node inside the category. This can be done by assigning the role again to the individual node with a different set of parameters. Roles that have been assigned to nodes override roles that have been assigned to a node category.

Examples of role assignment are given in sections 6.2.2 and 6.2.3.

3.2 Modules Environment

The *modules environment* is a third-party software (section 13.1) that allows users to modify their shell environment using pre-defined *modules*. A module may, for example, configure the user's shell to run a certain version of an application.

Details of the modules environment from a user perspective are discussed in the *User Manual*. However some aspects of it are relevant for administrators and are therefore discussed here.

3.2.1 Adding And Removing Modules

Modules may be loaded and unloaded, and also be combined for greater flexibility.

Modules currently installed are listed with:

```
module list
```

The modules available for loading are listed with:

```
module avail
```

Loading and removing specific modules is done with `module load` and `module remove`, using this format:

```
module load <MODULENAME1> [<MODULENAME2> ...]
```

For example, loading the `shared` module (section 3.2.2), the `gcc` compiler, the `openmpi` parallel library, and the `openblas/nehalem` 64-bit library, allows an MPI application to be compiled with Nehalem OpenBLAS optimizations:

Example

```
module add shared
module add gcc
module add openmpi/gcc
module add openblas/nehalem
mpicc -o myapp myapp.c
```

Specifying version numbers explicitly is typically only necessary when multiple versions of an application are installed and available. When there is no ambiguity, module names without a further path specification may be used.

3.2.2 Using Local And Shared Modules

Applications and their associated modules are divided into *local* and *shared* groups. Local applications are installed on the local file-system, whereas shared applications reside on a shared (i.e. imported) file-system.

It is recommended that the `shared` module be loaded by default for ordinary users. Loading it gives access to the modules belonging to shared applications, and allows the `module avail` command to show these extra modules.

Loading the `shared` module automatically for `root` is not recommended on a cluster where shared storage is not on the head node itself.

This is because root logins could be obstructed if this storage is not available, and if the root user relies on files in the shared storage.

On clusters without external shared storage, root can safely load the shared module automatically at login. This can be done by running the following command as root:

```
module initadd shared
```

Other modules can also be set to load automatically by the user at login by using "module initadd" with the full path specification. With the initadd option, individual users can customize their own default modules environment.

Modules can be combined in *meta-modules*. By default, the default-environment meta-module exists, which allows the loading of several modules at once by a user. Cluster administrators are encouraged to customize the default-environment meta-module to set up a recommended environment for their users. The default-environment meta-module is empty by default.

The administrator and users have the flexibility of deciding the modules that should be loaded in undecided cases via module *dependencies*. Dependencies can be defined using the prereq and conflict commands. The man page for modulefile gives details on configuring the loading of modules with these commands.

3.2.3 Setting Up A Default Environment For All Users

How users can set up particular modules to load automatically for their own use with the module initadd command is discussed in section 3.2.2.

How the administrator can set up particular modules to load automatically for all users by default is discussed in this section (section 3.2.3). In this example it is assumed that all users have just the following modules:

Example

```
[fred@ddnmon61 ~]$ module list
Currently Loaded Modulefiles:
  1) gcc/4.4.6     2) slurm
```

The Torque and Maui modules can then be set up by the administrator as a default for all users in the following 2 ways:

1. Defining part of a .profile to be executed for login shells. For example:

   ```
   [root@ddnmon61 ~]# cat /etc/profile.d/userdefaultmodules.sh
   module load shared
   module load torque
   module load maui
   ```

   Whenever users now carry out a bash login, these modules are loaded.

2. Instead of placing the modules directly in a script under profile.d like in the preceding item, a slightly more sophisticated way is to set the modules in the meta-module, /cm/shared/modulefiles/default-environment. For example:

```
[root@ddnmon61 ~]# cat /cm/shared/modulefiles/default-environment
#%Module1.0#####################################################
## default modulefile
##
proc ModulesHelp { } {
        puts stderr "\tLoads default environment modules for thi\
s cluster"
}
module-whatis   "adds default environment modules"

# Add any modules here that should be added by when a user loads\
 the 'default-environment' module
module add shared torque maui
```

The script `userdefaultmodules.sh` under `profile.d` then only needs to have the `default-environment` modules loaded in it:

```
[root@ddnmon61 ~]# cat /etc/profile.d/userdefaultmodules.sh
module load default-environment
```

Now, whenever the administrator adds or removes modules in the `default-environment` module, the module changes take effect for the user on login too.

3.2.4 Creating A Modules Environment Module

All module files are located in the `/cm/local/modulefiles` and `/cm/shared/modulefiles` trees. A module file is a TCL script in which special commands are used to define functionality. The `modulefile(1)` man page has more on this.

Cluster administrators can use the existing modules files as a guide to creating and installing their own modules for module environments, and can copy and modify a file for their own software if there is no environment provided for it already by DirectMon.

More details on the modules environment from the perspective of software installation are given in section 13.1.

3.3 Authentication

3.3.1 Changing Administrative Passwords On The Cluster

How to set up or change regular user passwords is not discussed here, but in Chapter 8 on user management.

Amongst the administrative passwords associated with the cluster are:

1. **The root password of the head node:** This allows a root login to the head node.

2. **The root password of the software images:** This allows a root login to a regular node, and is stored in the image file.

3. **The root password of the node-installer:** This allows a root login to the node when the node-installer, a stripped-down operating system, is running. The node-installer stage prepares the node for the

final operating system when the node is booting up. Section 6.4 discusses the node-installer in more detail.

4. **The root password of MySQL:** This allows a root login to the MySQL server.

To avoid having to remember the disparate ways in which to change these 4 passwords, the `cm-change-passwd` command runs a dialog prompting the administrator on which of them, if any, should be changed, as in the following example:

```
[root@ddnmon61 ~]# cm-change-passwd
With this utility you can easily change the following passwords:
  * root password of head node
  * root password of slave images
  * root password of node-installer
  * root password of mysql

Note: if this cluster has a high-availability setup with 2 head
      nodes, be sure to run this script on both head nodes.

Change password for root on head node? [y/N]: y
Changing password for root on head node.
Changing password for user root.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.

Change password for root in default-image [y/N]: y
Changing password for root in default-image.
Changing password for user root.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.

Change password for root in node-installer? [y/N]: y
Changing password for root in node-installer.
Changing password for user root.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.

Change password for MYSQL root user? [y/N]: y
Changing password for MYSQL root user.
Old password:
New password:
Re-enter new password:
```

For a failover configuration, the passwords are copied over automatically to the other head node when a change is made in the software image root password (case 2 on page 41).

For the remaining password cases (head root password, MySQL root password, and node-installer root password), the passwords are best "copied" over to the other head node by simply rerunning the script on the other head node.

Also, in the case of the password for software images used by the regular nodes: the new password that is set for a regular node only works on the node after the image on the node itself has been updated, with, for example, the `imageupdate` command (section 6.6.2). Alternatively, the new password can be made to work on the node by simply rebooting the node to pick up the new image.

The LDAP root password is a random string set during installation. Changing this is not done using `cm-change-password`. It can be changed as explained in Appendix O.

3.3.2 Logins Using `ssh`

The standard system login root password of the head node, the software image, and the node-installer, can be set using the `cm-change-passwd` command (section 3.3.1).

In contrast, `ssh` logins are set by default to be passwordless:

- For non-root users, an `ssh` passwordless login works if the `/home` directory that contains the authorized keys for these users is mounted. The `/home` directory is mounted by default.
- For the root user, an `ssh` passwordless login should always work since the authorized keys are stored in `/root`.

Users can be restricted from `ssh` logins

- on regular nodes using the `usernodelogin` (section 9.2.1) or "User node login" (section 9.2.2) settings.
- on the head node by modifying the sshd configuration on the head node. For example, to allow only root logins, the value of `AllowUsers` can be set in `/etc/ssh/sshd_config` to `root`. The man page for `sshd_config` has details on this.

3.3.3 Certificates

PEM Certificates And CMDaemon Front-end Authentication

While nodes in a DirectMon cluster accept ordinary `ssh` based logins , the cluster manager accepts public key authentication using `X509v3` certificates. Public key authentication using `X509v3` certificates means in practice that the person authenticating to the cluster manager must present their public certificate, and in addition must have access to the private key that corresponds to the certificate.

DirectMon uses the PEM format for certificates. In this format, the certificate and private key are stored as plain text in two separate PEM-encoded files, ending in `.pem` and `.key`.

Using `cmsh` and authenticating to the DirectMon: By default, one administrator certificate is created for root for the `cmsh` front end to interact with the cluster manager. The certificate and corresponding private key are thus found on a newly-installed DirectMon cluster on the head node at:

```
/root/.cm/cmsh/admin.pem
/root/.cm/cmsh/admin.key
```

The `cmsh` front end, when accessing the certificate and key pair as user `root`, uses this pair by default, so that prompting for authentication is then not a security requirement. The logic that is followed to access the certificate and key by default is explained in detail in item 2 on page 315.

Using `cmgui` and authenticating to the DirectMon: When an administrator uses the `cmgui` front end, the same certificate pair as used by `cmsh` is used. Running `cmgui` from a desktop, that is, a location other than the head node, is described in section 3.4.1, and being prompted for authentication when doing so is an expected security requirement. In theory, authentication is not a security requirement if the user is already logged into the head node and running `cmgui` from there as that user. However, for consistency reasons, the `cmgui` front end always prompts for user authentication to take place, unless the `Password` field and `Connect at start-up` checkbox in the dialog of figure 3.3 are both already filled in.

If the administrator certificate and key are replaced, then any other certificates signed by the original administrator certificate must be generated again using the replacement, because otherwise they will no longer function.

Certificate generation in general, including the generation and use of non-administrator certificates, is described in greater detail section 8.4.

Replacing A Temporary Or Evaluation License

In the preceding section, if a license is replaced, then regular user certificates need to be generated again. Similarly, if a temporary or evaluation license is replaced, regular user certificates need to be generated again. This is because the old user certificates are signed by a key that is no longer valid. The generation of non-administrator certificates and how they function is described in section 8.4.

3.3.4 Profiles

Certificates that authenticate to the cluster management infrastructure contain a *profile*.

A profile determines which cluster management operations the certificate holder may perform. The administrator certificate is created with the `admin` profile, which is a built-in profile that allows all cluster management operations to be performed. In this sense it is similar to the `root` account on Unix systems. Other certificates may be created with different profiles giving certificate owners access to a pre-defined subset of the cluster management functionality (section 8.4).

3.4 Cluster Management GUI

This section introduces the basics of the cluster management GUI (`cmgui`). This is the graphical interface to cluster management in DirectMon. It can be run from the head node or on a login node of the cluster using X11-forwarding:

Example

```
user@desktop:~> ssh -X root@mycluster cmgui
```

However, typically it is installed and run on the administrator's desktop computer. This saves user-discernable lag time if the user is hundreds of kilometers away from the head node.

3.4.1 Installing Cluster Management GUI On The Desktop

To install `cmgui` on a desktop computer a Firefox browser, version 10 or greater, must first be installed onto the operating system. The `cmgui` installation package corresponding to the CMDaemon version must then be downloaded to the desktop.

For a default repository configuration, doing a `yum update`, or `zypper up` for SLES-based distributions, ensures version compatibility on the head node. The appropriate installation package for the desktop can then be picked up from the head node of any DirectMon cluster under the directory:

```
/cm/shared/apps/cmgui/dist/
```

Installation packages are available for Linux, for Windows XP/Vista/Windows 7/Windows 8, and for Mac OS.

The installation package can be placed in a convenient and appropriately accessible location on the desktop.

On the Mac OS desktop, `cmgui` is installed from the Mac OS `pkg` file by clicking on it and following the installation procedure. The `cmgui` front end is then run by clicking on the `cmgui` icon.

On the MS Windows desktop, `cmgui` is installed from an `.exe` installer file by running it and following the installation procedure. After the installation, `cmgui` is started through the Start menu or through the desktop shortcut.

For Linux, the installation of a 64-bit Firefox browser on the desktop, and not a 32-bit version, is mandatory, before `cmgui` is installed. The version of Firefox on the desktop can be checked and replaced if necessary. The check can be done similarly to the following (some output elided):

Example

```
root@work:~# file /usr/bin/firefox
/usr/bin/firefox:  ELF 32-bit LSB executable, Intel 80386,...
root@work:~# apt-get install firefox:amd64
...
root@work:~# exit
```

For the Linux desktop, `cmgui` can then be installed and run by:

- copying over the `.tar.bz2` file
- untarring the `.tar.bz2` file
- running `cmgui` from inside the directory that the untar process created.

The install and run procedure may look similar to (some output elided):

Example

```
me@work:~$ scp root@ddnmon61:/cm/shared/apps/cmgui/dist/\
cmgui..r4557.tar.bz2 .
...
me@work:~$ tar -xjf cmgui..r4557.tar.bz2
...
me@work:~/cmgui--r4557$ ./cmgui
```

DirectMon `cmgui` Welcome Screen

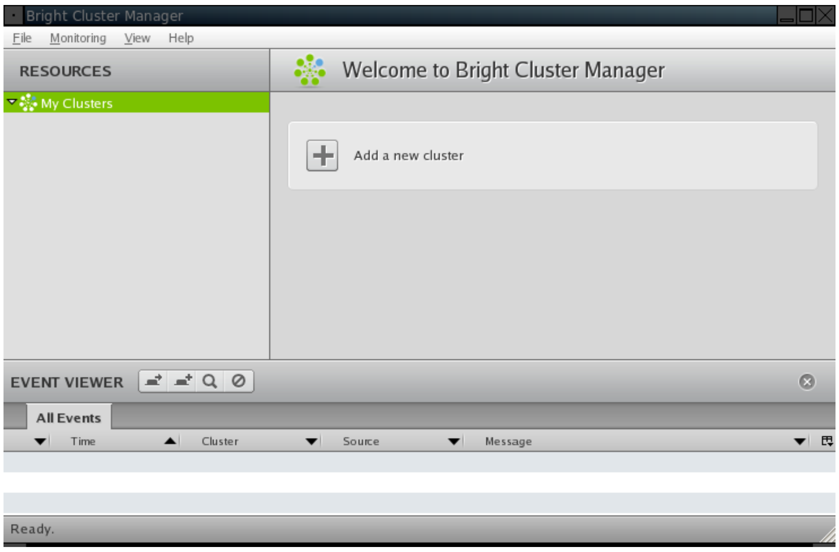
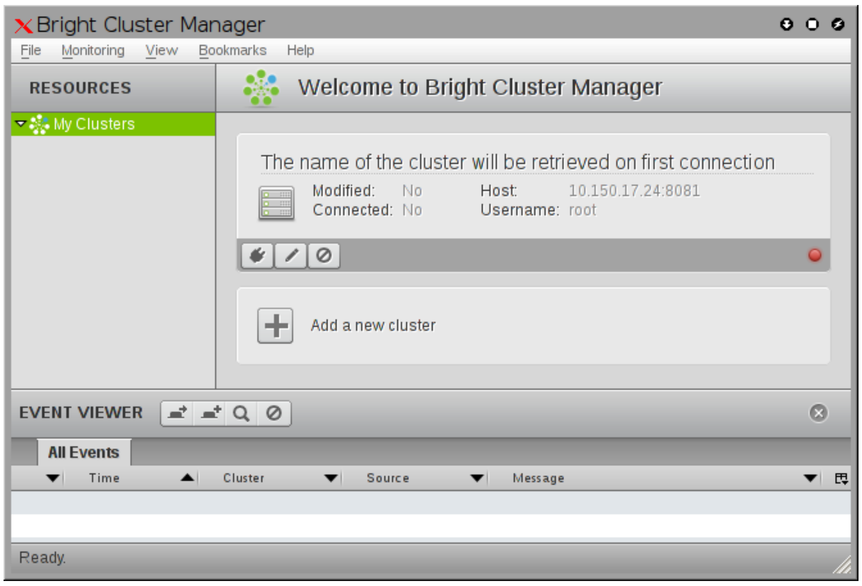


Figure 3.1: Cluster Manager GUI welcome screen

When `cmgui` is started for the first time, the welcome screen (figure 3.1) is displayed.

To configure `cmgui` for connections to a new DirectMon cluster, the cluster is added to `cmgui` by clicking the ⊞ button in the welcome screen. More clusters can be added within `cmgui` as needed.

After a cluster is added, the screen displays the connection parameters for the cluster (figure 3.2).


Bright Cluster Manager
File Monitoring View Bookmarks Help
RESOURCES
My Clusters
Welcome to Bright Cluster Manager
Add a new cluster
EVENT VIEWER
All Events
Time Cluster Source Message
Ready.

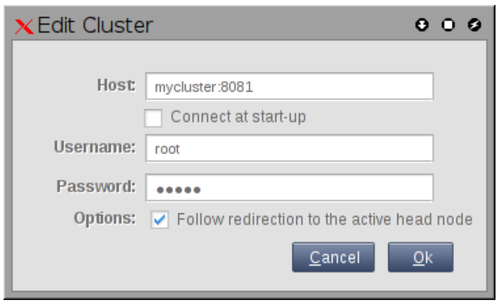
The name of the cluster will be retrieved on first connection

Modified: No Host: 10.150.17.24:8081
Connected: No Username: root

Figure 3.2: Connecting to a cluster

DirectMon `cmgui` Connection Parameters Dialog Window

Figure 3.3 shows the dialog window in which the connection parameters can be entered.



Host: mycluster:8081
☐ Connect at start-up
Username: root
Password: •••••
Options: ✓ Follow redirection to the active head node

Figure 3.3: Editing The Cluster Connection Parameters

The dialog window in figure 3.3 has several options:

- The `Host` field can be a name or an IP address. If the port on the host is not specified, then port 8081 is added automatically.
- The "`Connect at start-up`" checkbox option offers convenience by attempting to connect to the cluster right away when starting up `cmgui`, without waiting for the administrator to click on the connect button of figure 3.2.
- The "`Username`" field accepts the name of the user that `cmgui` is to run as. So, `cmgui` can be run by a non-root user as user `root` as well as run the other way round.
- The `Password` field affects connection behavior when starting up `cmgui`, or when clicking on the connect button of figure 3.2. The behavior is generally obvious, and depends on whether a password has been saved in the `Password` field:

1. With a correct password set, and all other fields in the dialog set correctly, a connection to the cluster is established when starting up `cmgui` if the "`Connect at start-up`" state has been set. The cluster overview screen (figure 3.4) for the cluster is then displayed.
2. With a blank password, but all other fields in the dialog set correctly, and if continuing in a current `cmgui` session where a previous successful connection to the cluster has been disconnected, then clicking on the connect button of figure 3.2 establishes a successful connection to the cluster, and the cluster overview screen (figure 3.4) for the cluster is then displayed.
3. With a blank password, but all other fields in the dialog set correctly, and if the cluster is being connected to for the first time during a new `cmgui` session, then a password prompt dialog is displayed if:
   - starting `cmgui` and if the "`Connect at start-up`" state has been set
     or
   - if `cmgui` is already running and the connect button in the screen shown in figure 3.2 is clicked.

   The cluster overview screen (figure 3.4) for the cluster in this case (case number 3) is only displayed after a successful authentication.

The administrator should be aware that storing the password is unsafe if untrusted people can access the `cmgui` files in which the password is stored. These are kept on the machine that runs the `cmgui` process, which may not be the machine that displays `cmgui` on its screen.

Avoiding Lag With `cmgui`

The `cmgui` front-end is often run on the head node of the cluster via an ssh -X login, with the -X option passing the X11 protocol instructions. With such a connection, the transport lag for X11 instructions is greater if the X-display server, where the administrator is viewing `cmgui`, is some distance away from the head where `cmgui` is actually running. Also, sometimes running `cmgui` from the head node directly via ssh -X is not possible because there is a bastion host or a similar jump host in the way.

A way to deal with the bastion/jump host problem is to carry out an ssh -X login to the bastion host first, and then to carry out an ssh -X login to the head node to run the `cmgui` session. This *double-ssh -X* connection is easy enough to implement. However it has several problems.

- The X11-protocol overhead and lag becomes larger because there are now two jumps between the cluster and the remote display server. Viewing `cmgui` in this way is therefore likely to be even more irritating than before.
- Microsoft windows `cmgui` clients will need to be displayed over an X-display server too, which means installing an X-display server for MS windows. This is often undesirable.

Because of these problems, alternatives to double-ssh -X are therefore often used:

- One workaround is to run `cmgui` over an ssh -X connection from the bastion host, and then use VNC (a remote desktop viewing software) from the remote display-server to the bastion host. This is usually more pleasant for interactive use than double-ssh -X because
  - the bastion host is usually close to the cluster, thus minimizing the X11 lag from the cluster head node to the bastion host.
  - using VNC in the connection to transport the desktop image from the bastion host to the remote display avoids the X11 protocol transport lag issue entirely for that part of the connection.

    For implementing VNC, an administrator should note that:
    * most modern standard versions of VNC can connect with compression
    * encryption and authentication for VNC must be enabled in order to avoid unauthorized VNC access. If enabling these is a problem, then the well-trusted OpenVPN utility can be used to provide these in a separate layer instead with good results.
- Usually the best workaround to set up `cmgui` to display on the desktop is to run `cmgui` as the desktop client on the remote display, and set up ssh port-forwarding on the bastion host. This again avoids the transport overhead and lag of X11 over double-ssh -X, and indeed avoids transporting the X11 protocol entirely. Instead, the ssh connection transports the CMDaemon SOAP calls directly. These SOAP calls run encrypted over the SSL port 8081 already, so that all that is needed is to forward them, which is what the ssh port-forwarding feature is able to do quite easily. The additional encryption from ssh on top of the SSL encryption already operating on the SOAP calls does no harm. It is unlikely to be a performance bottleneck, and in any case can be switched off or speeded up in some implementations or compiler options of ssh.

  The ssh port-forwarding connection can be configured by following these steps:
  - For failover clusters, to connect to the active node, `cmgui` needs to be aware that it should not follow redirection. To arrange this, the `cmgui` settings file `~/.cm/cmgui/clusters.dat` should be modified on the machine where the `cmgui` desktop client is running. The safest way to carry out the modification is to first make sure that the file is created by starting up the desktop `cmgui`, then stopping it. The `followRedirect` line in the `clusters.dat` file should then be changed to:

    ```
    followRedirect = false;
    ```

    Single-head clusters require no such change.

– Port-forwarding can then be set up from the cluster.

  If these parameters are used:

  * *cluster-ip*: the cluster address. This is the shared alias IP address for a failover cluster
  * *bast*: the bastion host.
  * *user*: the user name that the administrator is using
  * *admin-desktop*: the remote machine

  Then:

  * If the bastion host allows no inbound ssh connections from the administrator's remote desktop, then:
    · a remote-port-forwarding tunnel can be set up on it by running:

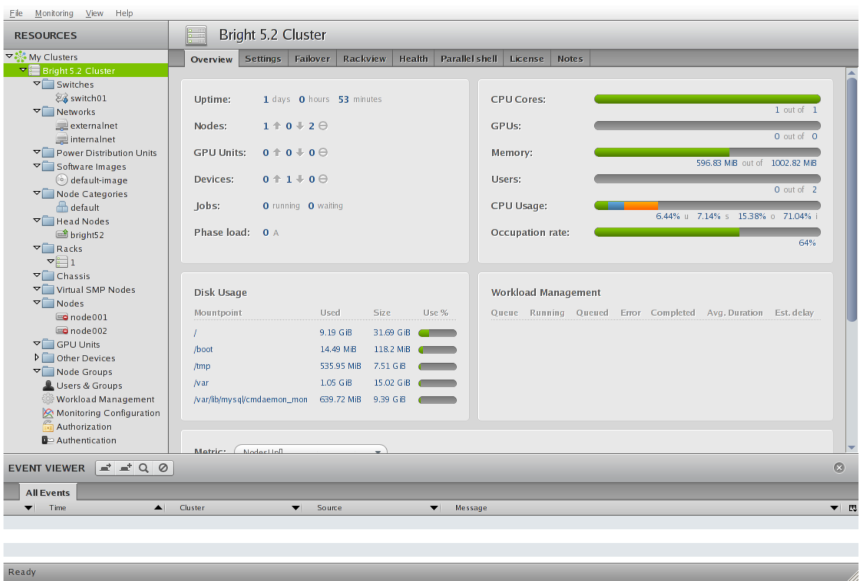
    ```
    bast:~$ ssh -R 8081:cluster-ip:8081 user@admin-\
    desktop
    ```

    A login to *admin-desktop* is prompted for, as part of the execution of the preceding command, and the login should be carried out.

    · on the remote desktop, after `cmgui` is started up, a connection to `localhost:8081` should be made. The value `localhost:8081` can be set in the `cmgui` cluster connection parameters dialog shown in figure 3.3.

DirectMon `cmgui` Default Display On Connection

Clicking on the `Connect` button establishes a connection to the cluster, and `cmgui` by default then displays a tabbed pane overview screen of the cluster (figure 3.4):



Uptime: 1 days 0 hours 53 minutes
Nodes: 1 ↑ 0 ↓ 2 ⊖
GPU Units: 0 ↑ 0 ↓ 0 ⊖
Devices: 0 ↑ 1 ↓ 0 ⊖
Jobs: 0 running 0 waiting
Phase load: 0 A

CPU Cores: 1 out of 1
GPUs: 0 out of 0
Memory: 596.83 MiB out of 1002.82 MiB
Users: 0 out of 2
CPU Usage: 6.44% u 7.14% s 15.38% o 71.04% i
Occupation rate: 64%

Disk Usage

Mountpoint	Used	Size	Use %
/	9.19 GiB	31.69 GiB	
/boot	14.49 MiB	118.2 MiB	
/tmp	535.95 MiB	7.51 GiB	
/var	1.05 GiB	15.02 GiB	
/var/lib/mysql/cmdaemon_mon	639.72 MiB	9.39 GiB	

Workload Management

Queue	Running	Queued	Error	Completed	Avg. Duration	Est. delay

Time	Cluster	Source	Message

Figure 3.4: Cluster Overview

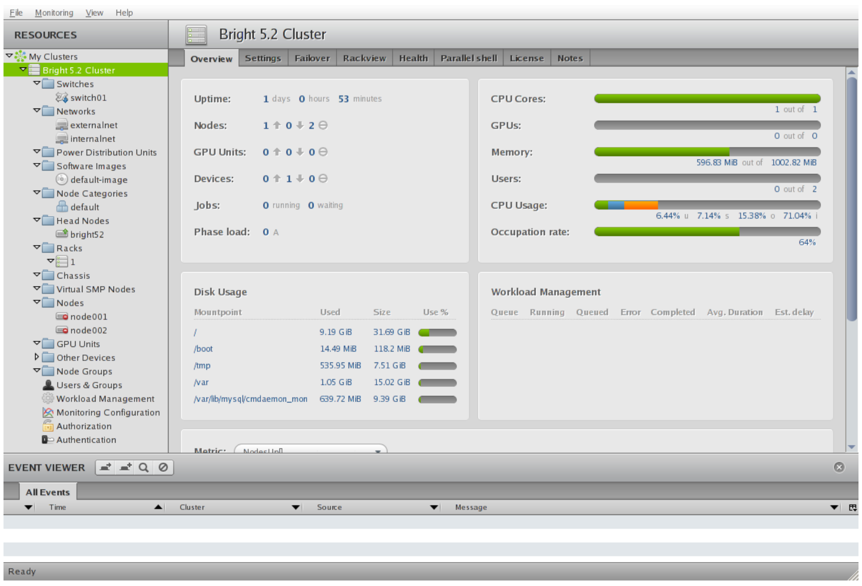
3.4.2 Navigating The Cluster Management GUI

Aspects of the cluster can be managed by administrators using `cmgui` (figure 3.4).

The resource tree, displayed on the left side of the window, consists of hardware resources such as nodes and switches as well as non-hardware resources such as `Users & Groups` and `Workload Management`. Selecting a resource opens an associated tabbed pane on the right side of the window that allows tab-related parameters to be viewed and managed.

The number of tabs displayed and their contents depend on the resource selected. The following standard tabs are available for most resources:

- `Overview`: provides an overview containing the most important status details for the resource.
- `Tasks`: accesses tasks that operate on the resource.
- `Settings`: allows configuration of properties of the resource.



Uptime:	1 days 0 hours 53 minutes
Nodes:	1 ⬆ 0 ⬇ 2 ⊖
GPU Units:	0 ⬆ 0 ⬇ 0 ⊖
Devices:	0 ⬆ 1 ⬇ 0 ⊖
Jobs:	0 running 0 waiting
Phase load:	0 A

CPU Cores:	1 out of 1
GPUs:	0 out of 0
Memory:	596.83 MiB out of 1002.82 MiB
Users:	0 out of 2
CPU Usage:	6.44% u 7.14% s 15.38% o 71.04% i
Occupation rate:	64%

Disk Usage

Mountpoint	Used	Size	Use %
/	9.19 GiB	31.69 GiB	
/boot	14.49 MiB	118.2 MiB	
/tmp	535.95 MiB	7.51 GiB	
/var	1.05 GiB	15.02 GiB	
/var/lib/mysql/cmdaemon_mon	639.72 MiB	9.39 GiB	

Workload Management

Queue	Running	Queued	Error	Completed	Avg. Duration	Est. delay

Metric: NodesUp

Figure 3.5: Node Settings

For example, the `Settings` tab of the `node001` resource (figure 3.5) displays properties, such as the hostname, that can be changed. The `Save` button on the bottom of the tab makes the changes active and permanent, while the `Revert` button undoes all unsaved changes.

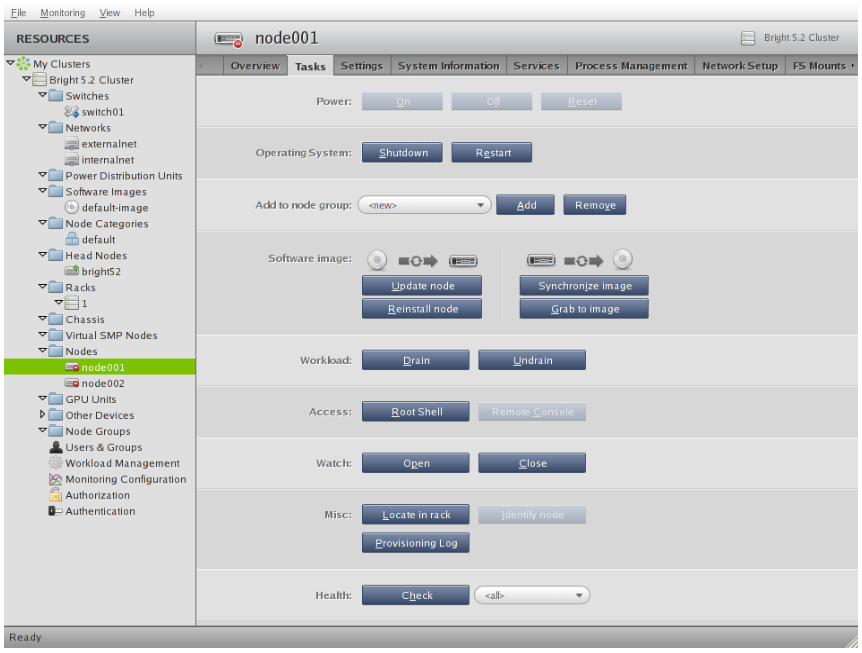
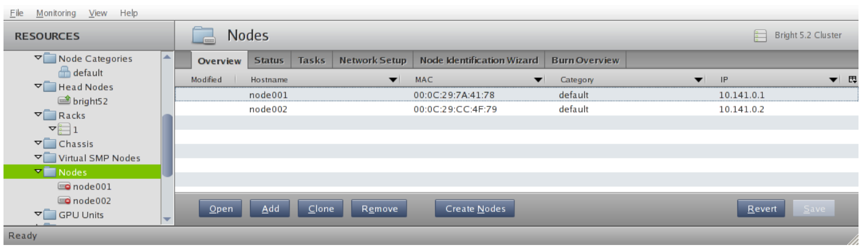


Figure 3.6: Node Tasks

Figure 3.6 shows the `Tasks` tab of the `node001` resource. The tab displays operations that can be performed on the `node001` resource. Details on setting these up, their use, and meaning are provided in the remaining chapters of this manual.

It is also possible to select a resource folder (rather than a resource item) in the tree. For example: `Node Categories`, `Nodes`, and `Networks`. Selecting a resource folder in the tree by default opens an `Overview` tab, which displays a list of resource items inside the folder. These are displayed in the resource tree and in the tabbed pane. Resource items in the tabbed pane can be selected, and operations carried out on them by clicking on the buttons at the bottom of the tabbed pane. For example, for `Nodes`, one or more nodes can be selected, and the `Open`, `Add`, `Clone` and `Remove` buttons can be clicked to operate on the selection (figure 3.7).



Modified	Hostname	MAC	Category	IP
	node001	00:0C:29:7A:41:78	default	10.141.0.1
	node002	00:0C:29:CC:4F:79	default	10.141.0.2

Figure 3.7: Nodes Overview

3.4.3 Advanced `cmgui` Features

This section describes some advanced features of `cmgui`. It may be skipped on first reading.

Within `cmgui` the right mouse button often brings up convenient options. Two of these are labor-saving options that synchronize properties within resource items.

Synchronize Values To Other Items In Current Resource

"`Synchronize values to other` <*items in current resource*>" is a menu option that synchronizes the properties of the item that is currently selected to other items within the same resource.

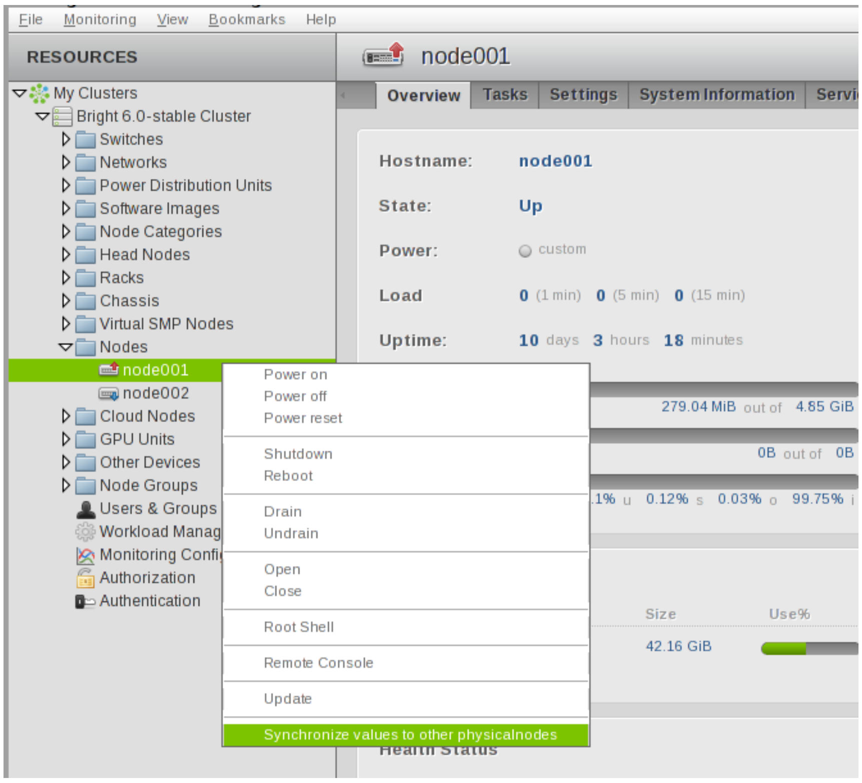
Some examples:

1. `Synchronize values to other fsmounts`: Within the current category of `Node Categories`, for a mount point within the `FS Mounts` tab, synchronize the mount point values for the selected mount point to other mount points under the same `FS Mounts` tab.
2. `Synchronize values to other network interfaces`: Synchronize network interface values of the selected network interface to other network interfaces, within the `Network Setup` tab.
   (a) For a head node, this is done within the selected head node in the `Head Nodes` resource.
   (b) For a regular node, this is done within the selected node in the `Nodes` resource.
3. `Synchronize values to other physicalnodes`: For the selected node in the `Nodes` resource, synchronize the values of the node to other nodes.

Item 2b can be regarded as a subset of item 3.

Item 3 in the list is a good example to elaborate upon, to illustrate what is meant:

Within the `Nodes` resource in the resources menu, right-clicking on a node item brings up a menu (figure 3.8),



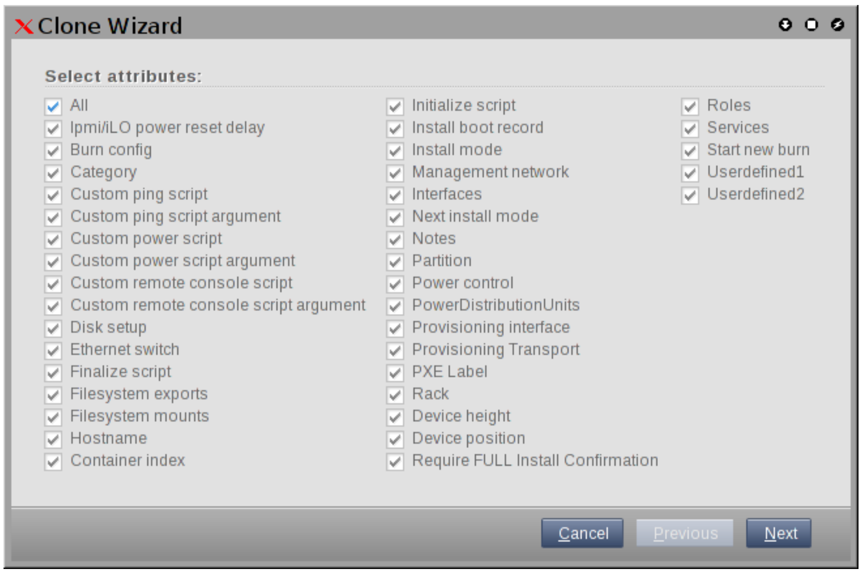
Hostname: node001
State: Up
Power: custom
Load 0 (1 min) 0 (5 min) 0 (15 min)
Uptime: 10 days 3 hours 18 minutes

279.04 MiB out of 4.85 GiB
0B out of 0B
1% u 0.12% s 0.03% o 99.75% i

Size	Use%
42.16 GiB	

Figure 3.8: Right Click Menu Options

Selecting `Synchronize Values to other physicalnodes` brings up a `Clone Wizard`. The wizard helps the administrator choose the node property values to be synchronized (figure 3.9), and also to decide to which of the other nodes (figure 3.10) in the `Nodes` resource these property values are to be synchronized to.


Clone Wizard

Select attributes:

✓ All	✓ Initialize script	✓ Roles
✓ Ipmi/iLO power reset delay	✓ Install boot record	✓ Services
✓ Burn config	✓ Install mode	✓ Start new burn
✓ Category	✓ Management network	✓ Userdefined1
✓ Custom ping script	✓ Interfaces	✓ Userdefined2
✓ Custom ping script argument	✓ Next install mode	
✓ Custom power script	✓ Notes	
✓ Custom power script argument	✓ Partition	
✓ Custom remote console script	✓ Power control	
✓ Custom remote console script argument	✓ PowerDistributionUnits	
✓ Disk setup	✓ Provisioning interface	
✓ Ethernet switch	✓ Provisioning Transport	
✓ Finalize script	✓ PXE Label	
✓ Filesystem exports	✓ Rack	
✓ Filesystem mounts	✓ Device height	
✓ Hostname	✓ Device position	
✓ Container index	✓ Require FULL Install Confirmation	

Figure 3.9: Clone Wizard Synchronization: Attributes Choice

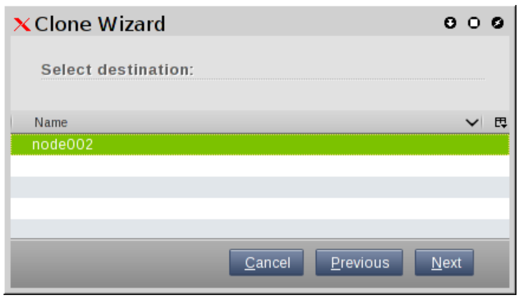
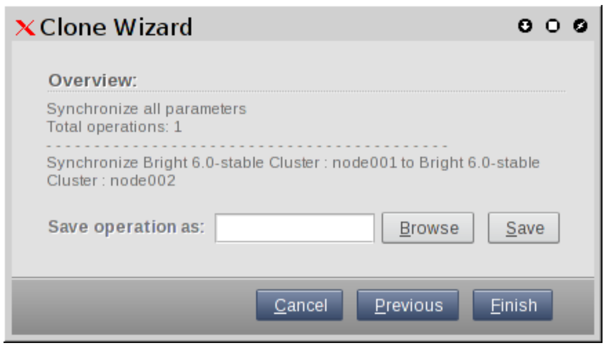


Figure 3.10: Clone Wizard Synchronization: Nodes Selection

A final overview (figure 3.11) explains what is going to be synchronized when the `Finish` button of the wizard is clicked.



Synchronize all parameters
Total operations: 1
- -
Synchronize Bright 6.0-stable Cluster : node001 to Bright 6.0-stable Cluster : node002

Figure 3.11: Clone Wizard Synchronization: Final Overview

Before clicking `Finish` button, the operation can also optionally be saved to a file.

- To save the file: The file location should be specified as a full path. Typically the directory `/cm/shared/apps/cmgui/` is used if `cmgui` is run from the head node.
- To load the file: A previously-saved operation can be re-executed by selecting it from the `Tools` menu.

Export To Other Items Outside Current Resource

"`Export to other` *<items outside current resource>*" is a menu option that exports the properties of the current item to other items outside the parent resource providing the item. So it synchronizes, but to items outside the direct parent resource. For example:

1. `Export to other Categories`: Within the current category of `Node Categories`, for a mount point within the `FS Mounts` tab, export the mount point values for the selected mount point to other mount points outside the original `FS Mounts` tab.
2. Export network interface values of the selected network interface to other network interfaces, outside the current `Network Setup` tab.
   (a) `Export to other masternodes` is the option used for a head node. It exports to the other head node items in the `Head Nodes` resource

(b) `Export to other physicalnodes` is the option used for a regular node. It exports to the other node items in the `Nodes` resource

The numbering of the items listed here corresponds to the items in the `Synchronize...` list from earlier (page 53), except that item 3 in the that earlier list has no corresponding export menu option.

Just like with the `Synchronize...` menu option from earlier (page 53), selecting the `Export...` menu option brings up a Clone Wizard. The wizard helps the administrator choose the values to be exported, where to export them to, and allows the operation to be saved for later re-execution from the `Tools` menu.

3.5 Cluster Management Shell

This section introduces the basics of the cluster management shell, `cmsh`. This is the command-line interface to cluster management in DirectMon. Since `cmsh` and `cmgui` give access to the same cluster management functionality, an administrator need not become familiar with both interfaces. Administrators intending to manage a cluster with only `cmgui` may therefore safely skip this section.

The `cmsh` front end allows commands to be run with it, and can be used in batch mode. Although `cmsh` commands often use constructs familiar to programmers, it is designed mainly for managing the cluster efficiently rather than for trying to be a good or complete programming language. For programming cluster management, the use of Python bindings (Appendix K) is generally recommended instead of using `cmsh` in batch mode.

Usually `cmsh` is invoked from an interactive session (e.g. through `ssh`) on the head node, but it can also be used to manage the cluster from outside.

3.5.1 Invoking `cmsh`

From the head node, `cmsh` can be invoked as follows:

```
[root@mycluster ~]# cmsh
[mycluster]%
```

Running `cmsh` without arguments starts an interactive cluster management session. To go back to the Unix shell, a user enters `quit`:

```
[mycluster]% quit
[root@mycluster ~]#
```

The `-c` flag allows `cmsh` to be used in batch mode. Commands may be separated using semi-colons:

```
[root@mycluster ~]# cmsh -c "main showprofile; device status apc01"
admin
apc01 ............... [   UP   ]
[root@mycluster ~]#
```

Alternatively, commands can be piped to `cmsh`:

```
[root@mycluster ~]# echo device status | cmsh
apc01 ............... [   UP    ]
mycluster ........... [   UP    ]
node001 ............. [   UP    ]
node002 ............. [   UP    ]
switch01 ............ [   UP    ]
[root@mycluster ~]#
```

In a similar way to Unix shells, `cmsh` sources dotfiles upon start-up in both batch and interactive mode. In the following list of dotfiles, a setting in the file that is in the shorter path will override a setting in the file with the longer path (i.e.: "shortest path overrides"):

- `~/.cm/cmsh/.cmshrc`
- `~/.cm/.cmshrc`
- `~/.cmshrc`

If there is no dotfile value set for the user, then, the file `/etc/cmshrc`, which is accessible to all users, is checked. If the check finds the value exists in `/etc/cmshrc`, then that value is used.

Sourcing settings is convenient when defining command aliases. Command aliases can be used to abbreviate longer commands. For example, putting the following in `.cmshrc` allows the `ds` command to be used as an alias for `device status`:

Example

```
alias ds device status
```

The options usage information for `cmsh` is obtainable with `cmsh -h` (figure 3.12).

```
Usage:
  cmsh [options] ................ Connect to localhost using default port
  cmsh [options] <--certificate|-i certfile> <--key|-k keyfile> <host[:port]>
            Connect to a cluster using certificate and key in PEM format

       Valid options:
       --help|-h ..................... Display this help
       --noconnect|-u ................ Start unconnected
       --controlflag|-z .............. ETX in non-interactive mode
       --nocolor|-o .................. Do not use misc. colors in the text output
       --noredirect|-r ............... Do not follow redirects
       --norc|-n ..................... Do not load cmshrc file on start-up
       --noquitconfirmation|-Q ....... Do not ask for quit confirmation
       --command|-c <"c1; c2; ..."> .. Execute commands and exit
       --file|-f <filename> .......... Execute commands in file and exit
       --echo|-x ..................... Echo all commands
       --quit|-q ..................... Exit immediately after error
```

Figure 3.12: Usage information for `cmsh`

3.5.2 Levels, Modes, Help, And Commands Syntax In `cmsh`

The *top-level* of `cmsh` is the level that `cmsh` is in when entered without any options.

To avoid overloading a user with commands, cluster management functionality has been grouped and placed in separate `cmsh` *modes*. Modes and their levels are a hierarchy available below the top-level, and therefore to perform cluster management functions, a user switches and descends into the appropriate mode.

Figure 3.13 shows the top-level commands available in `cmsh`. These commands are displayed when `help` is typed in at the top-level of `cmsh`:

```
alias ........................ Set aliases
category ..................... Enter category mode
cert ......................... Enter cert mode
color ........................ Manage console text color settings
cloud ........................ Enter cloud mode
connect ...................... Connect to cluster
delimiter .................... Display/set delimiter
device ....................... Enter device mode
disconnect ................... Disconnect from cluster
events ....................... Manage events
exit ......................... Exit from current object or mode
export ....................... Display list of aliases current list formats
help ......................... Display this help
history ...................... Display command history
jobqueue ..................... Enter jobqueue mode
jobs ......................... Enter jobs mode
list ......................... List state for all modes
main ......................... Enter main mode
modified ..................... List modified objects
monitoring ................... Enter monitoring mode
network ...................... Enter network mode
nodegroup .................... Enter nodegroup mode
partition .................... Enter partition mode
process ...................... Enter process mode
profile ...................... Enter profile mode
quit ......................... Quit shell
quitconfirmation ............. Manage the status of quit confirmation
rack ......................... Enter rack mode
refresh ...................... Refresh all modes
run .......................... Execute cmsh commands from specified file
session ...................... Enter session mode
softwareimage ................ Enter softwareimage mode
unalias ...................... Unset aliases
user ......................... Enter user mode
```

Figure 3.13: Top level commands in `cmsh`

All levels inside `cmsh` provide these top-level commands.

Passing a command as an argument to `help` gets details for it:

Example

```
[myheadnode]% help run
Name:
```

```
          run - Execute all commands in the given file(s)

Usage:
          run [OPTIONS] <filename> [<filename2> ...]

Options:
          -x, --echo
               Echo all commands

          -q, --quit
               Exit immediately after error

[myheadnode]%
```

In the general case, invoking `help` at any level without an argument provides the list of top-level commands, followed by commands that may be used at that level (list of top-level commands elided in example below):

Example

```
[myheadnode]% session
[myheadnode->session]% help
============================ Top =============================
...
========================== session ===========================
id ...................... Display current session id
killsession ............. Kill a session
list .................... Provide overview of active sessions
[myheadnode->session]%
```

In the preceding example, `session` mode is entered, and help without any argument lists the possible commands at that level.

To enter a mode, a user enters the mode name at the `cmsh` prompt. The prompt changes to indicate that `cmsh` is in the requested mode, and commands for that mode can then be run. To leave a mode, and go back up a level, the `exit` command is used. At the top-level, `exit` has the same effect as the `quit` command, that is, the user leaves `cmsh` and returns to the Unix shell. The string `..` is an alias for `exit`.

Example

```
[ddnmon61]% device
[ddnmon61->device]% list
Type                 Hostname (key)   MAC                Category
-------------------- ---------------- ------------------ ---------
EthernetSwitch       switch01         00:00:00:00:00:00
HeadNode             ddnmon61         00:0C:29:5D:55:46
PhysicalNode         node001          00:0C:29:7A:41:78  default
PhysicalNode         node002          00:0C:29:CC:4F:79  default
[ddnmon61->device]% exit
[ddnmon61]%
```

A command can also be executed in a mode without staying within that mode. This is done by specifying the mode before the command that is to be executed within that node. Most commands also accept arguments after the command. Multiple commands can be executed in one line by separating commands with semi-colons.

A `cmsh` input line has the following syntax:

<mode> <cmd> <arg> . . . <arg>; . . . ; <mode> <cmd> <arg> . . . <arg>

where *<mode>* and *<arg>* are optional. [2]

Example

```
[ddnmon61->network]% device status ddnmon61; list
ddnmon61 ............ [   UP   ]
Name (key)  Type     Netmask bits Base address Domain name
----------- -------- ------------ ------------ --------------------
externalnet External 16           192.168.1.0  ddn.com
globalnet   Global   0            0.0.0.0      cm.cluster
internalnet Internal 16           10.141.0.0   eth.cluster
[ddnmon61->network]%
```

In the preceding example, while in `network` mode, the `status` command is executed in `device` mode on the host name of the head node, making it display the status of the head node. The `list` command on the same line after the semi-colon still runs in `network` mode, as expected, and displays a list of networks.

3.5.3 Working With Objects

Modes in `cmsh` work with associated groupings of data called *objects*. For instance, `device` mode works with `device` objects, and `network` mode works with `network` objects. The commands used to deal with objects are the same in all modes, although generally not all of them function with an object:

Command	Description
`use`	Use the specified object. I.e.: Make the specified object the *current object*
`add`	Create the object and use it
`assign`	Assign a new object
`unassign`	Unassign an object
`clone`	Clone the object and use it
`remove`	Remove the object
`commit`	Commit local changes done to an object to the cluster management infrastructure
`refresh`	Undo local changes done to the object
`list`	List all objects at current level

...*continues*

[2] A more precise synopsis is:
`[<mode>] <cmd> [<arg> ... ] [; ... ; [<mode>] <cmd> [<arg> ... ]]`

...continued

Command	Description
`sort`	Sort the order of display for the `list` command
`format`	Set formatting preferences for `list` output
`foreach`	Execute a set of commands on several objects
`show`	Display all properties of the object
`swap`	Swap (exchange) the names of two objects
`get`	Display specified property of the object
`set`	Set a specified property of the object
`clear`	Set default value for a specified property of the object.
`append`	Append a value to a property of the object, for a multi-valued property
`removefrom`	Remove a value from a specific property of the object, for a multi-valued property
`modified`	List objects with uncommitted local changes
`usedby`	List objects that depend on the object
`validate`	Do a validation check on the properties of the object

Working with objects with these commands is demonstrated with several examples in this section.

Working With Objects: `use`

Example

```
[mycluster->device]% use node001
[mycluster->device[node001]]% status
node001 ............. [   UP   ]
[mycluster->device[node001]]% exit
[mycluster->device]%
```

In the preceding example, `use node001` issued from within `device` mode makes `node001` the *current object*. The prompt changes accordingly. The `status` command, without an argument, then returns status information just for `node001`, because making an object the current object makes all subsequent commands apply only to that object. Finally, the `exit` command unsets the current object.

Working With Objects: `add`, `commit`

Example

```
[mycluster->device]% add physicalnode node100 10.141.0.100
[mycluster->device*[node100*]]% category add test-category
[mycluster->category*[test-category*]]% device; use node100
[mycluster->device*[node100*]]% set category test-category
[mycluster->device*[node100*]]% commit
[mycluster->device[node100]]% exit
[mycluster->device]%
```

In the preceding example, within `device` mode, a new object `node100` is added, of type `physicalnode`, and with IP address

10.141.0.100. The category test-category is then added, and the test-category object level within category mode is automatically dropped into when the command is executed. This is usually convenient, but not in this example, where it is assumed a property still needs to be set at the device node object level. To return to device mode again, at the level it was left, a multiple command "device; use node100" is executed. The multiple command can actually be run here with the ";" character removed, because the use command implies the existence of the ";" character.

The category property of the node100 object is set to the newly created category test-category and the object is then committed to store it permanently. Until the newly added object has been committed, it remains a local change that is lost when cmsh is exited.

The commit command by default does not wait for the state change to complete. This means that the prompt becomes available right away. This means that it is not obvious that the change has taken place, which could be awkward if scripting with cmsh for cloning (discussed shortly) a software image (section 3.1.2). The -w|--wait option to the commit command works around this issue by waiting for the cloning of the software image to be completed before making the prompt available.

Asterisk tags in the prompt are a useful reminder of a modified state, with each asterisk indicating a tagged object that has an unsaved, modified property.

In most modes the add command takes only one argument, namely the name of the object that is to be created. However, in device mode an extra object-type, in this case physicalnode, is also required as argument, and an optional extra IP argument may also be specified. The response to "help add" while in device mode gives details:

```
[myheadnode->device]% help add
Name:
 dd - Create a new device of the given type with specified hostname

Usage:
 add <type> <hostname>
 add <type> <hostname> <ip>

Arguments:
 type
    physicalnode, virtualsmpnode, headnode, ethernetswitch,
    ibswitch, myrinetswitch, powerdistributionunit, genericdevice,
    racksensor, chassis, gpuunit
```

Working With Objects: clone, modified, remove, swap

Continuing on with the node object node100 that was created in the previous example, it can be cloned to node101 as follows:

Example

```
[mycluster->device]% clone node100 node101
Warning: The Ethernet switch settings were not cloned, and have to
be set manually
[mycluster->device*[node101*]]% exit
[mycluster->device*]% modified
```

```
State  Type                     Name
------ ------------------------ -----------------------------------
+      Cloned                   node101
[mycluster->device*]% commit
[mycluster->device]%
[mycluster->device]% remove node100
[mycluster->device*]% commit
[mycluster->device]%
```

The `modified` command is used to check what objects have uncommitted changes, and the new object `node101` that is seen to be modified, is saved with a `commit`. The device `node100` is then removed by using the `remove` command. A `commit` executes the removal.

The `modified` command corresponds roughly to the functionality of the `List of Changes` menu option under the `View` menu of `cmgui`'s main menu bar.

The "+" entry in the `State` column in the output of the `modified` command in the preceding example indicates the object is a newly added one, but not yet committed. Similarly, a "-" entry indicates an object that is to be removed on committing, while a blank entry indicates that the object has been modified without an addition or removal involved.

Cloning an object is a convenient method of duplicating a fully configured object. When duplicating a device object, `cmsh` will attempt to automatically assign a new IP address using a number of heuristics. In the preceding example, `node101` is assigned IP address `10.141.0.101`.

Sometimes an object may have been misnamed, or physically swapped. For example, `node001` exchanged physically with `node002` in the rack, or the hardware device `eth0` is misnamed by the kernel and should be `eth1`. In that case it can be convenient to simply swap their names via the cluster manager front end rather than change the physical device or adjust kernel configurations. This is equivalent to exchanging all the attributes from one name to the other.

For example, if the two interfaces on the head node need to have their names exchanged, it can be done as follows:

```
[mycluster->device]% use mycluster
[mycluster->device[mycluster]]% interfaces
[mycluster->device[mycluster]->interfaces]% list
Type         Network device name  IP                Network
------------ -------------------- ---------------- --------------
physical     eth0 [dhcp]          10.150.4.46      externalnet
physical     eth1 [prov]          10.141.255.254   internalnet
[ddnmon61->device[mycluster]->interfaces]% swap eth0 eth1; commit
[ddnmon61->device[mycluster]->interfaces]% list
Type         Network device name  IP                Network
------------ -------------------- ---------------- --------------
physical     eth0 [prov]          10.141.255.254   internalnet
physical     eth1 [dhcp]          10.150.4.46      externalnet
[mycluster->device[mycluster]->interfaces]% exit; exit
```

Working With Objects: `get`, `set`, `refresh`

The `get` command is used to retrieve a specified property from an object, and `set` is used to set it:

Example

```
[mycluster->device]% use node101
[mycluster->device[node101]]% get category
test-category
[mycluster->device[node101]]% set category default
[mycluster->device*[node101*]]% get category
default
[mycluster->device*[node101*]]% modified
State  Type                     Name
------ ------------------------ -------------------------------
       Device                   node101
[mycluster->device*[node101*]]% refresh
[mycluster->device[node101]]% modified
No modified objects of type device
[mycluster->device[node101]]% get category
test-category
[mycluster->device[node101]]%
```

Here, the `category` property of the `node101` object is retrieved by using the `get` command. The property is then changed using the `set` command. Using `get` confirms that the value of the property has changed, and the `modified` command reconfirms that `node101` has local uncommitted changes. The `refresh` command undoes the changes made, and the `modified` command confirms that no local changes exist. Finally the `get` command reconfirms that no local changes exist.

A string can be set as a revision label for any object:

Example

```
[mycluster->device[node101]]% set revision "changed on 10th May"
[mycluster->device*[node101*]]% get revision
[mycluster->device*[node101*]]% changed on 10th May 2011
```

This can be useful when using shell scripts with an input text to label and track revisions when sending commands to `cmsh`. How to send commands from the shell to `cmsh` is introduced in section 3.5.1.

Some properties are Booleans. For these, the values "`yes`", "`1`", "`on`" and "`true`" are equivalent to each other, as are their opposites "`no`", "`0`", "`off`" and "`false`". These values are case-insensitive.

Working With Objects: `clear`

Example

```
[mycluster->device]% set node101 mac 00:11:22:33:44:55
[mycluster->device*]% get node101 mac
00:11:22:33:44:55
[mycluster->device*]% clear node101 mac
[mycluster->device*]% get node101 mac
00:00:00:00:00:00
[mycluster->device*]%
```

The `get` and `set` commands are used to view and set the MAC address of `node101` without running the `use` command to make `node101` the *current object*. The `clear` command then unsets the value of the property. The result of `clear` depends on the type of the property it acts on. In the case of string properties, the empty string is assigned, whereas for MAC addresses the special value `00:00:00:00:00:00` is assigned.

Working With Objects: `list`, `format`, `sort`

The `list` command is used to list all `device` objects. The `-f` flag takes a format string as argument. The string specifies what properties are printed for each object, and how many characters are used to display each property in the output line. In following example a list of objects is requested, displaying the `hostname`, `ethernetswitch` and `ip` properties for each object.

Example

```
[ddnmon61->device]% list -f hostname:14,ethernetswitch:15,ip
hostname (key) ethernetswitch  ip
-------------- --------------- --------------------
apc01                          10.142.254.1
ddnmon61       switch01:46     10.142.255.254
node001        switch01:47     10.142.0.1
node002        switch01:45     10.142.0.2
switch01                       10.142.253.1
[ddnmon61->device]%
```

Running the `list` command with no argument uses the current format string for the mode.

Running the `format` command without arguments displays the current format string, and also displays all available properties including a description of each property. For example (output truncated):

Example

```
[ddnmon61->device]% format
Current list printing format:
-----------------------------
hostname:[10-14]

Valid fields:
-------------
Type                 : The type of the device
activation           : Date on which node was defined
banks                : Number of banks
bmcpassword          : Password used to send ipmi/ilo commands
bmcusername          : Username used to send ipmi/ilo commands
burnconfig           : Burning configuration
...
```

To change the current format string, a new format string can be passed as an argument to `format`.

The print specification of the `format` command uses the delimiter ":" to separate the parameter and the value for the width of the parameter column. For example, a width of 10 can be set with:

Example

```
[ddnmon61->device]% format hostname:10
[ddnmon61->device]% list
hostname (
----------
apc01
```

```
ddnmon61
node001
node002
switch01
```

A range of widths can be set, from a minimum to a maximum, using square brackets. A single minimum width possible is chosen from the range that fits all the characters of the column. If the number of characters in the column exceeds the maximum, then the maximum value is chosen. For example:

Example

```
[ddnmon61->device]% format hostname:[10-14]
[ddnmon61->device]% list
hostname (key)
--------------
apc01
ddnmon61
node001
node002
switch01
```

The parameters to be viewed can be chosen from a list of valid fields by running the format command without any options, as shown earlier.

Multiple parameters can be specified as a comma-separated list (with a colon-delimited width specification for each parameter). For example:

Example

```
[ddnmon61->device]% format hostname:[10-14],ethernetswitch:14,ip:20
[ddnmon61->device]% list
hostname (key) ethernetswitch  ip
-------------- -------------- --------------------
apc01                          10.142.254.1
ddnmon61       switch01:46     10.142.255.254
node001        switch01:47     10.142.0.1
node002        switch01:45     10.142.0.2
switch01                       10.142.253.1
```

The default parameter settings can be restored with the `-r|-reset` option:

Example

```
[ddnmon61->device]% format -r
[ddnmon61->device]% format | head -3
Current list printing format:
-----------------------------
type:22, hostname:[16-32], mac:18, category:16, ip:15, network:14,\
powerdistributionunits:16
[ddnmon61->device]%
```

The `sort` command sets the alphabetical sort order for the output of the `list` according precedence of the parameters specified.

Example

```
[ddnmon61->device]% sort type mac
[ddnmon61->device]% list -f type:15,hostname:15,mac
type            hostname (key)  mac
--------------- --------------- --------------------
HeadNode        ddnmon61        08:0A:27:BA:B9:43
PhysicalNode    node002         00:00:00:00:00:00
PhysicalNode    log001          52:54:00:DE:E3:6B
[ddnmon61->device]% sort type hostname
[ddnmon61->device]% list -f type:15,hostname:15,mac
type            hostname (key)  mac
--------------- --------------- --------------------
HeadNode        ddnmon61        08:0A:27:BA:B9:43
PhysicalNode    log001          52:54:00:DE:E3:6B
PhysicalNode    node002         00:00:00:00:00:00

[ddnmon61->device]% sort mac hostname
[ddnmon61->device]% list -f type:15,hostname:15,mac
type            hostname (key)  mac
--------------- --------------- --------------------
PhysicalNode    node002         00:00:00:00:00:00
HeadNode        ddnmon61        08:0A:27:BA:B9:43
PhysicalNode    log001          52:54:00:DE:E3:6B
```

The preceding `sort` commands can alternatively be specified with the `-s|--sort` option to the list command:

```
[ddnmon61->device]% list -f type:15,hostname:15,mac --sort type,mac
[ddnmon61->device]% list -f type:15,hostname:15,mac --sort type,hostname
[ddnmon61->device]% list -f type:15,hostname:15,mac --sort mac,hostname
```

Working With Objects: `append`, `removefrom`
When dealing with a property of an object that can take more than one value at a time—a list of values—the `append` and `removefrom` commands can be used to respectively append to and remove elements from the list. However, the `set` command may also be used to assign a new list at once. In the following example values are appended and removed from the `powerdistributionunits` properties of device `node001`. The `powerdistributionunits` properties represents the list of ports on power distribution units that a particular device is connected to. This information is relevant when power operations are performed on a node. Chapter 5 has more information on power settings and operations.

Example

```
[mycluster->device]% use node001
[mycluster->device[node001]]% get powerdistributionunits
apc01:1
[...device[node001]]% append powerdistributionunits apc01:5
[...device*[node001*]]% get powerdistributionunits
apc01:1 apc01:5
[...device*[node001*]]% append powerdistributionunits apc01:6
[...device*[node001*]]% get powerdistributionunits
apc01:1 apc01:5 apc01:6
[...device*[node001*]]% removefrom powerdistributionunits apc01:5
```

```
[...device*[node001*]]% get powerdistributionunits
apc01:1 apc01:6
[...device*[node001*]]% set powerdistributionunits apc01:1 apc01:02
[...device*[node001*]]% get powerdistributionunits
apc01:1 apc01:2
```

Working With Objects: `usedby`
Removing a specific object is only possible if other objects do not have references to it. To help the administrator discover a list of objects that depend on ("use") the specified object, the `usedby` command may be used. In the following example, objects depending on device `apc01` are requested. The `usedby` property of `powerdistributionunits` indicates that device objects `node001` and `node002` contain references to ("use") the object `apc01`. In addition, the `apc01` device is itself displayed as being in the `up` state, indicating a dependency of `apc01` on itself. If the device is to be removed, then the 2 references to it first need to be removed, and the device also first has to be brought to the `closed` state by using the `close` command.

Example

```
[mycluster->device]% usedby apc01
Device used by the following:
Type             Name       Parameter
---------------- ---------- ----------------------
Device           apc01      Device is up
Device           node001    powerDistributionUnits
Device           node002    powerDistributionUnits
[mycluster->device]%
```

Working With Objects: `validate`
Whenever committing changes to an object, the cluster management infrastructure checks the object to be committed for consistency. If one or more consistency requirements are not met, then `cmsh` reports the violations that must be resolved before the changes are committed. The `validate` command allows an object to be checked for consistency without committing local changes.

Example

```
[mycluster->device]% use node001
[mycluster->device[node001]]% clear category
[mycluster->device*[node001*]]% commit
Code  Field                    Message
----- ------------------------ ---------------------------
1     category                 The category should be set
[mycluster->device*[node001*]]% set category default
[mycluster->device*[node001*]]% validate
All good
[mycluster->device*[node001*]]% commit
[mycluster->device[node001]]%
```

Working With Objects: `show`
The `show` command is used to show the parameters and values of a specific object. For example for the object `node001`, the attributes displayed are (some output elided):

```
[mycluster->device[node001]]% show
Parameter                   Value
--------------------------- ------------------------------------
Activation                  Thu, 08 Aug 2013 16:57:38 CEST
Burn config                 <0 bytes>
Burning                     no
Category                    default
...
Software image              default-image
Start new burn              no
Type                        PhysicalNode
Userdefined1
Userdefined2
```

3.5.4 Accessing Cluster Settings

The management infrastructure of DirectMon is designed to allow cluster partitioning in the future. A cluster partition can be viewed as a virtual cluster inside a real cluster. The cluster partition behaves as a separate cluster while making use of the resources of the real cluster in which it is contained. Although cluster partitioning is not yet possible in the current version of DirectMon, its design implications do decide how some global cluster properties are accessed through `cmsh`.

In `cmsh` there is a `partition` mode which will, in a future version, allow an administrator to create and configure cluster partitions. Currently, there is only one fixed partition, called `base`. The `base` partition represents the physical cluster as a whole and cannot be removed. A number of properties global to the cluster exist inside the `base` partition. These properties are referenced and explained in remaining parts of this manual.

Example

```
[root@myheadnode ~]# cmsh
[myheadnode]% partition use base
[myheadnode->partition[base]]% show
Parameter                       Value
------------------------------- ------------------------------------------------
Administrator e-mail
BMC Password                    *********
BMC User ID                     4
BMC User name                   bright
Burn configs                    <2 in submode>
Cluster name                    My Cluster
Default burn configuration      default
Default category                default
Default software image          default-image
External network                externalnet
Failover                        not defined
Management network              internalnet
Headnode                        myheadnode
Name                            base
Name servers                    192.168.101.1
Node basename                   node
Node digits                     3
Notes                           <0 bytes>
```

```
Revision
Search domains                 clustervision.com
Time servers                   pool.ntp.org
Time zone                      America/Los_Angeles
```

3.5.5 Advanced `cmsh` Features

This section describes some advanced features of `cmsh` and may be skipped on first reading.

Command Line Editing

Command line editing and history features from the `readline` library are available. See `http://tiswww.case.edu/php/chet/readline/rluserman.html` for a full list of key-bindings.

The most useful features provided by `readline` are tab-completion of commands and arguments, and command history using the arrow keys.

Mixing `cmsh` And Unix Shell Commands

Occasionally it can be useful to be able to execute commands from Unix shells while performing cluster management. For this reason, `cmsh` allows users to execute Unix commands by prefixing the command with a "`!`" character:

Example

```
[mycluster]% !hostname -f
mycluster.cm.cluster
[mycluster]%
```

Executing the `!` command by itself will start an interactive login sub-shell. By exiting the sub-shell, the user will return to the `cmsh` prompt.

Besides simply executing commands from within `cmsh`, the output of operating system shell commands can also be used within `cmsh`. This is done by using the "backtick syntax" available in most Unix shells.

Example

```
[mycluster]% device use `hostname`
[mycluster->device[mycluster]]% status
mycluster ............... [   UP   ]
[mycluster->device[mycluster]]%
```

Output Redirection

Similar to Unix shells, `cmsh` also supports output redirection to the shell through common operators such as >, >> and `|`.

Example

```
[mycluster]% device list > devices
[mycluster]% device status >> devices
[mycluster]% device list | grep node001
Type           Hostname (key)     MAC (key)           Category
-------------- --------------     ------------------- ----------
PhysicalNode   node001            00:E0:81:2E:F7:96   default
```

Looping Over Objects With `foreach`

It is frequently convenient to be able to execute a `cmsh` command on several objects at once. The `foreach` command is available in a number of `cmsh` modes for this purpose. A `foreach` command takes a list of space-separated object names (the keys of the object) and a list of commands that must be enclosed by parentheses, i.e.: "(" and ")". The `foreach` command will then iterate through the objects, executing the list of commands on the iterated object each iteration.

Basic syntax for the `foreach` command: The basic `foreach` syntax is:

`foreach` *<obj>* · · · *<obj>* `(` *<cmd>*`;` · · · `;` *<cmd>* `)`

Example

```
[mycluster->device]% foreach node001 node002 (get hostname; status)
node001
node001 ............. [   UP   ]
node002
node002 ............. [   UP   ]
[mycluster->device]%
```

With the `foreach` command it is possible to perform `set` commands on groups of objects simultaneously, or to perform an operation on a group of objects.

Advanced options for the `foreach` command: The `foreach` command advanced options can be viewed from the help page:

```
[root@ddnmon61 ~]# cmsh -c "device help foreach"
```

The options can be classed as: grouping options, adding options, conditional options, and looping options.

- **Grouping options:** `--nodes`, `--group`, `--category`, `--rack`, `--chassis`

  In the `device` mode of `cmsh`, for example, the `foreach` command has grouping options for selecting devices. These options can be used to select the nodes to loop over, instead of passing a list of objects to `foreach` directly. For example, the `--category` (`-c`) option takes a node category argument, while the `--node` (`-n`) option takes a node-list argument. Node-lists (specification on page 73) can use the following syntax:
  <node>`,...,`*<node>*`,`*<node>*`..`*<node>*

  Example

  ```
  [demo->device]% foreach -c default (status)
  node001 ............. [   DOWN   ]
  node002 ............. [   DOWN   ]
  [demo->device]% foreach -g rack8 (status)
  ...
  [demo->device]% foreach -n node001,node008..node016,node032 (status)
  ...
  [demo->device]%
  ```

- **Adding options:** `--clone`, `--add`
  The `--clone` (`-o`) option allows the cloning (section 3.5.3) of objects in a loop. In the following example, from `device` mode, `node001` is used as the base object from which other nodes from `node022` up to `node024` are cloned:

  Example

  ```
  [ddnmon61->device]% foreach --clone node001 -n node022..node024 ()
  [ddnmon61->device*]% list | grep node
  Type           Hostname (key)  Ip
  ------------   --------------  -----------
  PhysicalNode   node001         10.141.0.1
  PhysicalNode   node022         10.141.0.22
  PhysicalNode   node023         10.141.0.23
  PhysicalNode   node024         10.141.0.24
  [ddnmon61->device*]% commit
  ```

  To avoid possible confusion: the cloned objects are merely objects (placeholder schematics and settings, with some different values for some of the settings, such as IP addresses, decided by heuristics). So it is explicitly not the software disk image of `node001` that is duplicated by object cloning to the other nodes.

  The `--add` (`-a`) option creates the device for a specified device type, if it does not exist. Valid types are shown in the help output, and include `physicalnode`, `headnode`, `ibswitch`.

- **Conditional options:** `--status`, `--quitonunknown`
  The `--status` (`-s`) option allows nodes to be filtered by the device status (section 3.1.1).

  Example

  ```
  [ddnmon61->device]% foreach -n node001..node004 --status UP (get IP)
  10.141.0.1
  10.141.0.3
  ```

  The `--quitonunknown` (`-q`) option allows the `foreach` loop to be exited when an unknown command is detected.

- **Looping options:** `*`, `--verbose`
  The wildcard character `*` with `foreach` implies all the objects that the `list` command lists for that mode. It is used without grouping options:

  Example

  ```
  [myheadnode->device]% foreach * (get ip; status)
  10.141.253.1
  switch01 ............ [  DOWN  ]
  10.141.255.254
  myheadnode .......... [   UP   ]
  10.141.0.1
  ```

```
node001 ............. [ CLOSED ]
10.141.0.2
node002 ............. [ CLOSED ]
[myheadnode->device]%
```

The `--verbose` (`-v`) option displays the loop headers during a running loop with time stamps, which can help in debugging.

Node List Syntax

Node list specifications, as used in the `foreach` specification and elsewhere, can be of several types. These types are best explained with node list specification examples:

- **adhoc (with a comma):**
  example: `node001,node003,node005,node006`
- **sequential (with two dots or square brackets):**
  example: `node001..node004`
  or, equvalently: `node00[1-4]`
  which is: `node001,node002,node003,node004`
- **rack-based:**
  This is intended to hint which rack a node is located in. Thus:
  - example: `r[1-2]n[01-03]`
    which is: `r1n01,r1n02,r1n03,r2n01,r2n02,r2n03`
    This might hint at two racks, r1 and r2, with 3 nodes each.
  - example: `rack[1-2]node0[1-3]`
    which is: `rack1node01,rack1node02,rack1node03,rack2node01,` `rack2node02,rack2node03`
    Essentially the same as the previous one, but for nodes that were named more verbosely.
- **sequential exclusion (negation):**
  example: `node001..node005,-node002..node003`
  which is: `node001,node004,node005`
- **sequential stride (every <*stride*> steps):**
  example: `node00[1..7]:2`
  which is: `node001,node003,node005,node007`
- **mixed list:**
  The square brackets and the two dots input specification cannot be used at the same time in one argument. Other than this, specifications can be mixed:
  - example: `r1n001..r1n003,r2n003`
    which is: `r1n001,r1n002,r1n003,r2n003`
  - example: `r2n003,r[3-5]n0[01-03]`
    which is: `r2n003,r3n001,r3n002,r3n003,r4n001,r4n002` `r4n003,r5n001,r5n002,r5n003`
  - example: `node[001-100],-node[004-100:4]`
    which is: every node in the 100 nodes, except for every fourth node.

Setting grouping syntax with the `groupingsyntax` command: "Grouping syntax" here refers to usage of dots and square brackets. In other words, it is syntax of how a grouping is marked so that it is accepted as a list. The list that is specified in this manner can be for input or output purposes.

The `groupingsyntax` command sets the grouping syntax using the following options:

- `bracket`: the square brackets specification.
- `dot`: the two dots specification.
- `auto`: the default. Setting `auto` means that:
  - either the `dot` or the `bracket` specification are accepted as input,
  - the `dot` specification is used for output.

The chosen `groupingsyntax` option can be made persistent by adding it to the `.cm/cmsh/.cmshrc` file (section 3.5.1).

Example

```
[root@ddnmon61 ~]# cat .cm/cmsh/.cmshrc
groupingsyntax auto
```

3.6 Cluster Management Daemon

The *cluster management daemon* or *CMDaemon* is a server process that runs on all nodes of the cluster (including the head node). The cluster management daemons work together to make the cluster manageable. When applications such as `cmsh` and `cmgui` communicate with the cluster management infrastructure, they are actually interacting with the cluster management daemon running on the head node. Cluster management applications never communicate directly with cluster management daemons running on non-head nodes.

The CMDaemon application starts running on any node automatically when it boots, and the application continues running until the node shuts down. Should CMDaemon be stopped manually for whatever reason, its cluster management functionality becomes unavailable, making it hard for administrators to manage the cluster. However, even with the daemon stopped, the cluster remains fully usable for running computational jobs using a workload manager.

The only route of communication with the cluster management daemon is through TCP port `8081`. The cluster management daemon accepts only SSL connections, thereby ensuring all communications are encrypted. Authentication is also handled in the SSL layer using client-side `X509v3` certificates (section 3.3).

On the head node, the cluster management daemon uses a MySQL database server to store all of its internal data. Monitoring data is also stored in a MySQL database.

3.6.1 Controlling The Cluster Management Daemon

It may be useful to shut down or restart the cluster management daemon. For instance, a restart may be necessary to activate changes when the cluster management daemon configuration file is modified. The cluster management daemon operation can be controlled through the following init script arguments to `/etc/init.d/cmd`:

Init Script Operation	Description
`stop`	stop the cluster management daemon
`start`	start the cluster management daemon
`restart`	restart the cluster management daemon
`status`	report whether cluster management daemon is running
`full-status`	report detailed statistics about cluster management daemon
`upgrade`	update database schema after version upgrade (*expert only*)
`debugon`	enable debug logging (*expert only*)
`debugoff`	disable debug logging (*expert only*)

Example

Restarting the cluster management daemon on the head node of a cluster:

```
[root@mycluster ~]# /etc/init.d/cmd restart
Waiting for CMDaemon (2916) to terminate...
                                                        [  OK  ]
Waiting for CMDaemon to start...                        [  OK  ]
[root@mycluster ~]#
```

Example

Viewing the resources used by CMDaemon, and some other useful information:

```
[root@ddnmon61 etc]# service cmd full-status
   CMDaemon version 1.5 is running (active).

   Current Time: Fri, 23 Aug 2013 17:35:51 (CEST)
   Startup Time: Fri, 23 Aug 2013 16:21:07 (CEST)
   Uptime: 1 hour, 14 minutes, 44 seconds

   CPU Usage: 12.832u 10.2984s (0.5%)
   Memory Usage: 87.3MB

   Sessions Since Startup: 30
   Active Sessions: 2

   Number of occupied worker-threads: 1
   Number of free worker-threads: 14

   Connections handled: 973
   Requests processed: 973
```

```
Total read: 613.5KB
Total written: 2.5MB

Average request rate: 0.216994 requests/s
Average bandwidth usage: 585.0B/s
```

Example

Restarting the cluster management daemon on a sequence of regular nodes, `node001` to `node040`, of a cluster:

```
[root@mycluster ~]# pexec -b -n=node001..node040 /etc/init.d/cmd restart
```

This uses `pexec`, the parallel shell command (section 12.1).

3.6.2 Configuring The Cluster Management Daemon

Some cluster configuration changes can be done by modifying the cluster management daemon configuration file. For the head node, this is located at:

```
/cm/local/apps/cmd/etc/cmd.conf
```

For regular nodes, it is located inside of the software image that the node uses.

Appendix C describes all recognized configuration file directives and how they can be used. Normally there is no need to modify the default settings.

After modifying the configuration file, the cluster management daemon must be restarted to activate the changes.

3.6.3 Configuring The Cluster Management Daemon Logging Facilities

CMDaemon generates log messages from specific internal subsystems, such as Workload Management, Service Management, Monitoring, Certs, and so on. By default, none of those subsystems generate detailed (debug-level) messages, as that would make the log file grow rapidly.

CMDaemon Global Debug Mode

It is possible to enable a global debug mode in CMDaemon as one of its status options (some text elided):

Example

```
[root@ddnmon61 etc]# service cmd help
Usage: /etc/init.d/cmd start|stop|...|debugon|debugoff|logconf
[root@ddnmon61 etc]# service cmd status
cmd (pid  1116) is running...
[root@ddnmon61 etc]# service cmd debugon
CMDaemon debug level on
```

Stopping debug level logs from running for too long by executing `service cmd debugoff` is a good idea, especially for production clusters. This is important in order to prevent swamping the cluster with unfeasibly large logs.

CMDaemon Subsystem Debug Mode

CMDaemon subsystems can generate debug logs separately per subsystem, including by severity level. This can be done by modifying the logging configuration file at:

```
/cm/local/apps/cmd/etc/logging.cmd.conf
```

For example, to set only CMDaemon debug output for Monitoring, at a severity level of `warning`, the file contents might look like:

Example

```
Severity {
   warning: MON
}
```

The `logconf` option to `cmd` reloads the logging configuration and makes the changes take effect without having to restart CMDaemon:

Example

```
[root@ddnmon61 etc]# service cmd logconf
```

Further details on setting debug options are given within the `logging.cmd.conf` file.

3.6.4 Configuration File Modification

As part of its tasks, the cluster management daemon modifies a number of system configuration files. Some configuration files are completely replaced, while other configuration files only have some sections modified. Appendix A lists all system configuration files that are modfied.

A file that has been generated entirely by the cluster management daemon contains a header:

```
# This file was automatically generated by cmd. Do not edit manually!
```

Such a file will be entirely overwritten, unless the `FrozenFile` configuration file directive (Appendix C) is used to keep it frozen.

Sections of files that have been generated by the cluster management daemon will read as follows:

```
# This section of this file was automatically generated by cmd. Do not \
edit manually!
# BEGIN AUTOGENERATED SECTION -- DO NOT REMOVE
...
# END AUTOGENERATED SECTION   -- DO NOT REMOVE
```

Such a file will have the auto-generated sections entirely overwritten, unless the `FrozenFile` configuration file directive is used to keep these sections frozen.

The `FrozenFile` configuration file directive in `cmd.conf` is set as suggested by this example:

Example

```
FrozenFile = { "/etc/dhcpd.conf", "/etc/postfix/main.cf" }
```

If the generated file or section of a file has a manually modified part, and when not using `FrozenFile`, then during overwriting an event is generated, and the manually modified configuration file is backed up to:

```
/var/spool/cmd/saved-config-files
```

Using `FrozenFile` can be regarded as a configuration technique (section 4.16.3), and one of various possible configuration techniques (section 4.16.1).

3.6.5 Configuration File Conflicts Between The Standard Distribution And DirectMon For Generated And Non-Generated Files

While DirectMon changes as little as possible of the standard distributions that it manages, there can sometimes be unavoidable issues. In particular, sometimes a standard distribution utility or service generates a configuration file that conflicts with what the configuration file generated by DirectMon carries out (Appendix A).

For example, the Red Hat security configuration tool `system-config-securitylevel` can conflict with what `shorewall` (section 13.2) does, while the Red Hat Authentication Configuration Tool `authconfig` (used to modify the `/etc/pam.d/system-auth` file) can conflict with the configuration settings set by DirectMon for LDAP and PAM.

In such a case the configuration file generated by DirectMon must be given precedence, and the generation of a configuration file from the standard distribution should be avoided. Sometimes using a fully or partially frozen configuration file (section 3.6.4) allows a workaround. Otherwise, the functionality of the DirectMon version usually allows the required configuration function to be implemented.

Details on the configuration files installed and updated by the package management system, for files that are “non-generated” (that is, not of the kind in section 3.6.4 or in the lists in Appendixes A.1 A.2 and A.3), are given in Appendix A.4.

4 Configuring The Cluster

After DirectMon software has been installed on the head node, the cluster must be configured. For convenience, the regular nodes on the cluster use a default software image stored on the head node. The image is supplied to the regular nodes during a process called provisioning (Chapter 6), and both the head node and the regular nodes can have their software modified to suit exact requirements (Chapter 10). This chapter however goes through a number of basic cluster configuration aspects that are required to get all the hardware up and running on the head and regular nodes.

Section 4.1 explains how a DirectMon license is viewed, verified, requested, and installed, thereby allowing CMDaemon to run. CMDaemon-based configuration of the cluster is covered in sections 4.2 to section 4.15:

Section 4.2 explains how some of the main cluster configuration settings can be changed.

Section 4.3 details how the internal and external network parameters of the cluster can be changed.

Section 4.4 describes the setting up of network bridge interfaces.

Section 4.5 describes VLAN configuration.

Section 4.6 describes the setting up of network bond interfaces.

Section 4.7 covers how InfiniBand is set up.

Section 4.8 describes how Baseboard Management Controllers such as IPMI and iLO are set up.

Section 4.9 describes how switches are set up.

Section 4.10 explains how disk layouts are configured, as well as how diskless nodes are set up.

Section 4.11 describes how NFS volumes are exported from an NFS server and mounted to nodes using the integrated approach of DirectMon.

Section 4.12 describes how services can be run from DirectMon.

Section 4.13 describes how a rack can be configured and managed with DirectMon.

Section 4.14 describes how a GPU unit such as the Dell PowerEdge C410x can be configured with DirectMon.

Section 4.15 describes how custom scripts can replace some of the default scripts in use.

Section 4.16 discusses configuration alternatives that are not based on CMDaemon.

More elaborate aspects of cluster configuration such as power management, user management, package management, and workload management are covered in later chapters.

4.1 The Cluster License

Any DirectMon installation requires a *license file* to be present on the head node. The license file specifies the conditions under which a particular DirectMon installation has been licensed.

Example

- the "`Licensee`" details, which include the name of the organization, is an attribute of the license file that specifies the condition that only the specified organization may use the software
- the "`Licensed nodes`" attribute specifies the maximum number of nodes that the cluster manager may manage. Head nodes are regarded as nodes too for this attribute.
- the "`Expiration date`" of the license is an attribute that sets when the license expires. It is sometimes set to a date in the near future so that the cluster owner effectively has a trial period. A new license with a longer period can be requested (section 4.1.3) after the owner decides to continue using the cluster with DirectMon

A license file can only be used on the machine for which it has been generated and cannot be changed once it has been issued. This means that to change licensing conditions, a new license file must be issued.

The license file is sometimes referred to as the *cluster certificate*, or *head node certificate*, because it is the X509v3 certificate of the head node, and is used throughout cluster operations. Its components are located under `/cm/local/apps/cmd/etc/`. Section 3.3 has more information on certificate-based authentication.

4.1.1 Displaying License Attributes

Before starting the configuration of a cluster, it is important to verify that the attributes included in the license file have been assigned the correct values. The license file is installed in the following location:

```
/cm/local/apps/cmd/etc/cert.pem
```

and the associated private key file is in:

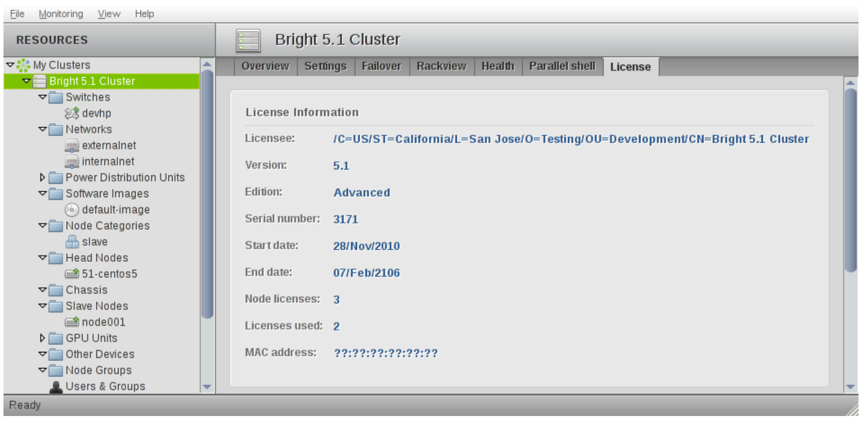
```
/cm/local/apps/cmd/etc/cert.key
```

To verify that the attributes of the license have been assigned the correct values, the `License` tab of the GUI can be used to display license details (figure 4.1):

Alternatively the `licenseinfo` in `cmsh main` mode may be used:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% main licenseinfo
License Information
```



Bright 5.1 Cluster

License Information

Licensee:	/C=US/ST=California/L=San Jose/O=Testing/OU=Development/CN=Bright 5.1 Cluster
Version:	5.1
Edition:	Advanced
Serial number:	3171
Start date:	28/Nov/2010
End date:	07/Feb/2106
Node licenses:	3
Licenses used:	2
MAC address:	??:??:??:??:??:??

Figure 4.1: License Information

```
----------------------- -------------------------------------------
Licensee                /C=US/ST=California/L=San Jose/O=DDN
                         Computing/OU=Temporary Licensing/CN=003040
Serial Number           4411
Start Time              Tue Apr 24 00:00:00 2012
End Time                Mon Dec 31 23:59:59 2012
Version
Edition                 Advanced
Pre-paid Nodes          10
Max Pay-per-use Nodes   1000
Node Count              2
MAC Address / Cloud ID  00:0C:29:E2:DA:2D
```

The license in the example above allows 1000 pay-per-use nodes to be used. It is tied to a specific MAC address, so it cannot be used anywhere. For convenience, the `Node Count` field in the output of `licenseinfo` shows the current number of nodes used.

4.1.2 Verifying A License—The `verify-license` Utility

The `verify-license` Utility Can Be Used When `licenseinfo` Cannot Be Used

Unlike the `licenseinfo` command in `cmsh` (section 4.1.1), the `verify-license` utility can check licenses even if the cluster management daemon is not running.

When an invalid license is used, the cluster management daemon cannot start. The license problem is logged in the cluster management daemon log file:

Example

```
[root@ddnmon61 ~]# /etc/init.d/cmd start
Waiting for CMDaemon to start...
CMDaemon failed to start please see log file.
[root@ddnmon61 ~]# tail -1 /var/log/cmdaemon
Dec 30 15:57:02 ddnmon61 CMDaemon: Fatal: License has expired
```

but further information cannot be obtained using `cmgui` and `cmsh`, because these clients themselves obtain their information from the cluster management daemon.

In such a case, the `verify-license` utility allows the troubleshooting of license issues.

Using The `verify-license` Utility To Troubleshoot License Issues

There are four ways in which the `verify-license` utility can be used:

1. Using `verify-license` with no options: simply displays a usage text:

Example

```
[root@ddnmon61 ~]# verify-license
Usage: /cm/local/apps/cmd/sbin/verify-license <path to certificate> \
<path to keyfile> <verify|info|monthsleft[=12]>
```

2. Using `verify-license` with the `info` option: prints license details:

Example

```
[root@ddnmon61 ~]# cd /cm/local/apps/cmd/etc/
[root@ddnmon61 etc]# verify-license cert.pem cert.key info
========= Certificate Information ===========
Version:
Edition:                    Advanced
Common name:                ddnmon61
Organization:               DDN
Organizational unit:        Development
Locality:                   San Jose
State:                      California
Country:                    US
Serial:                     4411
Starting date:              24 Apr 2012
Expiration date:            31 Dec 2012
MAC address:                00:0C:29:E2:DA:2D
Pre-paid nodes:             10
Max Pay-per-use Nodes:      1000
=============================================
[root@ddnmon61 etc]#
```

3. Using `verify-license` with the `verify` option: checks the validity of the license:

- If the license is valid, then no output is produced and the utility exits with exit-code `0`.
- If the license is invalid, then output is produced indicating what is wrong. Messages such as these are then displayed:
  - If the license is old:

    Example

    ```
    [root@ddnmon61 etc]# verify-license cert.pem cert.key verify
    License has expired
    License verification failed.
    ```

  - If the certificate is not from DDN:

    Example

```
        [root@ddnmon61 etc]# verify-license cert.pem cert.key verify
        Invalid license: This certificate was not signed by
        DDN
        License verification failed.
```

4. Using `verify-license` with the `monthsleft[=<`*`value`*`>]` option: If a number is set for `monthsleft` which is less than the number of months left until the license expires, then the `verify-license` command returns nothing. Otherwise the date of expiry for the license is displayed.

Example

```
[root@ddnmon61 etc]# date
Tue Mar 12 14:23:21 CET 2013
[root@ddnmon61 etc]# verify-license cert.{pem,key} monthsleft
DirectMon License expiration date: 31 Dec 2013
[root@ddnmon61 etc]# verify-license cert.{pem,key} monthsleft=9
[root@ddnmon61 etc]# verify-license cert.{pem,key} monthsleft=10
DirectMon License expiration date: 31 Dec 2013
```

4.1.3 Requesting And Installing A License Using A Product Key

The license file is introduced in section 4.1.

Is A License Needed?—Verifying License Attributes

Before installing a license, the license attributes should be verified (section 4.1.2) to check if installation is actually needed. If the attributes of the license are correct, the remaining parts of this section (4.1.3) may safely be skipped. Otherwise the *product key* (page 83) is used to install a license.

Incorrect license attributes will cause cluster configuration to fail or may lead to a misconfigured cluster. A misconfigured cluster may, for example, not have the ability to handle the full number of nodes. In particular, the license date should be checked to make sure that the license has not expired. If the license is invalid, and it should be valid according to the administrator, then the DDN reseller that provided the software should be contacted with details to correct matters.

If DirectMon is already running with a regular license, and if the license is due to expire, then reminders are sent to the administrator e-mail address (page 92).

The Product Key

The product key looks like: the following pattern of digits:

```
000354-515786-112224-207441-186713
```

The product key allows: the administrator:

- to obtain and activate a license, which allows the cluster to function
- to register the key using the DDN customer portal account. This allows cluster extension cloudbursting (section 7.2) to function.

The following terminology is used: when talking about product keys, locking, licenses, installation, and registration:

- **activating a license:** A product key is obtained from any DirectMon reseller. It is used to obtain and *activate* a license file. Activation means that DDN records that the product key has been used to obtain a license file. The license obtained by product key activation permits the cluster to work with particular settings. For example, the period of use and the number of nodes.
- **locking a product key:** The administrator is normally allowed to use a product key to activate a license only once. This is because a product key is *locked* on activation of the license. A locked state means that product key cannot activate a new license—it is "used up".

  An activated license only works on the hardware that the product key was used with.This could obviously be a problem if the administrator wants to move DirectMon to new hardware. In such a case, the locked state is revocable by sending a request to DDN to unlock the product key. A product key that is unlocked can be used once again to activate a new license.
- **license installation:** *License installation* occurs on the cluster after the license is activated and issued. The installation is done automatically if possible. Sometimes installation needs to be done manually, as explained in the section on the `request-license` script (page 84). The license can only work on the hardware it was specified for. After installation is complete, the cluster runs with the activated license.
- **product key registration:** *Product key registration* occurs on the customer portal account when the product key is associated with the account. Registered customers can view their cloud services billing information amongst other items.

Requesting A License With The `request-license` Script

If the license has expired, or if the license attributes are otherwise not correct, a new license file must be requested.

The request for a new license file is made using a product key (page 83) with the `request-license` command.

The `request-license` command is used to request and activate a license, and works most conveniently with a cluster that is able to access the internet. The request can also be made regardless of cluster connectivity to outside networks.

There are four options to use the product key to get the license:

1. **Direct WWW access:** If the cluster has access to the WWW port, then a successful completion of the `request-license` command obtains and activates the license. It also locks the product key.

   - **Proxy WWW access:** If the cluster uses a web-proxy, then the environment variable `http_proxy` must be set before the request-license command is run. From a bash prompt this is set with:

```
export http_proxy=<proxy>
```

where *<proxy>* is the hostname or IP address of the proxy. An equivalent alternative is that the ScriptEnvironment directive (page 594), which is a CMDaemon directive, can be set and activated (page 579).

2. **Off-cluster WWW access:** If the cluster does not have access to the WWW port, but the administrator does have off-cluster web-browser access, then the point at which the request-license command prompts "Submit certificate request to http://support.ddn.com/licensing/ ?" should be answered negatively. CSR (Certificate Sign Request) data generated is then conveniently displayed on the screen as well as saved in the file /cm/local/apps/cmd/etc/cert.csr.new. The cert.csr.new file may be taken off-cluster and processed with an off-cluster browser.

   The CSR file should not be confused with the private key file, cert.key.new, created shortly beforehand by the request-license command. In order to maintain cluster security, the private key file must, in principle, never leave the cluster.

   At the off-cluster web-browser, the administrator may enter the cert.csr.new content in a web form at:

   ```
   http://support.ddn.com/licensing
   ```

   A signed license text is returned. At DDN the license is noted as having been activated, and the product key is locked.

   The signed license text received by the administrator is in the form of a plain text certificate. As the web form response explains, it can be saved directly from most browsers. Cutting and pasting the text into an editor and then saving it is possible too, since the response is plain text. The signed license file, *<signedlicense>*, should then be put on the head node. If there is a copy of the file on the off-cluster machine, the administrator should consider wiping that copy in order to reduce information leakage.

   The command:

   ```
   install-license <signedlicense>
   ```

   installs the signed license on the head node, and is described further on page 86. Installation means the cluster now runs with the activated certificate.

3. **E-mail access:** If web access of any kind is denied to the administrator on- or off-cluster, but e-mail is allowed, then the procedure of option 2 can be followed partially.

   That is, instead of processing the cert.csr.new file with an off-cluster browser, the file is sent using an e-mail client to ca@ddn.com. The client can be on-cluster, if permitted, or it can be off-cluster.

As in option 2, the CSR file should not be confused with the private key file, `cert.key.new`, which must, in principle, never leave the cluster.

A signed certificate will be e-mailed back from the DirectMon License Desk. The license will be noted by DDN as having been activated, and the product key is put in a locked state. The signed certificate file, <*signedlicense*>, can then be handled further as described in option 2. That is, it is saved to the head node and installed with the `install-license` command.

4. **Fax or physical delivery:** If no internet access is available at all to the administrator, the CSR data may be faxed or sent as a physical delivery (postal mail print out, USB flash drive/floppy disk) to any DirectMon reseller. A certificate will be faxed or sent back in response, the license will be noted by DDN as having been activated, and the associated product key will be noted as being locked. The certificate can then be handled further as described in option 2.

Example

```
[root@ddnmon61 ~]# request-license
Product Key (XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX):
000354-515786-112224-207440-186713

Country Name (2 letter code): US
State or Province Name (full name): California
Locality Name (e.g. city): San Jose
Organization Name (e.g. company): DDN, Inc.
Organizational Unit Name (e.g. department): Development
Cluster Name: ddnmon61
Private key data saved to /cm/local/apps/cmd/etc/cert.key.new

MAC Address of primary head node (ddnmon61) for\
 eth0 [00:0C:29:87:B8:B3]:
Will this cluster use a high-availability setup\
 with 2 head nodes? [y/N] n

Certificate request data saved to /cm/local/apps/cmd/etc/cert.csr.new
Submit certificate request to http://support.ddn.com\
/licensing/ ?
[Y/n] y

Contacting http://support.ddn.com/licensing/...
License granted.
License data was saved to /cm/local/apps/cmd/etc/cert.pem.new
Install license ? [Y/n] n
Use "install-license /cm/local/apps/cmd/etc/cert.pem.new" to install\
 the license.
```

Installing A License With The `install-license` Script

Referring to the preceding example:

The administrator is prompted to enter the (internal network facing) MAC address.

After the certificate request is sent to DDN and approved, the license is granted.

If the prompt "`Install license?`" is answered with a "`Y`" (the default), the `install-license` script is run.

If the prompt is answered with an "`n`" then the `install-license` script must be run separately in order to complete installation of the license.

The `install-license` script takes the temporary location of the new license file generated by `request-license` as its argument, and installs related files on the head node. Running it completes the license installation on the head node.

Example

Assuming the new certificate is saved as `cert.pem.new`:

```
[root@ddnmon61 ~]# install-license /cm/local/apps/cmd/etc/cert.pem.new
========= Certificate Information ========
Version:
Edition:             Advanced
Common name:         ddnmon61
Organization:        DDN, Inc.
Organizational unit: Development
Locality:            San Jose
State:               California
Country:             US
Serial:              9463
Starting date:       23 Dec 2012
Expiration date:     31 Dec 2013
MAC address:         08:0A:27:BA:B9:43
Pre-paid nodes:          10
Max Pay-per-use Nodes:   1000
==========================================

Is the license information correct ? [Y/n] y

Installed new license

Restarting Cluster Manager Daemon to use new license: OK
```

Re-Installing A License For Upgrading From Single Head To High-Availability Configuration

A high availability (HA) cluster uses two head nodes so that if one fails the other can take over (Chapter 15).

HA is available only in the Advanced Edition of DirectMon. The Standard Edition can be upgraded to the Advanced Edition by requesting the DirectMon reseller for a new product key.

When originally installing HA using a Standard Edition product key, the cluster license is configured as a single head node during the installation covered in Chapter 2. After the cluster license is activated for the single head node, then to upgrade to an HA configuration requires a new product key, and requires activating a new license. To get a new product key the DDN reseller that issued the original key should be contacted. After the reseller has given the new key, the license can be replaced by running the `request-license` tool once more. During the rerun, the administrator is prompted to enter the (`internalnet`-facing) MAC address on the new second head node.

When originally installing HA using an Advanced Edition product key, the cluster is configured first as a single-headed cluster during the installation as covered in Chapter 2. After the cluster license is activated, no new product key and no new license is required to be able to install HA.

The `verify-license` utility can be run (section 4.1.2) to check the license. A cluster with two head nodes shows their MAC addresses separated by a `|` character.

Only with an HA license in place is it possible to complete the preparation stage of the head nodes for HA (section 15.2.1).

Re-Installing A License After Replacing The Hardware

If the head node hardware has changed, then a request should be put in to unlock the product key so that a new license can be requested and installed using the product key with the `request-license` script, followed by running the `install-license` script. The `install-license` script may not actually be needed, but it does no harm to run it just in case afterwards.

Re-Installing A License After Wiping Or Replacing The Hard Drive

If the head node hardware has otherwise not changed:

- The full drive image can be copied on to a blank drive and the system will work as before.
- Alternatively, if a new installation from scratch is done
  - then after the installation is done, a license can be requested and installed once more using the same product key, using the `request-license` command. Because the product key is normally locked when the previous license request was done, a request to unlock the product key usually needs to be sent to DDN before the license request can be executed.
  - If the administrator wants to avoid using the `request-license` command and having to type in a product key, then some certificate key pairs must be placed on the new drive from the old drive, in the same locations. The procedure that can be followed is:
    1. in the directory `/cm/local/apps/cmd/etc/`, the following key pair is copied over:
       * `cert.key`
       * `cert.pem`

       Copying these across means that `request-license` does not need to be used.
    2. The `admin.{pem|key}` key pair files can then be placed in the directory `/root/.cm/cmsh/`. Two options are:
       * the following key pair can be copied over:
         · `admin.key`
         · `admin.pem`

         or
       * a fresh `admin.{pem|key}` key pair can be generated instead via a `cmd -b` option:

Example

```
[root@ddnmon61 ~]# service cmd stop
[root@ddnmon61 ~]# cmd -b
[root@ddnmon61 ~]# [...]
 Tue Jan 21 11:47:54 [ CMD ]  Info: Created certificate in admin.pem
 Tue Jan 21 11:47:54 [ CMD ]  Info: Created certificate in admin.key
[root@ddnmon61 ~]# [...]
[root@ddnmon61 ~]# chmod 600 admin.*
[root@ddnmon61 ~]# mv admin.* /root/.cm/cmsh/
[root@ddnmon61 ~]# service cmd start
```

Rebooting Nodes After An Install

The first time a product key is used: After using a product key with the command `request-license` during a cluster installation, and then running `install-license`, a reboot is required of all nodes in order for them to pick up and install their new certificates (section 6.4.1). The `install-license` script has at this point already renewed the administrator certificates for use with `cmsh` and `cmgui` on the head node. The parallel execution command `pexec reboot` suggested towards the end of the `install-license` script output is what can be used to reboot all other nodes. Since such a command is best done by an administrator manually, `pexec reboot` is not scripted.

The subsequent times that the same product key is used: If a license has become invalid, a new license may be requested. On running the command `request-license` for the cluster, the administrator is prompted on whether to re-use the existing keys and settings from the existing license.

- If the existing keys are kept, a `pexec reboot` is not required. This is because these keys are X509v3 certificates issued from the head node. For these:
  - Any node certificates (section 6.4.1) that were generated using the old certificate are therefore still valid and so regenerating them for nodes via a reboot is not required, allowing users to continue working uninterrupted. On reboot new node certificates are generated and used if needed.
  - User certificates (section 8.4) also remain valid, but only while CMDaemon is not restarted. They become invalid in any case with a new license on boot since they do not regenerate automatically. It is therefore advised to install a permanent license as soon as possible, or alternatively, to not bother creating user certificates until a permanent license has been set up for the cluster.
- If the existing keys are not re-used, then node communication ceases until the nodes are rebooted. If there are jobs running on the DirectMon nodes, they cannot then complete.

After the license is installed, verifying the license attribute values is a good idea. This can be done using the `licenseinfo` command in `cmsh`, or the `License` tab in `cmgui`'s cluster resource tabbed pane (section 4.1.1).

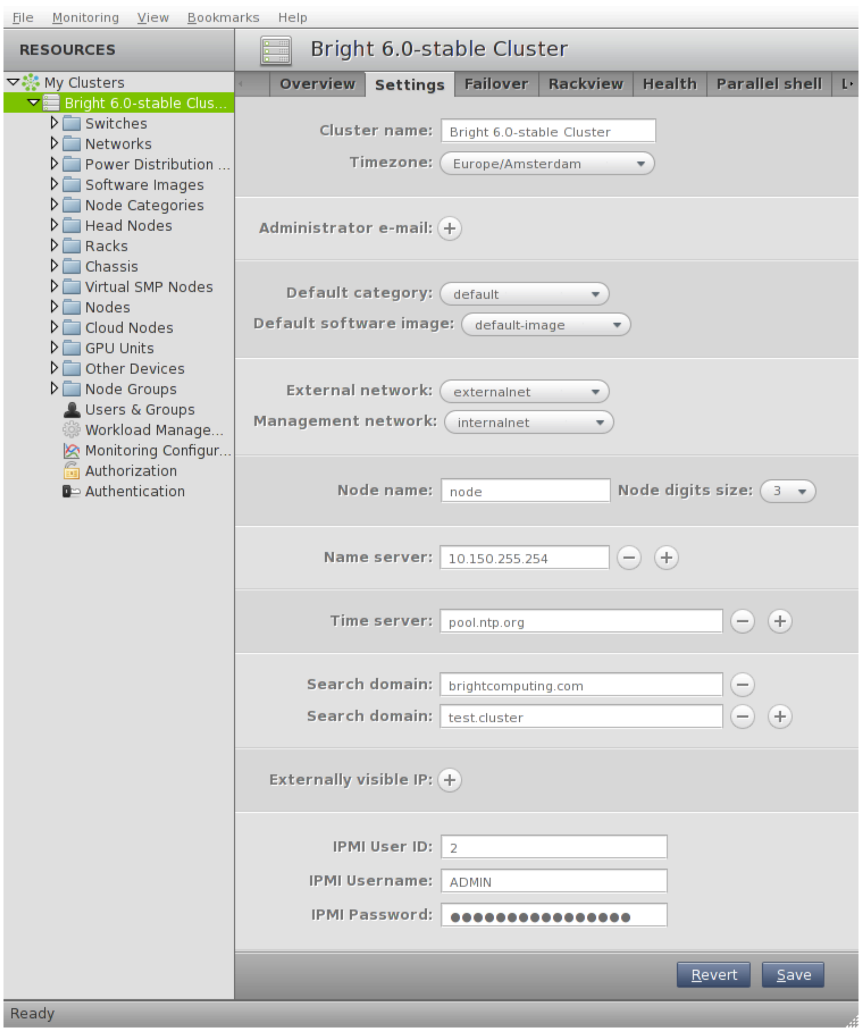
The License Log File

License installation and changes are logged in

```
/var/spool/cmd/license.log
```

to help debug issues.

4.2 Main Cluster Configuration Settings



Cluster name: Bright 6.0-stable Cluster
Timezone: Europe/Amsterdam
Administrator e-mail:
Default category: default
Default software image: default-image
External network: externalnet
Management network: internalnet
Node name: node Node digits size: 3
Name server: 10.150.255.254
Time server: pool.ntp.org
Search domain: brightcomputing.com
Search domain: test.cluster
Externally visible IP:
IPMI User ID: 2
IPMI Username: ADMIN
IPMI Password: ●●●●●●●●●●●●●●●●

Figure 4.2: Cluster Settings

The `Settings` tab for the overall cluster in `cmgui` (figure 4.2) allows changes to be made to some of the main cluster settings related to names, the administrator e-mail address, networks, and some miscellaneous items.

4.2.1 Cluster Configuration: Various Name-Related Settings

In the tab are the following settings, used for names throughout the cluster:

- `Cluster name`: (default: `DDN -stable Cluster`).
- `Default category`: (default: `default`)

- `Default software image`: (default: `default-image`)
- How the nodes of the cluster are named:
  - `Node name`: the base prefix (default prefix: `node`)
  - `Node digits size`: number of digits in suffix of node name (default size: `3`)

4.2.2 Cluster Configuration: Some Network-Related Settings

These following network-related settings are also covered in the context of external network settings for the cluster object, in section 4.3.3, as well as in the quickstart in Appendix F.

Nameserver And Search Domains Used By Cluster

The tab can be used to set the IP address of the nameserver and the names of the search domains for the cluster.

By default, the nameserver is the internal (regular-nodes-facing) IP address of the head node. Multiple nameservers can be added. If the value is set to `0.0.0.0`, then the address supplied via DHCP to the head node is used. Alternatively, a static value can be set.

Instead of using `cmgui`, the changes to the `nameserver` and `searchdomain` values can instead be carried using `cmsh` in partition mode (page 103).

The names of the search domains used by the cluster have a limit of 6 search domains. This is a hardcoded limit imposed by the Linux operating system. Use of FQDNs is advised as a workaround instead of trying to set more search domains.

Externally Visible IP Address

The externally visible IP address are public (non-RFC 1918) IP addresses to the cluster. These can be set to be visible on the external network.

Time server(s)

Time server hostnames can be specified for the NTP client on the head node.

Time Zone

The time zone setting for the cluster is applied to the time that the cluster uses.

- In `cmgui` the time zone can be selected from a menu displayed by clicking on the `Timezone` button in the `Settings` tab of the cluster resource (figure 4.2).
- In `cmsh`, the time zone can be selected in `partition` mode, using the `base` object. Tab-completion prompting after entering "`set timezone`" displays a list of possible time zones, from which one can be chosen:

  Example

  ```
  [ddnmon61]% partition use base
  [ddnmon61->partition[base]]% set timezone america/los_angeles
  [ddnmon61->partition*[base*]]% commit
  ```

4.2.3 Miscellaneous Settings

BMC (IPMI) Settings

The BMC (Baseboard Management Controller) access settings can be configured here:

- `BMC User ID`: (default: 4)
- `BMC Username`: (default: `ddn`)
- `BMC Password`: (default: random string generated during head node installation)

Administrator E-mail Setting

By default, the distribution which DirectMon runs on sends e-mails for the administrator to the root e-mail address. The administrator e-mail address can be changed within DirectMon so that these e-mails are received elsewhere.

- In `cmgui`, an e-mail address can be set in the `Administrator e-mail` section in the `Settings` tab of the cluster resource (figure 4.2).
- In `cmsh`, the e-mail address (or space-separated addresses) can be set in `partition` mode, using the `base` object as follows:

  Example

```
[ddnmon61]% partition use base
[ddnmon61->partition[base]]% set administratore-mail goldstein@oce\
ania.com obrien@oceania.com
[ddnmon61->partition*[base*]]% commit
```

By default, a month before the cluster license expires, a reminder e-mail is sent to the administrator account by CMDaemon. A daily reminder is sent if the expiry is due within a week.

4.3 Network Settings

A simplified quickstart guide to setting up the external head node network configuration on a vendor-prepared cluster is given in Appendix J. This section (4.3) covers network configuration more thoroughly.

After the cluster is set up with the correct license as explained in section 4.1, the next configuration step is to define the networks that are present (sections 4.3.1 and 4.3.2).

During DirectMon installation at least three default network objects were created:

`internalnet`: the primary internal cluster network, or management network. This is used for booting non-head nodes and for all cluster management communication. In the absence of other internal networks, `internalnet` it is also used for storage and communication between compute jobs. Changing the configuration of this network is described on page 103 under the subheading "Changing Internal Network Parameters For The Cluster".

`externalnet`: the network connecting the cluster to the outside world (typically a corporate or campus network). Changing the configuration of this network is described on page 100 under the subheading "Changing External Network Parameters For The Cluster".

`globalnet`: the special network used to set the domain name for nodes so that they can be resolved whether they are cloud nodes or not. This network is described further on page 107 under the subheading "Changing The Global Network Parameters For The Cluster".

For a Type 1 cluster (section 2.3.6) the internal and external networks are illustrated conceptually by figure 4.3.

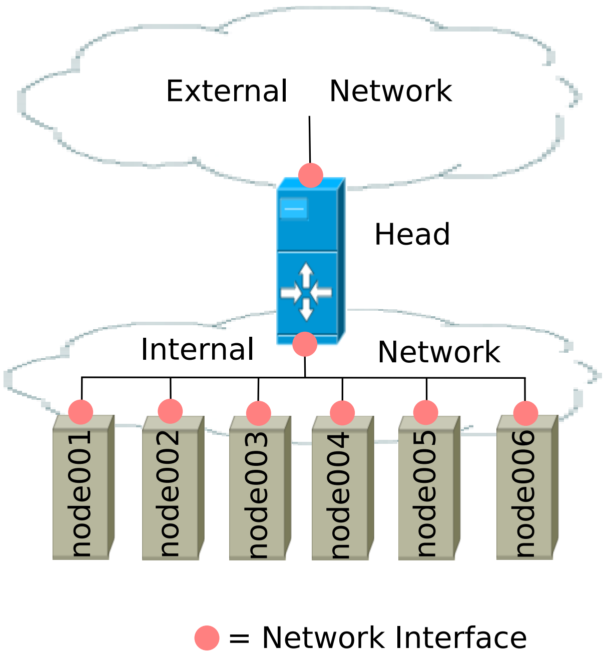


Figure 4.3: Network Settings Concepts

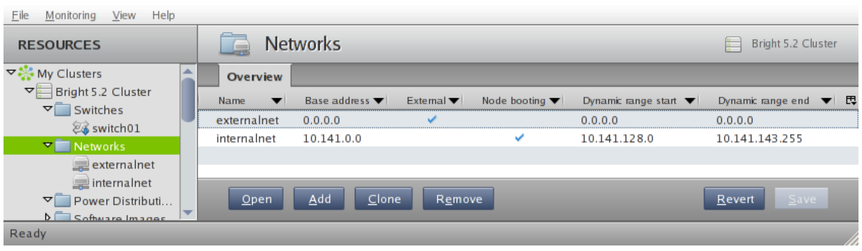
The configuration of network settings is completed when, after having configured the general network settings, specific IP addresses are then also assigned to the interfaces of devices connected to the networks.

- Changing the configuration of the head node external interface is described on page 101 under the subheading "The IP address of the cluster".
- Changing the configuration of the internal network interfaces is described on page 105 under the subheading "The IP addresses and other interface values of the internal network".
- Changing the configuration of `globalnet` is described on page 107 under the subheading "Changing The Global Network Parameters For The Cluster". IP addresses are not configured at the `globalnet` network level itself.

4.3.1 Configuring Networks

The `network` mode in `cmsh` gives access to all network-related operations using the standard object commands. Section 3.5.3 introduces `cmsh` modes and working with objects.

In cmgui, a network can be configured by selecting the Networks item in the resource tree (figure 4.4).

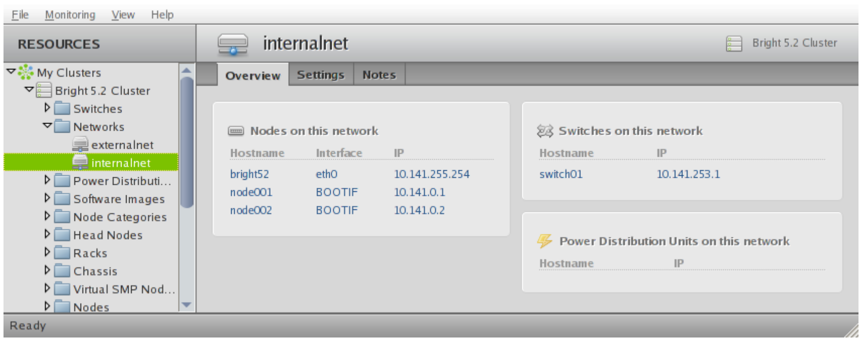


Name	Base address	External	Node booting	Dynamic range start	Dynamic range end
externalnet	0.0.0.0	✓		0.0.0.0	0.0.0.0
internalnet	10.141.0.0		✓	10.141.128.0	10.141.143.255

Figure 4.4: Networks

In the context of the OSI Reference Model, each network object represents a layer 3 (i.e. Network Layer) IP network, and several layer 3 networks can be layered on a single layer 2 network (e.g. routes on an Ethernet segment).

Selecting a network such as internalnet or externalnet in the resource tree displays its tabbed pane. By default, the tab displayed is the Overview tab. This gives a convenient overview of the IP addresses of items assigned in the selected network, grouped as nodes, switches, Power Distribution Units, GPU units, and other devices (figure 4.5).



Nodes on this network

Hostname	Interface	IP
bright52	eth0	10.141.255.254
node001	BOOTIF	10.141.0.1
node002	BOOTIF	10.141.0.2

Switches on this network

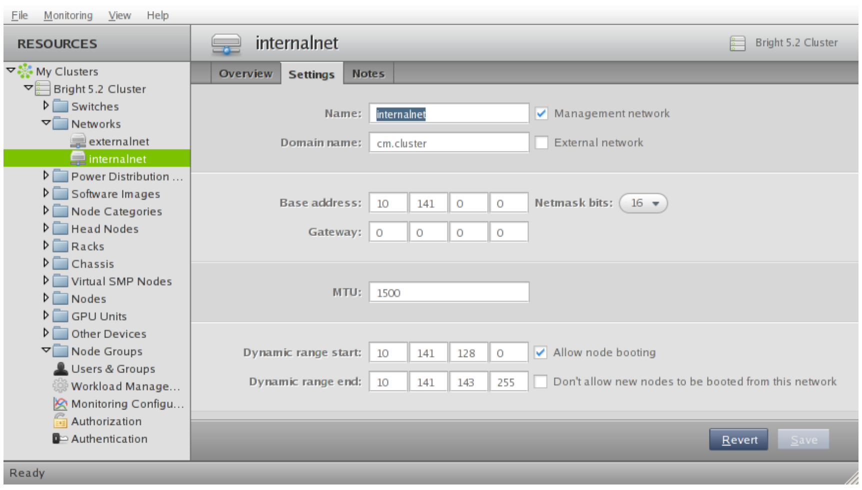
Hostname	IP
switch01	10.141.253.1

Power Distribution Units on this network

Hostname	IP

Figure 4.5: Network Overview

Selecting the Settings tab allows a number of network properties to be changed (figure 4.6).



Name:	internalnet	✓ Management network
Domain name:	cm.cluster	☐ External network
Base address:	10 141 0 0	Netmask bits: 16
Gateway:	0 0 0 0	
MTU:	1500	
Dynamic range start:	10 141 128 0	✓ Allow node booting
Dynamic range end:	10 141 143 255	☐ Don't allow new nodes to be booted from this network

Figure 4.6: Network Settings

The properties of figure 4.6 are introduced in table 4.3.1.

Table 4.3.1: Network Configuration Settings

Property	Description
`Name`	Name of this network.
`Domain name`	DNS domain associated with the network.
`Management network`	Switch to treat the network as having nodes managed by the head node.
`External network`	Switch to treat the network as an external network.
`Base address`	Base address of the network (also known as the *network address*)
`Gateway`	Default route IP address. `0.0.0.0` means a default value is accepted. The default value is taken from a DHCP lease if DHCP is set in CMDaemon (using `cmgui`: with a DHCP checkbox, using `cmsh`: by setting`dhcp on` in the interface device for the regular node or head node). For regular nodes, a DHCP lease value or specific node value is overridden by values set at the node category level.
`Netmask bits`	Prefix-length, or number of bits in netmask. The part after the "/" in CIDR notation.
`MTU`	Maximum Transmission Unit. The maximum size of an IP packet transmitted without fragmenting.
`Dynamic range start/end`	Start/end IP addresses of the DHCP range temporarily used by nodes during PXE boot on the internal network.
`Allow node booting`	Nodes set to boot from this network (useful in the case of nodes on multiple networks).
`Don't allow new nodes to boot from this network`	When set to true, new nodes are not offered a PXE DHCP IP address from this network, i.e. DHCPD is "locked down". A DHCP "deny unknown-clients" option is set by this action, so no new DHCP leases are granted to unknown clients for the network. Unknown clients are nodes for which DirectMon has no MAC addresses associated with the node. • It can be set in `cmsh` via the network mode, selecting a network, and then setting a value for `lockdowndhcpd`. • it can be set in `cmgui` via the `Networks` resource, selecting a network item, and then choosing the `Settings` tabbed pane.

In basic networking concepts, a network is a range of IP addresses. The first address in the range is the *base address*. The length of the range, i.e. the *subnet*, is determined by the *netmask*, which uses CIDR notation. CIDR notation is the so-called / ("slash") representation, in which, for example, a CIDR notation of 192.168.0.1/28 implies an IP address of 192.168.0.1 with a traditional netmask of 255.255.255.240 applied to the 192.168.0.0 network. The netmask 255.255.255.240 implies that bits 28–32 of the 32-bit dotted-quad number 255.255.255.255 are unmasked, thereby implying a 4-bit-sized host range of 16 (i.e. 2^4) addresses.

The `sipcalc` utility installed on the head node is a useful tool for calculating or checking such IP subnet values (`man sipcalc` or `sipcalc -h` for help on this utility):

Example

```
user@ddnmon61:~$ sipcalc 192.168.0.1/28
-[ipv4 : 192.168.0.1/28] - 0

[CIDR]
Host address            - 192.168.0.1
Host address (decimal)  - 3232235521
Host address (hex)      - C0A80001
Network address         - 192.168.0.0
Network mask            - 255.255.255.240
Network mask (bits)     - 28
Network mask (hex)      - FFFFFFF0
Broadcast address       - 192.168.0.15
Cisco wildcard          - 0.0.0.15
Addresses in network    - 16
Network range           - 192.168.0.0 - 192.168.0.15
Usable range            - 192.168.0.1 - 192.168.0.14
```

Every network has an associated DNS domain which can be used to access a device through a particular network. For `internalnet`, the default DNS domain is set to `cm.cluster`, which means that the hostname `node001.cm.cluster` can be used to access device `node001` through the primary internal network. If a dedicated storage network has been added with DNS domain `storage.cluster`, then `node001.storage.cluster` can be used to reach `node001` through the storage network. Internal DNS zones are generated automatically based on the network definitions and the defined nodes on these networks. For networks marked as external, no DNS zones are generated.

4.3.2 Adding Networks

Once a network has been added, it can be used in the configuration of network interfaces for devices.

In `cmgui` the `Add` button in the networks overview tab of figure 4.4 can be used to add a new network. After the new network has been added, the `Settings` tab (figure 4.6) can be used to further configure the newly added network.

In `cmsh`, a new network can be added from within `network` mode using the `add` or `clone` commands.

The default assignment of networks (internalnet to Management network and externalnet to External network) can be changed using cmgui in the Settings tab of the cluster object (figure 4.2).

In cmsh the assignment to Management network and External network can be set or modified from the base object in partition mode:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% partition use base
[ddnmon61->partition[base]]% set managementnetwork internalnet; commit
[ddnmon61->partition[base]]% set externalnetwork externalnet; commit
```

4.3.3 Changing Network Parameters

After both internal and external networks are defined, it may be necessary to change network-related parameters from their original or default installation settings.

Changing The Head Or Regular Node Hostname

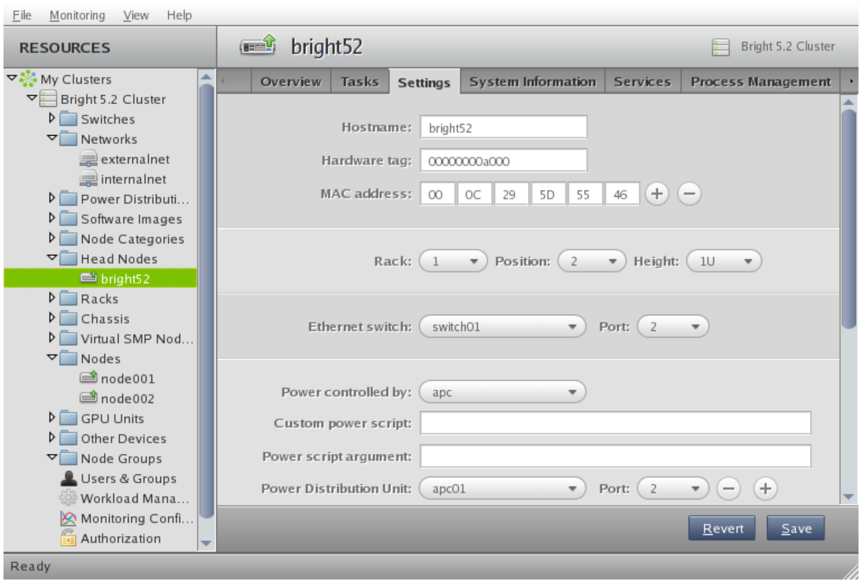
To reach the head node from inside the cluster, the alias master may be used at all times. Setting the hostname of the head node itself to master is not recommended.

The name of a cluster is sometimes used as the hostname of the head node. The cluster name, the head node hostname, and the regular node hostnames, all have their own separate names as a property of their corresponding objects. The name can be changed in a similar manner for each.

For example, to change the hostname of the head node, the device object corresponding to the head node must be modified.

- In cmgui, the device listed under Head Nodes in the resource tree is selected and its Settings tab selected from the tabbed pane (figure 4.7). The hostname is changed by modifying the Hostname property and clicking on Save. When setting a hostname, a domain is not included.

  After the change, as suggested by cmgui, the head node must be rebooted.



Hostname: bright52
Hardware tag: 00000000a000
MAC address: 00 0C 29 5D 55 46
Rack: 1 Position: 2 Height: 1U
Ethernet switch: switch01 Port: 2
Power controlled by: apc
Custom power script:
Power script argument:
Power Distribution Unit: apc01 Port: 2

Figure 4.7: Head Node Settings

- In `cmsh`, the hostname of the head node can also be changed, via device mode:

  Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device use ddnmon61
[ddnmon61->device[ddnmon61]]% set hostname foobar
[foobar->device*[foobar*]]% commit
[foobar->device[foobar]]%
Tue Jan 22 17:35:29 2013 [warning] foobar: Reboot required: Hostname \
changed
[foobar->device[foobar]]% quit
[root@ddnmon61 ~]# sleep 30; hostname -f foobar.cm.cluster
[root@ddnmon61 ~]#
```

  The prompt string shows the new hostname after a short time, when a new shell is started.

  After the hostname has been set, as suggested by `cmsh`, the head node must be rebooted.

Adding Hostnames To The Internal Network

Additional hostnames, whether they belong to the cluster nodes or not, can be added as name/value pairs to the `/etc/hosts` file(s) within the cluster. This should be done only outside the specially-marked CMDaemon-managed section. It can be done to the file on the head node, or to the file on the software image for the regular nodes, or to both, depending on where the resolution is required.

However, for hosts that are on the internal network of the cluster, such as regular nodes, it is easier and wiser to avoid adding additional hostnames via `/etc/hosts`.

Instead, it is recommended to let DirectMon manage host name resolution for devices on the `internalnet` through its DNS server on the internalnet interface. The host names can be added to the additionalhostnames object, from within interfaces submode for the head node. The interfaces submode is accessible from the device mode. Thus, for the head node, with eth1 as the interface for internalnet:

Example

```
[ddnmon61]% device use ddnmon61
[ddnmon61->device[ddnmon61]]% interfaces
[ddnmon61->device[ddnmon61]->interfaces]% use eth1
[ddnmon61->device[ddnmon61]->interfaces[eth1]]% set additionalhostnames\
 test
[ddnmon61->device*[ddnmon61*]->interfaces*[eth1*]]% commit
[ddnmon61->device[ddnmon61]->interfaces[eth1]]%
Fri Oct 12 16:48:47 2012 [notice] ddnmon61: Service named was restarted
[ddnmon61->device[ddnmon61]->interfaces[eth1]]% !ping test
PING test.cm.cluster (10.141.255.254) 56(84) bytes of data.
...
```

Multiple hostnames can be added as space-separated entries.

The named service automatically restarts within about 20 seconds after committal, implementing the configuration changes. This kind of restart is a feature (section 4.12.1) of changes made to service configurations by cmgui or cmsh.

Changing External Network Parameters For The Cluster

The external network parameters of the cluster: When a cluster interacts with an external network, such as a company or a university network, its connection behavior is determined by the settings of two objects: firstly, the external network settings of the Networks resource, and secondly, by the cluster network settings.

1. **The external network object** contains the network settings for all objects configured to connect to the external network, for example, a head node. Network settings are configured in the Settings tab of the Networks resource of cmgui. Figure 4.6 shows a settings tab for when the internalnet item has been selected, but in the current case the externalnet item must be selected instead. The following parameters can then be configured:

   - the IP network parameters of the cluster (but not the IP address of the cluster):
     - Base address: the network address of the external network (the "IP address of the external network"). This is not to be confused with the IP address of the cluster, which is described shortly after this.
     - Netmask bits: the netmask size, or prefix-length, of the external network, in bits.
     - Gateway: the default route for the external network.
     - Dynamic range start and Dynamic range end: Not used by the external network configuration.

- the `Domain name`: the network domain (LAN domain, i.e. what domain machines on the external network use as their domain),
- network name (what the external network itself is called): by default this is `externalnet` on a newly installed Type 1 cluster,
- the `External network` checkbox: this is checked for a Type 1 cluster,
- and MTU size (the maximum value for a TCP/IP packet before it fragments on the external network—the default value is 1500).

2. **The cluster object** contains other network settings used to connect to the outside. These are configured in the `Settings` tab of the cluster object resource in `cmgui` (figure 4.2):

   - e-mail address(es) for the cluster administrator,
   - any additional external name servers used by the cluster to resolve external host names. As an aside: by default, only the head node name server is configured, and by default it only serves hosts on the internal network via the internal interface. Enabling the `PublicDNS` directive (Appendix C) allows the head node name server to resolve queries about internal hosts from external hosts via the external interface.
   - the DNS search domain (what the cluster uses as its domain),
   - and NTP time servers (used to synchronize the time on the cluster with standard time) and time zone settings.

   These settings can also be adjusted in `cmsh` in the `base` object under `partition` mode.

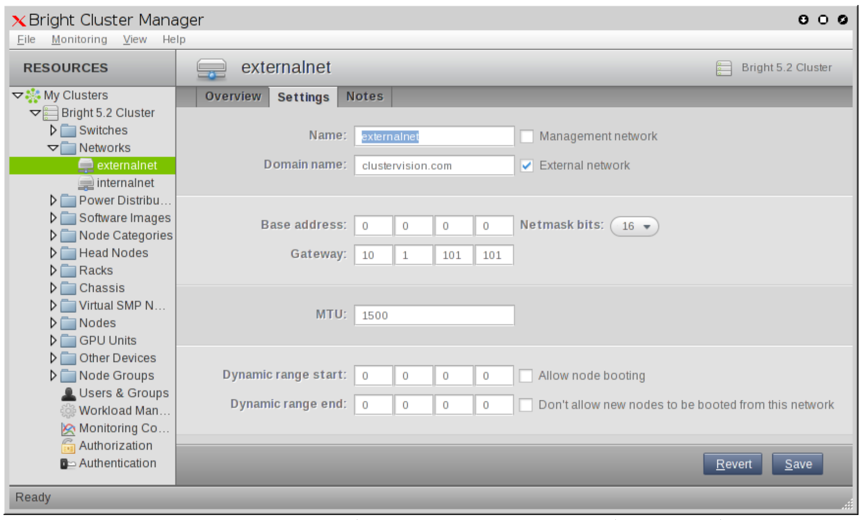
Changing the networking parameters of a cluster (apart from the IP address of the cluster) therefore requires making changes in the settings of the two preceding objects.

The IP address of the cluster: The cluster object itself does not contain an IP address value. This is because it is the cluster network topology type that determines whether a direct interface exists from the cluster to the outside world. Thus, the IP address of the cluster in the Type 1, Type 2, and Type 3 configurations (section 2.3.6) is defined by the cluster interface that faces the outside world. For Type 1, this is the interface of the head node to the external network (figure 4.3). For Type 2 and Type 3 interfaces the cluster IP address is effectively that of an upstream router, and thus not a part of DirectMon configuration. Thus, logically, the IP address of the cluster is not a part of the cluster object or external network object configuration.

For a Type 1 cluster, the head node IP address can be set in DirectMon, separately from the cluster object settings. This is then the IP address of the cluster according to the outside world.

Setting the network parameters of the cluster and the head node IP address: These values can be set using `cmgui` or `cmsh`:

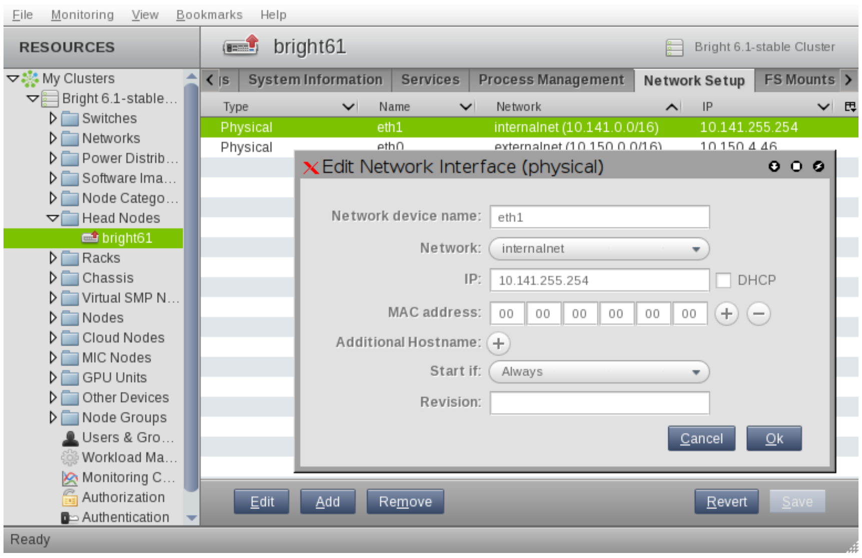
With `cmgui`**:** The associated cluster network settings tabs are accessed as shown in figure 4.8 for the external network object, and as shown in figure 4.2 for the cluster object.



Name: externalnet ☐ Management network
Domain name: clustervision.com ☑ External network
Base address: 0 0 0 0 Netmask bits: 16
Gateway: 10 1 101 101
MTU: 1500
Dynamic range start: 0 0 0 0 ☐ Allow node booting
Dynamic range end: 0 0 0 0 ☐ Don't allow new nodes to be booted from this network

Figure 4.8: Network Settings For External Network

Setting the static IP address of the head node can be done by selecting the head node from the `Head Nodes` resources tab, then selecting the `Network Setup` tabbed pane, then selecting the interface for the address. Clicking on the `Edit` button opens up an editor that allows the IP address to be changed (figure 4.9).



Type	Name	Network	IP
Physical	eth1	internalnet (10.141.0.0/16)	10.141.255.254
Physical	eth0	externalnet (10.150.0.0/16)	10.150.4.46

Network device name: eth1
Network: internalnet
IP: 10.141.255.254 ☐ DHCP
MAC address: 00 00 00 00 00 00
Additional Hostname:
Start if: Always
Revision:

Figure 4.9: Setting The IP Address On A Head Node In `cmgui`

With `cmsh`**:** The preceding `cmgui` configuration can also be done in `cmsh`, using the `network`, `partition` and `device` modes, as in the following example:

Example

```
[ddnmon61]% network use externalnet
```

```
[ddnmon61->network[externalnet]]% set baseaddress 192.168.1.0
[ddnmon61->network*[externalnet*]]% set netmaskbits 24
[ddnmon61->network*[externalnet*]]% set gateway 192.168.1.1
[ddnmon61->network*[externalnet*]]% commit
[ddnmon61->network[externalnet]]% partition use base
[ddnmon61->partition[base]]% set nameservers 192.168.1.1
[ddnmon61->partition*[base*]]% set searchdomains x.com y.com
[ddnmon61->partition*[base*]]% append timeservers ntp.x.com
[ddnmon61->partition*[base*]]% commit
[ddnmon61->partition[base]]% device use ddnmon61
[ddnmon61->device[ddnmon61]]% interfaces
[ddnmon61->device[ddnmon61]->interfaces]% use eth1
[ddnmon61->device[ddnmon61]->interfaces[eth1]]% set ip 192.168.1.176
[ddnmon61->device[ddnmon61]->interfaces*[eth1*]]% commit
[ddnmon61->device[ddnmon61]->interfaces[eth1]]%
```

After changing the external network configurations, a reboot of the head node is necessary to activate the changes.

Using DHCP to supply network values for the external interface: Connecting the cluster via DHCP on the external network is not generally recommended for production clusters. This is because DHCP-related issues can complicate networking troubleshooting compared with using static assignments.

For a Type 1 network, the cluster and head node can be made to use some of the DHCP-supplied external network values as follows:

- In `cmgui`, the DHCP checkbox of figure 4.9 can be ticked
- Alternatively, in `cmsh`, within interfaces mode for the head node interface, the value of the parameter DHCP can be set:

```
[ddnmon61->device[ddnmon61]->interfaces[eth0]]% set dhcp yes
```

The gateway address, the name server(s), and the external IP address of the head node are then obtained via a DHCP lease. Time server configuration for `externalnet` is not picked up from the DHCP server, having been set during installation (figure 2.24). The time servers can be changed using `cmgui` as in figure 4.2, or using `cmsh` in `partition` mode as in the preceding example. The time zone can be changed similarly.

It is usually sensible to reboot after implementing these changes in order to test the changes are working as expected.

Changing Internal Network Parameters For The Cluster

When a cluster interacts with the internal network that the regular nodes and other devices are on, its connection behavior with the devices on that network is determined by settings in:

1. the internal network of the `Networks` resource (page 104)
2. the cluster network for the internal network (page 105)
3. the individual device network interface (page 105)
4. the node categories network-related items for the device (page 106), in the case of the device being a regular node.

In more detail:

1. The internal network object: has the network settings for all devices connecting to the internal network, for example, a login node, a head node via its `internalnet` interface, or a managed switch on the internal network. In `cmgui`, the settings can be configured in the `Settings` tab for the `internalnet` item selected in the `Networks` resource (figure 4.6). In `cmsh` the settings can be configured by changing the values in the `internalnet` object within `cmsh`'s `network` mode. Parameters that can be changed include:

- the IP network parameters of the internal network (but not the internal IP address):
  - "`Base address`": the internal network address of the cluster (the "IP address of the internal network"). This is not to be confused with the IP address of the internal network interface of the head node. The default value is `10.141.0.0`.
  - "`Netmask bits`": the netmask size, or prefix-length, of the internal network, in bits. The default value is `16`.
  - `Gateway`: the default gateway route for the internal network. If unset, or 0.0.0.0 (the default), then its value is decided by the DHCP server on the head node, and nodes are assigned a default gateway route of `10.141.255.254`, which corresponds to using the head node as a default gateway route for the interface of the regular node. The effect of this DHCP parameter is overridden by any default gateway route value set by the value of `Default gateway` in the node category. A node-specific gateway value cannot be set in CMDaemon, so a node category value is the final "decider" in the hierarchy.
  - "`Dynamic range start`" and "`Dynamic range end`": These are the DHCP ranges for nodes. DHCP is unset by default. When set, unidentified nodes can use the dynamic IP address values assigned to the node by the node-installer. These values range by default from `10.141.128.0` to `10.141.143.255`. Using this feature is not generally recommended, in order to avoid the possibility of conflict with the static IP addresses of identified nodes.
  - `Allow node booting`: This allows nodes to boot from the provisioning system controlled by CMDaemon. The parameter is normally set for the management network (that is the network over which CMDaemon communicates to manage nodes) but booting can instead be carried out over a separate physical non-management network. Booting over InfiniBand is one of the possibilities (section 6.1.3). Only if the `Allow node booting` option is ticked does ticking the "`Don't allow new nodes to be booted from this network`" checkbox have any effect, and stop new nodes from booting. New nodes are those nodes which are detected but the cluster cannot identify based on CMDaemon records. Details are given in Chapter 6 about booting, provisioning, and how a node is detected as new.

- the "domain name" of the network. This is the LAN domain, which is the domain machines on this network use as their domain. By default, set to cm.cluster.
- the network name, or what the internal network is called. By default, set to internalnet.
- The MTU size, or the maximum value for a TCP/IP packet before it fragments on this network. By default, set to 1500.

2. The cluster object: has other network settings that the internal network in the cluster uses. These particulars are configured in the Settings tab of the cluster object resource in cmgui (figure 4.2). The cmsh equivalents can be configured from the base object in partition mode. Values that can be set include:

- the "Management network". This is the network over which CMDaemon manages the nodes. By default, it is set to internalnet for Type 1 and Type 2 networks, and managementnet in Type 3 networks. It is a partition-level cluster-wide setting by default. Partition-level settings can be overridden by the category level setting, and node-level settings can override category level or partition level setings.
- the "Node name" can be set to decide the prefix part of the node name. By default, set to node.
- the "Node digits" can be set to decide the possible size of numbers used for suffix part of the node name. By default, set to 3.
- the "Default category". This sets the category the nodes are in by default. By default, it is set to default.
- the "Default software image". This sets the image the nodes use by default, for new nodes that are not in a category yet. By default, it is set to default-image.
- the "Name server". This sets the name server used by the cluster. By default, it is set to the head node. The default configuration uses the internal network interface and serves only internal hosts. Internal hosts can override using this value at category level (page 107). By default external hosts cannot use this service. To allow external hosts to use the service for resolving internal hosts, the PublicDNS directive (Appendix C) must be set to True.

3. The internal IP addresses and other internal interface values: The "Network Setup" tab can be chosen for a node selected from the Nodes resource in cmgui. The tab then allows network configuration to be done for nodes in a similar way to figure 4.9. In cmsh this can be done by changing properties from the interfaces submode that is within the device mode for the node.

The items that can be set include:

- the Network device name: By default, this is set to BOOTIF for a node that boots from the same interface as the one from which it is provisioned.

- the Network: By default, this is set to a value of internalnet.
- the IP address: By default, this is automatically assigned a static value, in the range 10.141.0.1 to 10.141.255.255, with the first node being given the first address. Using a static IP address is recommended for reliablity, since it allows nodes that lose connectivity to the head node to continue operating. The static address can be changed manually, in case there is an IP address or node ID conflict due to an earlier inappropriate manual intervention.

  Administrators who want DHCP addressing on the internal network, despite the consequences, can set it via a checkbox.
- Additional Hostname: In the case of nodes this is in addition to the default node name set during provisioning. The node name set during provisioning takes a default form of node<*3 digit number*>, as explained earlier on page 105 in the section describing the cluster object settings.

For, example, a regular node that has an extra interface, eth1, can have its values set as follows:

Example

```
[ddnmon61] device interfaces node001
[ddnmon61->device[node001]->interfaces]% add physical eth1
[ddnmon61->...->interfaces*[eth1*]]% set network externalnet
[ddnmon61->...->interfaces*[eth1*]]% set additionalhostname extra01
[ddnmon61->...->interfaces*[eth1*]% set ip 10.141.1.1
[ddnmon61->...->interfaces*[eth1*]]% commit
[ddnmon61->...->interfaces[eth1]]% ..
[ddnmon61->...->interfaces]% ..
[ddnmon61->device[node001]]% reboot
```

4. Node category network values: are settings for the internal network that can be configured for node categories using the Settings tab in cmgui, or the category mode in cmsh, for a particular category of node. If there are individual node settings that can be configured in cmgui or cmsh, then the node settings override the corresponding category settings for those particular nodes.

The category properties involved in internal networking that can be set include:

- Default gateway: The default gateway route for nodes in the node category. If unset, or 0.0.0.0 (the default), then the node default gateway route is decided by the internal network object Gateway value. If the default gateway is set as a node category value, then nodes use the node category value as their default gateway route instead.
- Management network: The management network is the network used by CMDaemon to manage devices. The default setting is a property of the node object. It can be set as a category property.

- `Name server`, `Time server`, `Search domain`: The default setting for these on all nodes is set by the node-installer to refer to the head node, and is not configurable at the node level using `cmgui` or `cmsh`. The setting can however be set as a category property, either as a space-separated list of entries or as a single entry, to override the default value.

Application To Generic Network Devices: The preceding details for the internal network parameters of the cluster, starting from page 103, are applicable to regular nodes, but they are often also applicable to generic network devices (section 3.1.1). Benefits of adding a generic device to be managed by DirectMon include that:

- the name given to the device during addition is automatically placed in the internal DNS zone, so that the device is accessible by name
- the device status is automatically monitored via a SYN ping (Appendix H.2.1).
- the device can be made to work with the health check and metric framework. The scripts used in the framework will however almost certainly have to be customized to work with the device

After changing network configurations, a reboot of the device is necessary to activate the changes.

Changing The Global Network Parameters For The Cluster

The global network `globalnet` is a unique network used to set up a common name space for all nodes in a cluster in DirectMon. It is required in DirectMon because of the added cloud extension functionality, described in Chapter 7. Regular nodes and regular cloud nodes are thus both named under the `globalnet` domain name, which is `cm.cluster` by default. So, for example, if default host names for regular nodes (node001, node002, ...) and regular cloud nodes (cnode001, cnode002, ...) are used, node addresses with the domain are:

- node001.cm.cluster for node001
- cnode001.cm.cluster for cnode001

The only parameter that can be sensibly changed on `globalnet` is the domain name, which is `cm.cluster` by default.

Removing `globalnet` should not be done because it will cause various networking failures, even for clusters deploying no cloud nodes.

Details on how resolution is carried out with the help of `globalnet` are given in section 7.4.4.

4.4 Configuring Bridge Interfaces

Bridge interfaces can be used to divide one physical network into two separate network segments at layer 2 without creating separate IP subnets. A bridge thus connects the two networks together at layer 3 in a protocol-independent way.

The network device name given to the bridge interface is of the form `br`*`<number>`*. The following example demonstrates how to set up a bridge interface in `cmsh`, where the name `br0` is stored by the parameter `networkdevicename`.

Example

```
[ddnmon61->device[node001]->interfaces]% add bridge br0
[ddnmon61->...->interfaces*[br0*]]% set network internalnet
[ddnmon61->...->interfaces*[br0*]]% set ip 10.141.1.1
[ddnmon61->...->interfaces*[br0*]]% show
Parameter                      Value
------------------------------ ------------------
Additional Hostnames
DHCP                           no
Forward Delay                  0
IP                             10.141.1.1
Interfaces
MAC                            00:00:00:00:00:00
Network                        internalnet
Network device name            br0
Revision
SpanningTreeProtocol           no
Type                           bridge
```

A bridge interface is composed of one or more physical interfaces. The IP and network fields of the member interfaces must be empty:

Example

```
[ddnmon61->...->interfaces*[br0*]]% set interfaces eth1 eth2; exit
[ddnmon61->...->interfaces*]% clear eth1 ip; clear eth1 network
[ddnmon61->...->interfaces*]% clear eth2 ip; clear eth2 network
[ddnmon61->...->interfaces*]% use br0; commit
```

The `BOOTIF` interface is also a valid interface option.

Currently, the following bridge properties can be set:

- `SpanningTreeProtocol`: sets the spanning tree protocol to be on or off. The default is off.
- `Forward Delay`: sets the delay for forwarding Ethernet frames, in seconds. The default is 0.

Additional properties, if required, can be set manually using the `brctl` command in the OS shell.

When listing interfaces in `cmsh`, if an interface is a member of a bond (or bridge) interface, then the corresponding bond or bridge interface name is shown in parentheses after the member interface name:

Example

```
[headnode->device[node001]->interfaces]% list
Type         Network device name  IP               Network
------------ -------------------- ---------------- ----------------
bond         bond0 [prov]         10.141.128.1     internalnet
bridge       br0                  10.141.128.2     internalnet
physical     eth0                 10.141.0.1       internalnet
physical     eth1 (bond0)         0.0.0.0
physical     eth2 (bond0)         0.0.0.0
physical     eth3 (br0)           0.0.0.0
physical     eth4 (br0)           0.0.0.0
```

It is possible to create a bridge interface with no IP address configured, that is, with an IP address of 0.0.0.0. This can be done for security reasons, or when the number of available IP addresses on the network is scarce. As long as such a bridge interface has a network assigned to it, it is properly configured on the nodes and functions as a bridge on the network.

4.5 Configuring VLAN interfaces

A VLAN (Virtual Local Area Network) is an independent logical LAN within a physical LAN network. VLAN tagging is used to segregate VLANs from each other. VLAN tagging is the practice of inserting a VLAN ID tag into a packet frame header, so that each packet can be associated with a VLAN.

The physical network then typically has sections that are VLAN-aware or VLAN-unaware. The nodes and switches of the VLAN-aware section are configured by the administrator to decide which traffic belongs to which VLAN.

A VLAN interface can be configured for an interface within DirectMon using `cmsh` or `cmgui`.

4.5.1 Configuring A VLAN Interface Using `cmsh`

In the following `cmsh` session, a regular node interface that faces a VLAN-aware switch is made VLAN-aware, and assigned a new interface—an alias interface. It is also assigned a network, and an IP address within that network:

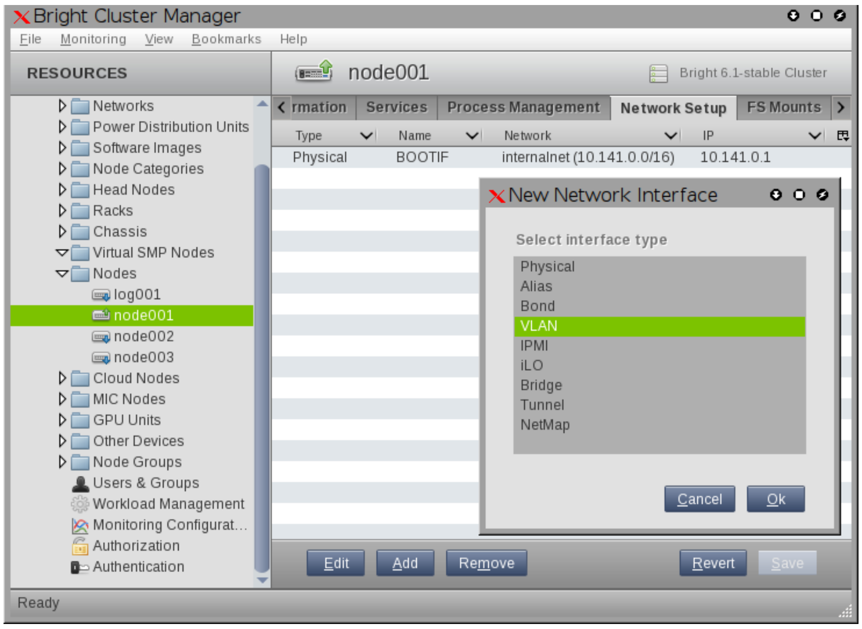
Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device interfaces node001
[ddnmon61->device[node001]->interfaces]% list
Type         Network device name  IP              Network
------------ -------------------- --------------- ------------
physical     BOOTIF [prov]        10.141.0.1      internalnet
Arguments:
   type
     physical, bond, bridge, tunnel, netmap, alias, bmc, vlan

[ddnmon61->device[node001]->interfaces]% add vlan BOOTIF.1
[ddnmon61->device*[node001*]->interfaces*[BOOTIF.1*]]% commit
[ddnmon61->device[node001]->interfaces[BOOTIF.1]]% show

Parameter                  Value
-------------------------- ------------------------
Additional Hostname
DHCP                       no
IP                         0.0.0.0
MAC                        00:00:00:00:00:00
Network
```

```
Network device name       BOOTIF.1
Reorder HDR               no
Revision
Type                      vlan
[ddnmon61->...[BOOTIF.1]]% set network internalnet
[ddnmon61->...[BOOTIF.1*]]% set ip 10.141.2.1; commit
```

4.5.2 Configuring A VLAN Interface Using `cmgui`

Within `cmgui` a VLAN interface can be configured by selecting its node from the resources, selecting the `Network Setup` tabbed pane, and clicking on the `Add` button. The interface type is then selected and a dialog opens up allowing an IP address, network, and other options to be set (figure 4.10).



Type	Name	Network	IP
Physical	BOOTIF	internalnet (10.141.0.0/16)	10.141.0.1

Figure 4.10: Configuring A VLAN via `cmgui`

4.6 Configuring Bonded Interfaces

4.6.1 Adding A Bonded Interface

The Linux bonding driver allows multiple physical network interfaces that have been configured previously (for example, as on page 106) to be bonded as a single logical bond interface. The behavior of such interfaces is determined by their bonding mode. The modes provide facilities such as hot standby or load balancing.

The driver is included by default on head nodes. To configure a non-head node to use a bonded interface, a Linux kernel module called the `bonding` module must be included in the kernel modules list of the software image of the node. A bonding interface can then be created, and its properties set as follows:

Example

```
[ddnmon61->device[node001]->interfaces]% add bond bond0
[...->device*[node001*]->interfaces*[bond0*]]% set network internalnet
```

```
[...->device*[node001*]->interfaces*[bond0*]]% set ip 10.141.128.1
[...->device*[node001*]->interfaces*[bond0*]]% set mode 3
[...->device*[node001*]->interfaces*[bond0*]]% set interfaces eth1 eth2
```

Each bonded interface has a unique set of object properties, called bonding options, that specify how the bonding device operates. The bonding options are a string containing one or more options formatted as `option=value` pairs, with pairs separated from each other by a space character.

A bonded interface also always has a mode associated with it. By default it is set to `0`, corresponding to a balanced round-robin packet transmission.

The 7 bonding modes are:

- 0 – balance-rr
- 1 – active-backup
- 2 – balance-xor
- 3 – broadcast
- 4 – 802.3ad
- 5 – balance-tlb
- 6 – balance-alb

Technically, outside of `cmgui` or `cmsh`, the bonding mode is just another bonding option specified as part of the options string. However in DirectMon the bonding mode value is set up using the dedicated `mode` property of the bonded interface, for the sake of clarity. To avoid conflict with the value of the `mode` property, trying to commit a bonding mode value as an `option=value` pair will fail validation.

4.6.2 Single Bonded Interface On A Regular Node

A single bonded interface on a node can be configured and coexist in several ways on nodes with multiple network interfaces. Possibilities and restrictions are illustrated by the following:

- The bonded interface may be made up of two member interfaces, and a third interface outside of the bond could be the boot interface. (The boot interface is the node interface used to PXE boot the node before the kernel loads (section 6.1)).
- The boot interface could itself be a member of the bonded interface. If the boot interface is a member of a bonded interface, then this is the first bonded interface when interfaces are listed as in the example on page 108.
- The bonded interface could be set up as the provisioning interface. However, the provisioning interface cannot itself be a member of a bonded interface. (The provisioning interface is the node's interface that picks up the image for the node after the initial ramdisk is running. Chapter 6 covers this in more detail).
- A bonded interface can be set up as the provisioning interface, while having a member interface which is used for PXE booting.

4.6.3 Multiple Bonded Interface On A Regular Node

A node can also have multiple bonded interfaces configured. Possibilities and restrictions are illustrated by the following:

- Any one of the configured bonded interfaces can be configured as the provisioning interface. However, as already mentioned in the case of single bond interfaces (section 4.6.2), a particular member of a bonded interface cannot be made the provisioning interface.
- When a bonded interface is set as the provisioning interface, then during the node-installer phase of boot, the node-installer brings up the necessary bonded interface along with all its member interfaces so that node provisioning is done over the bonded interface.

4.6.4 Bonded Interfaces On Head Nodes And HA Head Nodes

It is also possible to configure bonded interfaces on head nodes.

For a single head node setup, this is analogous to setting up bonding on regular nodes.

For a high availability (HA) setup (chapter 15), bonding is possible for internalnet as well as for externalnet, but it needs the following extra changes:

- For the bonded interface on internalnet, the shared internal IP alias interface name (the value of networkdevicename, for example, eth0:0 in figure 15.1) for that IP address should be renamed to the bonded alias interface name on internalnet (for example, bond0:0).
- For the bonded interface on externalnet, the shared external IP alias interface name (the value of networkdevicename, for example, eth1:0 in figure 15.1) for that IP address should be renamed to the bonded alias interface name on externalnet (for example, bond1:0).
- Additionally, when using a bonded interface name for the internal network, the value of the provisioning network interface name (the value of provisioninginterface, for example, eth0) for the head nodes, must be changed to the name of the bonded interface (for example, bond0 [prov]) on the internal network. The provisioninginterface value setting is described further on page 221.

Example

```
[headnode1->device[headnode1]->interfaces]% list
Type         Network device name  IP                Network
------------ -------------------- ----------------- ------------
alias        bond0:0              10.141.255.252    internalnet
alias        bond1:0              10.150.57.1       externalnet
bond         bond0 [prov]         10.141.255.254    internalnet
bond         bond1                10.150.57.3       externalnet
physical     eth0 (bond0)         0.0.0.0
physical     eth1 (bond0)         0.0.0.0
physical     eth2 (bond1)         0.0.0.0
physical     eth3 (bond1)         0.0.0.0
```

4.6.5 Tagged VLAN On Top Of a Bonded Interface

It is possible to set up a tagged VLAN interface on top of a bonded interface. There is no requirement for the bonded interface to have an IP address configured in this case. The IP address can be set to `0.0.0.0`, however a network must be set.

Example

```
[headnode1->device[node001]->interfaces]% list
Type         Network device name  IP               Network
------------ -------------------- ---------------- ------------
bond         bond0                0.0.0.0          internalnet
physical     eth0 [prov]          10.141.0.1       internalnet
physical     eth1 (bond0)         0.0.0.0
physical     eth2 (bond0)         0.0.0.0
vlan         bond0.3              10.150.1.1       othernet
```

4.6.6 Further Notes On Bonding

If using bonding to provide failover services, `miimon`, which is off by default (set to 0), should be given a value. The `miimon` setting is the time period in milliseconds between checks of the interface carrier state. A common value is `miimon=100`.

When listing interfaces in `cmsh`, if an interface is a member of a bond or bridge interface, then the corresponding bonded or bridge interface name is shown in parentheses after the member interface name. Section 4.4, on configuring bridge interfaces, shows an example of such a listing from within `cmsh` on page 108.

More on bonded interfaces (including a detailed description of bonding options and modes) can be found at `http://www.kernel.org/doc/Documentation/networking/bonding.txt`.

4.7 Configuring InfiniBand Interfaces

On clusters with an InfiniBand interconnect, the InfiniBand Host Channel Adapter (HCA) in each node must be configured before it can be used. This section describes how to set up the InfiniBand service on the nodes for regular use. Setting up InfiniBand for booting and provisioning purposes is described in Chapter 6, while setting up InfiniBand for NFS is described in section 4.11.5.

4.7.1 Installing Software Packages

On a standard DirectMon cluster, the OFED (OpenFabrics Enterprise Distribution) packages that are part of the Linux base distribution are used. These packages provide RDMA implementations allowing high bandwidth/low latency interconnects on OFED hardware. The implementations can be used by InfiniBand hardware, and iWarp protocol hardware such as the hardware-accelerated RDMA over ethernet provided by Intel.

By default, all relevant OFED packages are installed on the head node and software images. It is possible to replace the distribution OFED with an OFED provided by the DirectMon repository or another custom version. The replacement can be for the entire cluster, or only for certain software images. Administrators may choose to switch to a different OFED version if the HCAs used are not supported by the distribution OFED version, or to increase performance by using an OFED version that has been optimized for a particular HCA. Installing the DirectMon OFED packages is covered in section 13.6.

If the InfiniBand network is enabled during installation, then the `rdma` script runs during the `init` stage of booting up for the enabled nodes. For SLES and Linux distributions based on versions prior to Red Hat 6, the `openibd` script is used instead of the `rdma` script.

The `rdma` or `openibd` script takes care of loading the relevant InfiniBand HCA kernel modules. When adding an InfiniBand network after installation, it may be necessary to use `chkconfig` manually to configure the `rdma` or `openibd` script to be run at boot-time on the head node and inside the software images.

4.7.2 Subnet Managers

Every InfiniBand subnet requires at least one subnet manager to be running. The subnet manager takes care of routing, addressing and initialization on the InfiniBand fabric. Some InfiniBand switches include subnet managers. However, on large InfiniBand networks or in the absence of a switch-hosted subnet manager, a subnet manager needs to be started on at least one node inside of the cluster. When multiple subnet managers are started on the same InfiniBand subnet, one instance will become the active subnet manager whereas the other instances will remain in passive mode. It is recommended to run 2 subnet managers on all InfiniBand subnets to provide redundancy in case of failure.

On a Linux machine that is not running DirectMon, an administrator sets a subnet manager service[1] to start at boot-time with a command such as: “`chkconfig opensm on`”. However, for clusters managed by DirectMon, a subnet manager is best set up using CMDaemon. There are two ways of setting CMDaemon to start up the subnet manager on a node at boot time:

1. by assigning a role.

   In `cmsh` this can be done with:

   ```
   [root@ddnmon61 ~]# cmsh -c "device roles <node>; assign subnet\
   manager; commit"
   ```

   where `<node>` is the name of a node on which it will run, for example: `ddnmon61`, `node001`, `node002`...

   In `cmgui` the subnet manager role is assigned by selecting a head node or regular node from the resources tree, and assigning it the “`Subnet Manager Role`” from the “`Roles`” tab.

2. by setting the service up. Services are covered more generally in section 4.12.

   In `cmsh` this is done with:

   Example

[1]usually `opensm`, but `opensmd` in SLES

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device services node001
[ddnmon61->device[node001]->services]% add opensm
[ddnmon61->device[node001]->services*[opensm*]]% set autostart yes
[ddnmon61->device[node001]->services*[opensm*]]% set monitored yes
[ddnmon61->device[node001]->services*[opensm*]]% commit
[ddnmon61->device[node001]->services[opensm]]%
```

In `cmgui` the subnet manager service is configured by selecting a head node or regular node from the resources tree, and adding the service to it.

When the head node in a cluster is equipped with an InfiniBand HCA, it is a good candidate to run as a subnet manager for smaller clusters.

On large clusters a dedicated node is recommended to run the subnet manager.

4.7.3 InfiniBand Network Settings

Although not strictly necessary, it is recommended that InfiniBand interfaces are assigned an IP address (i.e. IP over IB). First, a network object in the cluster management infrastructure should be created. The procedure for adding a network is described in section 4.3.2. The following settings are recommended as defaults:

Property	Value
`Name`	`ibnet`
`Domain name`	`ib.cluster`
`Type`	`internal`
`Base address`	`10.149.0.0`
`Netmask bits`	`16`
`MTU`	`65520`

Once the network has been created all nodes must be assigned an InfiniBand interface on this network. The easiest method of doing this is to create the interface for one node device and then to clone that device several times.

For large clusters, a labor-saving way to do this is using the `addinterface` command (section 4.8.1) as follows:

```
[root@ddnmon61 ~]# echo "device
addinterface -n node001..node150 physical ib0 ibnet 10.149.0.1
commit" | cmsh -x
```

When the head node is also equipped with an InfiniBand HCA, it is important that a corresponding interface is added and configured in the cluster management infrastructure.

Example

Assigning an IP address on the InfiniBand network to the head node:

```
[ddnmon61->device[ddnmon61]->interfaces]% add physical ib0
[ddnmon61->device[ddnmon61]->interfaces*[ib0*]]% set network ibnet
[ddnmon61->device[ddnmon61]->interfaces*[ib0*]]% set ip 10.149.255.254
[ddnmon61->device[ddnmon61]->interfaces*[ib0*]]% commit
```

As with any change to the network setup, the head node needs to be restarted to make the above change active.

4.7.4 Verifying Connectivity

After all nodes have been restarted, the easiest way to verify connectivity is to use the `ping` utility

Example

Pinging node015 while logged in to node014 through the InfiniBand interconnect:

```
[root@node014 ~]# ping node015.ib.cluster
PING node015.ib.cluster (10.149.0.15) 56(84) bytes of data.
64 bytes from node015.ib.cluster (10.149.0.15): icmp_seq=1 ttl=64
time=0.086 ms
...
```

If the ping utility reports that ping replies are being received, the InfiniBand is operational. The `ping` utility is not intended to benchmark high speed interconnects. For this reason it is usually a good idea to perform more elaborate testing to verify that bandwidth and latency are within the expected range.

The quickest way to stress-test the InfiniBand interconnect is to use the Intel MPI Benchmark (IMB), which is installed by default in `/cm/shared/apps/imb/current`. The `setup.sh` script in this directory can be used to create a template in a user's home directory to start a run.

Example

Running the Intel MPI Benchmark using `openmpi` to evaluate performance of the InfiniBand interconnect between `node001` and `node002`:

```
[root@ddnmon61 ~]# su - cmsupport
[cmsupport@ddnmon61 ~]$ cd /cm/shared/apps/imb/current/
[cmsupport@ddnmon61 current]$ ./setup.sh
[cmsupport@ddnmon61 current]$ cd ~/BenchMarks/imb/3.2.2
[cmsupport@ddnmon61 3.2.2]$ module load openmpi/gcc
[cmsupport@ddnmon61 3.2.2]$ module initadd openmpi/gcc
[cmsupport@ddnmon61 3.2.2]$ make -f make_mpi2
[cmsupport@ddnmon61 3.2.2]$ mpirun -np 2 -machinefile ../nodes\
 IMB-MPI1 PingPong
#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec
            0         1000         0.78         0.00
            1         1000         1.08         0.88
            2         1000         1.07         1.78
            4         1000         1.08         3.53
            8         1000         1.08         7.06
           16         1000         1.16        13.16
           32         1000         1.17        26.15
           64         1000         1.17        52.12
          128         1000         1.20       101.39
          256         1000         1.37       177.62
          512         1000         1.69       288.67
         1024         1000         2.30       425.34
         2048         1000         3.46       564.73
         4096         1000         7.37       530.30
         8192         1000        11.21       697.20
        16384         1000        21.63       722.24
        32768         1000        42.19       740.72
        65536          640        70.09       891.69
       131072          320       125.46       996.35
       262144          160       238.04      1050.25
       524288           80       500.76       998.48
      1048576           40      1065.28       938.72
      2097152           20      2033.13       983.71
      4194304           10      3887.00      1029.07


# All processes entering MPI_Finalize
```

To run on nodes other than `node001` and `node002`, the `../nodes` file must be modified to contain different hostnames. To perform other benchmarks, the `PingPong` argument should be omitted.

4.8 Configuring BMC (IPMI/iLO) Interfaces

DirectMon can initialize and configure the baseboard management controller (BMC) that may be present on devices. This ability can be set during the installation on the head node (section 2.3.7), or it can be set after installation as described in this section. The IPMI or iLO interface that is exposed by a BMC is treated in the cluster management infrastructure as a special type of network interface belonging to a device. In the most common setup a dedicated network (i.e. IP subnet) is created for BMC communication. The `10.148.0.0/16` network is used by default for BMC interfaces by DirectMon.

4.8.1 BMC Network Settings

The first step in setting up a BMC is to add the BMC network as a network object in the cluster management infrastructure. The procedure for adding a network is described in section 4.3.2. The following settings are recommended as defaults:

Property	Value
Name	bmcnet, ilonet, or ipminet
Domain name	bmc.cluster, ilo.cluster, or ipmi.cluster
External network	false
Base address	10.148.0.0
Netmask bits	16
Broadcast address	10.148.255.255

Once the network has been created, all nodes must be assigned a BMC interface on this network. The easiest method of doing this is to create the interface for one node device and then to clone that device several times.

For larger clusters this can be laborious, and a simple bash loop can be used to do the job instead:

```
[ddnmon61 ~]# for ((i=1; i<=150; i++)) do
echo "
device interfaces node$(printf '%03d' $i)
add bmc ipmi0
set network bmcnet
set ip 10.148.0.$i
commit"; done | cmsh -x     # -x usefully echoes what is piped into cmsh
```

The preceding loop can conveniently be replaced with the addinterface command, run from within the device mode of cmsh:

```
[ddnmon61 ~]# echo "
device
addinterface -n node001..node150 bmc ipmi0 bmcnet 10.148.0.1
commit" | cmsh -x
```

The help text for addinterface gives more details on how to use it.

In order to be able to communicate with the BMC interfaces, the head node also needs an interface on the BMC network. Depending on how the BMC interfaces are physically connected to the head node, the head node has to be assigned an IP address on the BMC network one way or another. There are two possibilities for how the BMC interface is physically connected:

- When the BMC interface is connected to the primary internal network, the head node should be assigned an alias interface configured with an IP address on the BMC network.
- When the BMC interface is connected to a dedicated physical network, the head node must also be physically connected to this network. A physical interface must be added and configured with an IP address on the BMC network.

Example

Assigning an IP address on the BMC network to the head node using an alias interface:

```
[ddnmon61->device[ddnmon61]->interfaces]% add alias eth0:0
[ddnmon61->device[ddnmon61]->interfaces*[eth0:0*]]% set network bmcnet
[ddnmon61->device[ddnmon61]->interfaces*[eth0:0*]]% set ip 10.\
148.255.254
[ddnmon61->device[ddnmon61]->interfaces*[eth0:0*]]% commit
[ddnmon61->device[ddnmon61]->interfaces[eth0:0]]%
Mon Dec 6 05:45:05 ddnmon61: Reboot required: Interfaces have been\
 modified
[ddnmon61->device[ddnmon61]->interfaces[eth0:0]]% quit
[root@ddnmon61 ~]# /etc/init.d/network restart
```

As with any change to the network setup, the head node needs to be restarted to make the above change active, although in this particular case restarting the `network` service would suffice.

4.8.2 BMC Authentication

The node-installer described in Chapter 6 is responsible for the initialization and configuration of the BMC interface of a device. In addition to a number of network-related settings, the node-installer also configures BMC authentication credentials. By default BMC interfaces are configured with username `ddn` and a random password that is generated during the installation of the head node. The values can be read with the "`get`" command of `cmsh` in `partition` mode.

For example, in `cmsh`, the current values of the BMC username and password for the entire cluster can be obtained and changed as follows:

Example

```
[ddnmon61]% partition use base
[ddnmon61->partition[base]]% get bmcusername
ddn
[ddnmon61->partition[base]]% get bmcpassword
Za4ohni1ohMa2zew
[ddnmon61->partition[base]]% set bmcusername bmcadmin
[ddnmon61->partition*[base*]]% set bmcpassword
enter new password: ******
retype new password: ******
[ddnmon61->partition*[base*]]% commit
[ddnmon61->partition[base]]%
```

In `cmgui`, selecting the cluster item in the resources pane, and then using the `Settings` tabbed pane, allows the BMC settings to be edited.

It is possible to change the BMC authentication credentials cluster-wide or by category. Category settings override cluster-wide settings. The relevant properties are:

Property	Description
`BMC User ID`	User type. Normally set to `4` for administrator access.
`BMC User Name`	User name
`BMC Password`	Password for specified user name

The DirectMon stores the configured BMC username and password. These are used:

- to configure the BMC interface from the node-installer
- to authenticate to the BMC interface after it has come up

BMC management operations, such as power cycling nodes and collecting hardware metrics, can then be performed.

4.8.3 Interfaces Settings

Interface Name

It is recommended that the network device name of a BMC interface start with `ipmi` or `ilo`, according to whether the BMC is running with IPMI or iLO. Numbers are appended to the base name, resulting in, for example: `ipmi0`.

Obtaining The IP address

BMC interfaces can have their IP addresses configured statically, or obtained from a DHCP server.

Only a node with a static BMC IP address has BMC power management done by DirectMon. A node with a DHCP-assigned BMC IP address, requires custom BMC power management (section 5.1.4) due to its dynamic nature.

4.9 Configuring Switches And PDUs

4.9.1 Configuring With The Manufacturer's Configuration Interface

Network switches and PDUs that will be used as part of the cluster should be configured with the PDU/switch configuration interface described in the PDU/switch documentation supplied by the manufacturer. Typically the interface is accessed by connecting via a web browser or telnet to an IP address preset by the manufacturer.

The IP settings of the PDU/switch must be configured to match the settings of the device in the cluster management software.

- In `cmgui` this is done by selecting the `Switches` resource, selecting the particular switch from within that resource, then selecting the associated `Settings` tab (figure 4.11). The IP address can then be set and saved.
- In `cmsh` this can be done in `device` mode, with a `set` command:

  Example

  ```
  [root@ddnmon61 ~]# cmsh
  [ddnmon61]% device
  [ddnmon61->device]% set switch01 ip 10.141.253.2
  [ddnmon61->device*]% commit
  ```

4.9.2 Configuring SNMP

Moreover, in order to allow the cluster management software to communicate with the switch or PDU, SNMP must be enabled and the SNMP community strings should be configured correctly. By default, the SNMP community strings for switches and PDUs are set to `public` and `private` for respectively read and write access. If different SNMP community strings have been configured in the switch or PDU, the `readstring` and `writestring` properties of the corresponding switch device should be changed.

Example

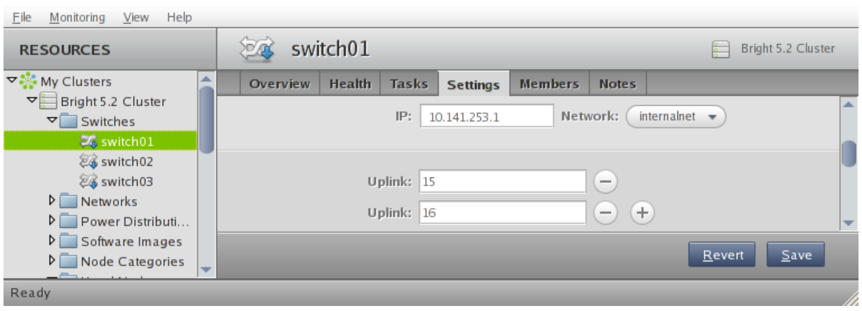
```
[ddnmon61]% device use switch01
[ddnmon61->device[switch01]]% get readstring
public
[ddnmon61->device[switch01]]% get writestring
private
[ddnmon61->device[switch01]]% set readstring public2
[ddnmon61->device*[switch01*]]% set writestring private2
[ddnmon61->device*[switch01*]]% commit
```

4.9.3 Uplink Ports

Uplink ports are switch ports that are connected to other switches. CMDaemon must be told about any switch ports that are uplink ports, or the traffic passing through an uplink port will lead to mistakes in what CMDaemon knows about port and MAC correspondence. Uplink ports are thus ports that CMDaemon is told to ignore.

To inform CMDaemon about what ports are uplink ports, `cmgui` or `cmsh` are used:

- In `cmgui`, the switch is selected from the `Switches` folder, and the `Settings` tabbed pane is opened (figure 4.11). The port number corresponding to uplink port number is filled in the blank field beside the "`Uplink:`" label. More uplinks can be appended by clicking on the ⊕ widget. The state is saved with the `Save` button.



IP: 10.141.253.1 Network: internalnet

Uplink: 15

Uplink: 16

Figure 4.11: Notifying CMDaemon About Uplinks With `cmgui`

- In `cmsh`, the switch is accessed from the `device` mode. The uplink port numbers can be appended one-by-one with the `append` command, or `set` in one go by using space-separated numbers.

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device
[ddnmon61->device]% set switch01 uplinks 15 16
[ddnmon61->device*]% set switch02 uplinks 01
[ddnmon61->device*]% commit
successfully committed 3 Devices
```

4.9.4 The showport MAC Address to Port Matching Tool

The showport command can be used in troubleshooting network topology issues, as well as checking and setting up new nodes (section 6.4.2).

Basic Use Of showport

In the device mode of cmsh is the showport command, which works out which ports on which switch are associated with a specified MAC address.

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device
[ddnmon61->device]% showport 00:30:48:30:73:92
[ddnmon61->device]% switch01:12
```

When running showport, CMDaemon on the head node queries all switches until a match is found.

If a switch is also specified using the "-s" option, then the query is carried out for that switch first. Thus the preceding example can also be specified as:

```
[ddnmon61->device]% showport -s switch01 00:30:48:30:73:92
[ddnmon61->device]% switch01:12
```

If there is no port number returned for the specified switch, then the scan continues on other switches.

Mapping All Port Connections In The Cluster With showport

A list indicating the port connections and switches for all connected devices that are up can be generated using this script:

Example

```
#!/bin/bash
for nodename in $(cmsh -c "device; foreach * (get hostname)")
do
  macad=$(cmsh -c "device use $nodename; get mac")
  echo -n "$macad $nodename "
  cmsh -c "device showport $macad"
done
```

The script may take a while to finish its run. It gives an output like:

Example

```
00:00:00:00:00:00 switch01: No ethernet switch found connected to this m\
ac address
00:30:48:30:73:92 ddnmon61: switch01:12
00:26:6C:F2:AD:54 node001: switch01:1
00:00:00:00:00:00 node002: No ethernet switch found connected to this ma\
c address
```

4.10 Disk Layouts: Disked, Semi-Diskless, And Diskless Node Configuration

Configuring the disk layout for head and regular nodes is done as part of the initial setup (section 2.3.14). For regular nodes, the disk layout can also be re-configured by DirectMon once the cluster is running. For a head node, however, the disk layout cannot be re-configured after installation by DirectMon, and head node disk layout reconfiguration must then therefore be treated as a regular Linux system administration task, typically involving backups and resizing partitions.

The remaining parts of this section on disk layouts therefore concern regular nodes, not head nodes.

4.10.1 Disk Layouts

A disk layout is specified using an XML schema (appendix D.1). The disk layout typically specifies the devices to be used, its partitioning scheme, and mount points. Possible disk layouts include the following:

- Default layout (appendix D.2)
- RAID setup (appendix D.3)
- LVM setup (appendix D.5)
- Diskless setup (appendix D.6)
- Semi-diskless setup (appendix D.7)

4.10.2 Disk Layout Assertions

Disk layouts can be set to *assert*

- that particular hardware be used, using XML element tags such as `vendor` or `requiredSize` (appendix D.8)
- custom assertions using an XML `assert` element tag to run scripts placed in CDATA sections (appendix D.9)

4.10.3 Changing Disk Layouts

A disk layout applied to a category of nodes is inherited by default by the nodes in that category. A disk layout that is then applied to an individual node within that category overrides the category setting. This is an example of the standard behavior for categories, as mentioned in section 3.1.3.

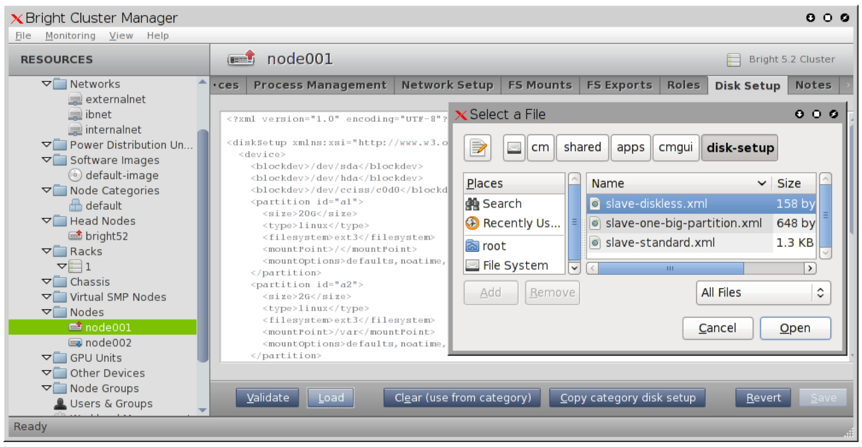
By default, the cluster is configured with a standard layout specified in section D.2. The layouts can be accessed from `cmgui` or `cmsh`, as is illustrated by the example in section 4.10.4, which covers changing a node from disked to diskless mode:

4.10.4 Changing A Disk Layout From Disked To Diskless

The XML schema for a node configured for diskless operation is shown in appendix D.6. This can often be deployed as is, or it can be modified during deployment using `cmgui` or `cmsh` as follows:

Changing A Disk Layout Using `cmgui`

To change a disk layout with `cmgui`, the current disk layout is accessed by selecting a node category or a specific node from the resource tree. The "`Disk Setup`" tab is then selected. Clicking on the `Load` button shows several possible configurations that can be loaded up, and if desired, edited to suit the situation (figure 4.12). To switch from the existing



```
<?xml version="1.0" encoding="UTF-8"?

<diskSetup xmlns:xsi="http://www.w3.o
  <device>
    <blockdev>/dev/sda</blockdev>
    <blockdev>/dev/hda</blockdev>
    <blockdev>/dev/cciss/c0d0</blockd
    <partition id="a1">
      <size>20G</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/</mountPoint>
      <mountOptions>defaults,noatime,
    </partition>
    <partition id="a2">
      <size>2G</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/var</mountPoint>
      <mountOptions>defaults,noatime,
    </partition>
```

Name	Size
slave-diskless.xml	158 by
slave-one-big-partition.xml	648 by
slave-standard.xml	1.3 KB

Figure 4.12: Changing A Disked Node To A Diskless Node With `cmgui`

disk layout to a diskless one, the diskless XML configuration is loaded and saved to the node or node category.

Changing A Disk Layout Using `cmsh`

To edit an existing disk layout from within `cmsh`, the existing XML configuration is accessed by editing the `disksetup` property in `device` mode for a particular node, or by editing the `disksetup` property in `category` mode for a particular category. Editing is done using the `set` command, which opens up a text editor:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device use node001
[ddnmon61->device[node001]]% set disksetup
```

After editing and saving the XML configuration, the change is then committed to CMDaemon with the `commit` command.

If the `disksetup` setting for a device is deleted, using the `clear` command, then the category level `disksetup` property is used by the device. This is in accordance with the usual behavior for node values that override category values (section 3.1.5).

Instead of editing an existing disk layout, another XML configuration can also be assigned. A diskless configuration may be chosen and set as follows:

Example

```
[ddnmon61->device[node001]]% set disksetup /cm/shared/apps/cmgui/disk-s\
etup/slave-diskless.xml
```

In these preceding cmgui and cmsh examples, after committing the change and rebooting the node, the node then functions entirely from its RAM, without using its own disk.

However, RAM is usually a scarce resource, so administrators often wish to optimize diskless nodes by freeing up the RAM on them from the OS that is using the RAM. Freeing up RAM can be accomplished by providing parts of the filesystem on the diskless node via NFS from the head node. That is, mounting the regular node with filesystems exported via NFS from the head node. The details of how to do this are a part of section 4.11, which covers the configuration of NFS exports and mounts in general.

4.11 Configuring NFS Volume Exports And Mounts

NFS allows Unix NFS clients shared access to a file system on an NFS server. The accessed file system is called an NFS volume by remote machines. The NFS server exports the filesystem to selected hosts or networks, and the clients can then mount the exported volume locally.

An unformatted filesystem cannot be used. The drive must be partitioned beforehand with `fdisk` or similar partitioning tools, and its filesystem formatted with `mkfs` or similar before it can be exported.

In DirectMon, the head node is typically used to export an NFS volume to the regular nodes, and the regular nodes then mount the volume locally.

If auto-mounting is used, then the configuration files for exporting should be set up on the NFS server, and the mount configurations set up on the software images. The service "`autofs`" or the equivalent can be set up using `cmgui` via the "`Services`" tab (section 4.12) on the head node.

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device use ddnmon61
[ddnmon61->device[ddnmon61]]% services
[ddnmon61->device[ddnmon61]->services]% add autofs
[ddnmon61->device*[ddnmon61*]->services*[autofs*]]% show
Parameter                      Value
------------------------------ -------------------------------------
Autostart                      no
Belongs to role                no
Monitored                      no
Revision
Run if                         ALWAYS
Service                        autofs
[ddnmon61->device*[ddnmon61*]->services*[autofs*]]% set autostart yes
[ddnmon61->device*[ddnmon61*]->services*[autofs*]]% commit
[ddnmon61->device[ddnmon61]->services[autofs]]% category use default
[ddnmon61->category[default]]% services
[ddnmon61->category[default]->services]% add autofs
[ddnmon61->category*[default*]->services*[autofs*]]% set autostart yes
[ddnmon61->category*[default*]->services*[autofs*]]% commit
[ddnmon61->category[default]->services[autofs]]%
```

The rest of this section describes the configuration of NFS for static mounts, using `cmgui` or `cmsh`.

Sections 4.11.1 and 4.11.2 explain how exporting and mounting of filesystems is done in general by an administrator using `cmgui` and `cmsh`, and considers some mounting behavior that the administrator should be aware of.

Section 4.11.3 discusses how filesystems in general on a diskless node can be replaced via mounts of NFS exports.

Section 4.11.4 discusses how the root (/) filesystem on a diskless node can be replaced via mounts of NFS exports.

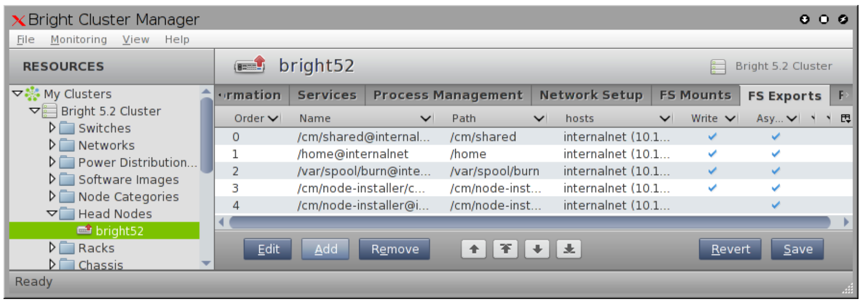
Section 4.11.5 discusses how OFED InfiniBand or iWarp drivers can be used to provide NFS over RDMA.

4.11.1 Exporting A Filesystem Using `cmgui` And `cmsh`

Exporting A Filesystem Using `cmgui`

As an example, if an NFS volume exists at "`ddnmon61:/modeldata`" it can be exported using `cmgui` as follows:

The head node is selected from under the "`Head Nodes`" resource, and the "`FS Exports`" tab is then selected. This shows the list of exports (figure 4.13).

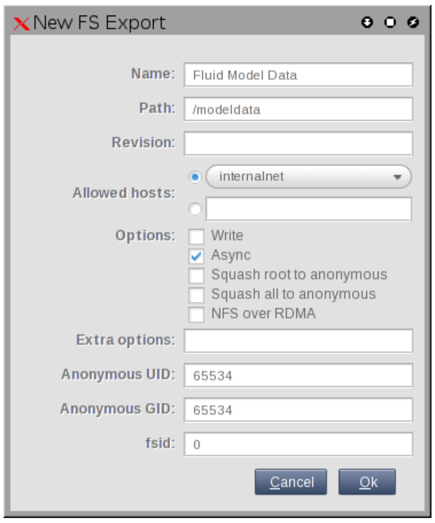


bright52

Order	Name	Path	hosts	Write	Asy...
0	/cm/shared@internal...	/cm/shared	internalnet (10.1...	✓	✓
1	/home@internalnet	/home	internalnet (10.1...	✓	✓
2	/var/spool/burn@inte...	/var/spool/burn	internalnet (10.1...	✓	✓
3	/cm/node-installer/c...	/cm/node-inst...	internalnet (10.1...	✓	✓
4	/cm/node-installer@i...	/cm/node-inst...	internalnet (10.1...		✓

Figure 4.13: NFS Exports From A Head Node Viewed Using `cmgui`

Using the `Add` button, a new entry (figure 4.14) can be configured with values:



Name:	Fluid Model Data
Path:	/modeldata
Revision:	
Allowed hosts:	● internalnet ○
Options:	☐ Write ☑ Async ☐ Squash root to anonymous ☐ Squash all to anonymous ☐ NFS over RDMA
Extra options:	
Anonymous UID:	65534
Anonymous GID:	65534
fsid:	0

Figure 4.14: Setting Up An NFS Export Using `cmgui`

For this example, the value for "Name" is set arbitrarily to "Fluid Model Data", the value for Path is set to /modeldata, and the value for "Allowed hosts" is set from the selection menu to allowing access to internalnet (this is by default 10.141.0.0/16 in CIDR notation).

By leaving the Write option disabled, read-only access is kept. Saving this means the NFS server now provides NFS access to this file system for internalnet.

The network can be set to other values using CIDR notation, and also to particular hosts such as just node001 and node002, by specifying a value of "node001 node002" instead. Other settings and options are also possible and are given in detail in the man pages for exports(5).

Exporting A Filesystem Using cmsh

The equivalent to the preceding cmgui NFS export procedure can be done in cmsh by using the fsexports submode on the head node (some output elided):

Example

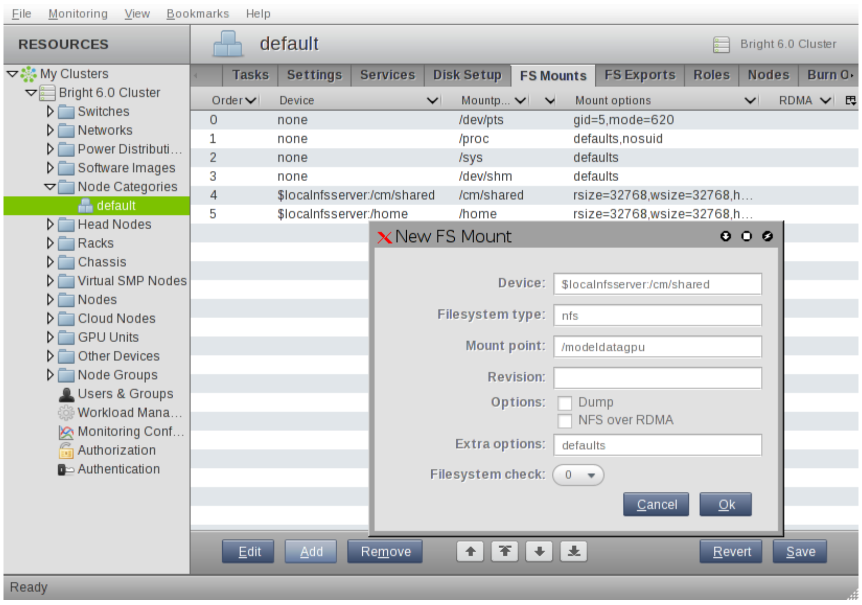
```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device use ddnmon61
[ddnmon61->device[ddnmon61]]% fsexports
[...->fsexports]% add "Fluid Model Data"
[...->fsexports*[Fluid Model Data*]]% set path /modeldata
[...[Fluid Model Data*]]% set hosts 10.141.0.0/16
[...[Fluid Model Data*]]% commit
[...->fsexports[Fluid Model Data]]% list | grep Fluid
Name (key)           Path           Hosts            Write
-------------------  -------------  ---------------  ------
Fluid Model Data     /modeldata     10.141.0.0/16    no
```

4.11.2 Mounting A Filesystem Using cmgui And cmsh

Continuing on with the export example from the preceding section, the administrator decides to mount the remote filesystem over the default category of nodes. Nodes can also mount the remote filesystem individually, but that is usually not a common requirement in a cluster. The administrator also decides not to re-use the exported name from the head node. That is, the remote mount name modeldata is not used locally, even though NFS allows this and many administrators prefer to do this. Instead, a local mount name of /modeldatagpu is used, perhaps because it avoids confusion about which filesystem is local to a person who is logged in, and perhaps to emphasize the volume is being mounted by nodes with GPUs.

Mounting A Filesystem Using cmgui

Thus, in cmgui, values for the remote mount point (ddnmon61:/modeldata), the file system type (nfs), and the local mount point (/modeldatagpu) can be set in category mode, while the remaining options stay at their default values (figure 4.15).



Order	Device	Mountp...		Mount options	RDMA
0	none	/dev/pts		gid=5,mode=620	
1	none	/proc		defaults,nosuid	
2	none	/sys		defaults	
3	none	/dev/shm		defaults	
4	$localnfsserver:/cm/shared	/cm/shared		rsize=32768,wsize=32768,h…	
5	$localnfsserver:/home	/home		rsize=32768,wsize=32768,h…	

New FS Mount

Device:	$localnfsserver:/cm/shared
Filesystem type:	nfs
Mount point:	/modeldatagpu
Revision:	
Options:	☐ Dump ☐ NFS over RDMA
Extra options:	defaults
Filesystem check:	0

Cancel Ok

Figure 4.15: Setting Up NFS Mounts On A Node Category Using `cmgui`

Clicking on the `Ok` button saves the values, creating the local mount point, and the volume can now be accessed by nodes in that category.

Mounting A Filesystem Using `cmsh`

The equivalent to the preceding `cmgui` NFS mount procedure can be done in `cmsh` by using the `fsmounts` submode, for example on the `default` category. The `add` method under the `fsmounts` submode sets the mountpoint path, in this case `/modeldatagpu` (some output elided):

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% category use default
[ddnmon61->category[default]]% fsmounts
[ddnmon61->category[default]->fsmounts]% add /modeldatagpu
[ddnmon61->...*[/modeldatagpu*]]% set device ddnmon61:/modeldata
[ddnmon61->...*[/modeldatagpu*]]% set filesystem nfs
[ddnmon61->category*[default*]->fsmounts*[/modeldatagpu*]]% commit
[ddnmon61->category[default]->fsmounts[/modeldatagpu]]%
Device                   Mountpoint (key)     Filesystem
---------------------    ------------------   ----------
...
ddnmon61:/modeldata      /modeldatagpu        nfs
[ddnmon61->category[default]->fsmounts[/modeldatagpu]]% show
Parameter              Value
-------------------    ---------------------
Device                 ddnmon61:/modeldata
Dump                   no
Filesystem             nfs
Filesystem Check       0
Mount options          defaults
Mountpoint             /modeldatagpu
```

Values can be `set` for “`Mount options`” other than default. For example, the “`noac`” flag can be added as follows:

```
[ddnmon61->...[/modeldatagpu]]% set mountoptions defaults,noac; commit
```

General Considerations On Mounting A Filesystem

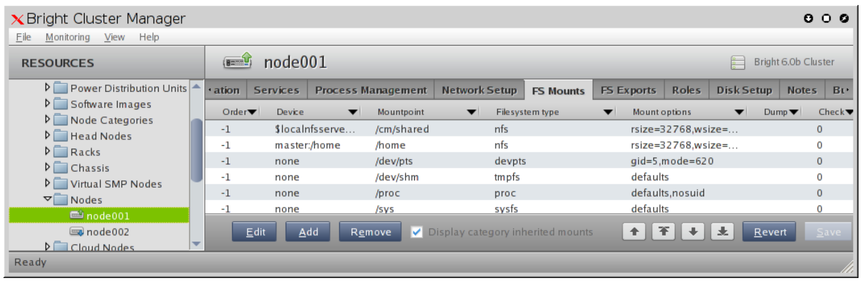
There may be a requirement to segregate the access of nodes. For example, in the case of the preceding, because some nodes have no associated GPUs.

Besides the "`Allowed hosts`" options of NFS exports mentioned earlier in section 4.11.1, DirectMon offers two more methods to fine tune node access to mount points:

- Nodes can be placed in another category that does not have the mount point.
- Nodes can have the mount point set, not by category, but per device within the `Nodes` resource. For this, the administrator must ensure that nodes that should have access have the mount point explicitly set.

Other considerations on mounting are that:

- When adding a mount point object:
  - The settings take effect right away by default on the nodes or node categories.
  - If `noauto` is set as a mount option, then the option only takes effect on explicitly mounting the filesystem.
  - If "`AutomaticMountAll=0`" is set as a CMDaemon option, then `/etc/fstab` is written, but the `mount -a` command is not run by CMDaemon. However, since `mount -a` is run by the distribution init script system during boot, a reboot of the node will implement a mount change.
- While a mount point object may have been removed, umount does not take place until reboot, to prevent mount changes outside of the cluster manager. If a umount needs to be to done without a reboot, then it should be done manually, for example, using the `pexec` command (section 12.1), to allow the administrator to take appropriate action if umounting goes wrong.
- When manipulating mount points, the administrator should be aware which mount points are inherited by category, and which are set for the individual node.
  - In `cmgui`, viewing category-inherited mount points for an individual node requires checking the checkbox for "`Display category inherited mounts`" in the `FS Mounts` tabbed view for that node (figure 4.16).



Bright Cluster Manager — node001 — Bright 6.0b Cluster — FS Mounts

Order	Device	Mountpoint	Filesystem type	Mount options	Dump	Check
-1	$localnfsserve...	/cm/shared	nfs	rsize=32768,wsize=...		0
-1	master:/home	/home	nfs	rsize=32768,wsize=...		0
-1	none	/dev/pts	devpts	gid=5,mode=620		0
-1	none	/dev/shm	tmpfs	defaults		0
-1	none	/proc	proc	defaults,nosuid		0
-1	none	/sys	sysfs	defaults		0

✓ Display category inherited mounts

Figure 4.16: Display Of Category Inherited Mounts In `cmgui`

– In `cmsh`, the category a mount belongs to is displayed in brackets. This is displayed from within the `fsmounts` submode of the `device` mode for a specified node:

Example

```
[root@ddnmon61 ~]# cmsh -c "device; fsmounts node001; list"

Device                    Mountpoint (key)     Filesystem
------------------------- -------------------- ----------
[default] none            /dev/pts             devpts
[default] none            /proc                proc
[default] none            /sys                 sysfs
[default] none            /dev/shm             tmpfs
[default] $localnfsserv+  /cm/shared           nfs
[default] ddnmon61:/home  /home                nfs
ddnmon61:/cm/shared/exa+  /home/examples       nfs
[root@ddnmon61 ~]#
```

To remove a mount point defined at category level for a node, it must be removed from within the category, and not from the specific node.

Mount Order Considerations

Care is sometimes needed in deciding the order in which mounts are carried out.

- For example, if both `/usr/share/doc` and a replacement directory subtree `/usr/share/doc/compat-gcc-34-3.4.6java` are to be used, then the stacking order should be that `/usr/share/doc` is mounted first. This order ensures that the replacement directory subtree overlays the first mount. If, instead, the replacement directory were the first mount, then it would be overlaid, inaccessible, and inactive.

- There may also be dependencies between the subtrees to consider, some of which may prevent the start up of applications and services until they are resolved. In some cases, resolution may be quite involved.

The order in which such mounts are mounted can be modified with the `up` and `down` commands within the `fsmounts` submode of `cmsh`, or by using the arrow buttons at the bottom of the `FS Mounts` tabbed pane.

4.11.3 Mounting A Filesystem Subtree For A Diskless Node Over NFS

NFS Vs `tmpfs` For Diskless Nodes

For diskless nodes (appendix D.6), the software image (section 3.1.2) is typically installed from a provisioning node by the node-installer during the provisioning stage, and held as a filesystem in RAM on the diskless node with the `tmpfs` filesystem type.

It can be worthwhile to replace subtrees under the diskless node filesystem held in RAM with subtrees provided over NFS. This can be particularly worthwhile for less frequently accessed parts of the diskless node filesystem. This is because, although providing the files over NFS is much slower than accessing it from RAM, it has the benefit of freeing up RAM for tasks and jobs that run on diskless nodes, thereby increasing the cluster capacity.

An alternative "semi-diskless" way to free up RAM is to use a local disk on the node itself for supplying the subtrees. This is outlined in appendix D.7.

Moving A Filesystem Subtree Out Of `tmpfs` To NFS

To carry out subtree provisioning over NFS, the subtrees are exported and mounted using the methods outlined in the previous examples in sections 4.11.1 and 4.11.2. For the diskless case, the exported filesystem subtree is thus a particular path under `/cm/images/<image>`[2] on the provisioning node, and the subtree is mounted accordingly under `/` on the diskless node.

While there are no restrictions placed on the paths that may be mounted in DirectMon™, the administrator should be aware that mounting certain paths such as `/bin` is not possible.

When `cmgui` or `cmsh` are used to manage the NFS export and mount of the subtree filesystem, then `tmpfs` on the diskless node is reduced in size due to the administrator explicitly excluding the subtree from `tmpfs` during provisioning.

An example might be to export `/cm/images/default-image` from the head node, and mount the directory available under it, `usr/share/doc`, at a mount point `/usr/share/doc` on the diskless node. In `cmsh`, such an export can be done by creating an FS export object corresponding to the software image object `defaultimage` with the following indicated properties (some prompt output elided):

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device use ddnmon61; fsexports
[ddnmon61->device[ddnmon61]->fsexports]% add defaultimage
[ddnmon61...defaultimage*]]% set path /cm/images/default-image
[ddnmon61...defaultimage*]]% set hosts 10.141.0.0/16
[ddnmon61...defaultimage*]]% commit
[ddnmon61...defaultimage]]% list | grep defaultimage
Name (key)        Path                         Hosts          Write
----------------  ------------------------  -------------  -----
defaultimage      /cm/images/default-image  10.141.0.0/16  no
```

[2]by default *<image>* is `default-image` on a newly-installed cluster

As the output to `list` shows, the NFS export should be kept read-only, which is the default. Appropriate parts of the export can then be mounted by a node or node category. The mount is defined by setting the mount point, the `nfs` filesystem property, and the export device. For example, for a node category (some output elided):

```
[br...defaultimage]]% category use default
[ddnmon61->category[default]]% fsmounts
[ddnmon61->category[default]->fsmounts]% add /usr/share/doc
[ddnmon61->...*[/usr/share/doc*]]% set device ddnmon61:/cm/images/defau\
lt-image/user/share/doc
[ddnmon61->...*[/usr/share/doc*]]% set filesystem nfs
[ddnmon61->category*[default*]->fsmounts*[/usr/share/doc*]]% commit
[ddnmon61->category[default]->fsmounts[/usr/share/doc]]% list
Device                            Mountpoint (key)     Filesystem
---------------------             ------------------   ----------
...                               ...                  ...
ddnmon61:/cm/images/usr/share/doc /usr/share/doc       nfs
[ddnmon61->category[default]->fsmounts[/usr/share/doc]]% show
Parameter          Value
----------------   -----------------------------------------------
Device             ddnmon61:/cm/images/default-image/usr/share/doc
Dump               no
Filesystem         nfs
Filesystem Check   0
Mount options      defaults
Mountpoint         /usr/share/doc
```

Other mount points can be also be added according to the judgment of the system administrator. Some consideration of mount order may be needed, as discussed on page 130 under the subheading “Mount Order Considerations”.

An Example Of Several NFS Subtree Mounts

The following mounts save about 500MB from `tmpfs` on a diskless node with CentOS 6, as can be worked out from the following subtree sizes:

```
[root@ddnmon61 ~]# cd /cm/images/default-image/
[root@ddnmon61 default-image]# du -sh usr/share/locale usr/java usr/sha\
re/doc usr/src
262M    usr/share/locale
78M     usr/java
107M    usr/share/doc
45M     usr/src
```

The filesystem mounts can then be created using the techniques in this section. After doing that, the result is then something like (some lines omitted):

```
[root@ddnmon61 default-image]# cmsh
[ddnmon61]% category use default; fsmounts
[ddnmon61->category[default]->fsmounts]% list -f device:53,mountpoint:17
device                                                mountpoint (key)
---------------------------------------------------   -----------------
...                                                   ...
master:/cm/shared                                     /cm/shared
```

```
master:/home                                          /home
ddnmon61:/cm/images/default-image/usr/share/locale  /usr/share/locale
ddnmon61:/cm/images/default-image/usr/java          /usr/java
ddnmon61:/cm/images/default-image/usr/share/doc     /usr/share/doc
ddnmon61:/cm/images/default-image/usr/src           /usr/src
[ddnmon61->category[default]->fsmounts]%
```

Diskless nodes that have NFS subtree configuration carried out on them can be rebooted to start them up with the new configuration.

4.11.4 Mounting The Root Filesystem For A Diskless Node Over NFS

Mounting the root filesystem over NFS is a special case of mounting filesystem subtrees for a diskless node over NFS (section 4.11.3). The difference this time is that an initial root file system is deployed on the node via NFS as part of the standard Linux boot procedure. Some `tmpfs` mounts are then overlaid on top of parts of this filesystem.

The node being configured must have a disk setup that is diskless (section 4.10.4) for its node or node category.

If configuring a node category, it can be configured as follows on a Centos 6 system:

1. The full diskless disk setup is partitioned first:

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% category use default
[ddnmon61->category[default]]% set disksetup /cm/shared/apps/cmgui/disk
-setup/slave-diskless.xml
[ddnmon61->category*[default*]]% commit   #puts full OS in RAM
```

2. Then the default image is exported to the internal network:

```
[ddnmon61->category[default]]% device use master; fsexports
[ddnmon61->device[ddnmon61]->fsexports]% add /cm/images/default-image
...*]->fsexports*[/cm/images/default-image*]]% set hosts internalnet
...*]->fsexports*[/cm/images/default-image*]]% set name default@internal
...*]->fsexports*[default@internal*]]% commit   #exports default image
[ddnmon61->device[ddnmon61]->fsexports[default@internal]]% quit
```

3. The exported root filesystem is now mounted over the initial root filesystem:

```
[root@ddnmon61 ~]# cmsh -c "category use default; fsmounts; add /; set\
 device master:/cm/images/default-image; set mountoptions defaults,ro;\
 set filesystem nfs; commit"        #adds readonly root via nfs
```

4. The filesystem for / however has plenty of parts that need to be writable too. These are now mounted back in minimal subtrees into RAM. Keeping these mounts minimal means that the RAM used by the node is minimal. These read/write subtrees can be specified in a file, with one line per subtree:

```
[root@ddnmon61 ~]# cat mountfiles      #for centos
/var/cache
/var/log
/var/lock
/var/run
/var/tmp
```

```
/var/spool
/tmp
/dev
/cm/local/apps/cmd/etc
/cm/local/apps/openldap/etc
/cm/local/apps/sge/var
/cm/local/apps/torque/var
/cm/local/apps/slurm/var
/cm/local/apps/pbspro/var
/cm/local/apps/openlava/var
/etc
```

The `/cm/local/apps/openlava/var` directory does not exist by default, and so it should only be placed in `mountfiles` if the `openlava` software has been installed (section 9.5.5). The various workload manager `/var` directories for SGE/Torque/Slurm/PBS Pro/openlava are only needed if these workload managers are to be setup and used.

The subtrees can be mounted with a `for` loop:

```
[root@ddnmon61 ~]# (echo "category use default; fsmounts"
  for i in $(<mountfiles)
  do
    echo "add $i; set device tmpfs; set filesystem tmpfs; exit"
  done
  echo "commit" ) | cmsh
```

If there is further software in the root filesystem that needs write access, that too should be mounted back into RAM.

The diskless nodes cannot be powered off with a simple `poweroff` or rebooted with a `reboot`. This is because the parts of the filesystem required to carry out these actions are unmounted before they are called in the diskless configuration. The `-f|-force` option to these commands forces a `poweroff` or `reboot`, but should only be used after first cleanly unmounting any writable shared filesystems, such as `/cm/shared` and `/home`. This is because the forcing options interrupt I/O syncing when invoked, which can result in a corrupted filesystem.

4.11.5 Configuring NFS Volume Exports And Mounts Over RDMA With OFED Drivers

Drivers To Use For NFS over RDMA Must Be From The Parent Distribution

The use of the RDMA protocol (section 4.7) to provide NFS, by installing updated DirectMon OFED drivers (section 13.6) is currently not supported. This is because these drivers are packaged by DDN from the vendor (Mellanox or Qlogic) releases, and the vendor releases themselves do not support NFS over RDMA.

However, NFS over RDMA does work within DirectMon with the OFED drivers that the parent distribution provides, in the case of RHEL6/SL6/CENTOS6. For the remaining distributions, this option can be selected, but NFS will fall back to using the default NFS TCP/IP protocol.

When using NFS over RDMA, `ibnet`, the IP network used for InfiniBand, should be set. Section 4.7.3 explains how that can be done.

Exporting With `cmgui` And `cmsh` Using NFS Over RDMA

With the drivers installed, a volume export can be carried out using NFS over RDMA.

The procedure using cmgui is much the same as done in section 4.11.1 ("Exporting A Filesystem Using cmgui"), except for that the ibnet network should be selected instead of the default internalnet, and the "NFS over RDMA" checkbox should be ticked.

The procedure using cmsh is much the same as done in section 4.11.1 ("Exporting A Filesystem Using cmsh"), except that the ibnet network (normally with a recommended value of 10.149.0.0/16) should be set, and the rdma option should be set.

Example

(based on the example in section 4.11.1)

```
...
[...->fsexports*[Fluid Model Data*]]% set path /modeldata
[...[Fluid Model Data*]]% set hosts ibnet
[...[Fluid Model Data*]]% set rdma yes
[...[Fluid Model Data*]]% commit
...
```

Mounting With cmgui And cmsh Using NFS Over RDMA

The mounting of the exported file systems using NFS over RDMA can then be done.

The procedure using cmgui is largely like that in section 4.11.2, ("Mounting A Filesystem Using cmgui"), except that the Device entry must point to master.ib.cluster so that it resolves to the correct NFS server address for RDMA, and the checkbox for NFS over RDMA must be ticked.

The procedure using cmsh is similar to that in section 4.11.2, ("Mounting A Filesystem Using cmsh"), except that device must be mounted to the ibnet, and the rdma option must be set, as shown:

Example

(based on the example in section 4.11.2)

```
...
[ddnmon61->category[default]->fsmounts]% add /modeldatagpu
[ddnmon61->...*[/modeldatagpu*]]% set device ddnmon61.ib.cluster:/modeld\
ata
[ddnmon61->...*[/modeldatagpu*]]% set filesystem nfs
[ddnmon61->...*[/modeldatagpu*]]% set rdma yes
[ddnmon61->category*[default*]->fsmounts*[/modeldatagpu*]]% commit
...
```

4.12 Managing And Configuring Services

4.12.1 Why Use The Cluster Manager For Services?

Unix services can be managed from the command line using the standard distribution tools, chkconfig and /etc/init.d/<*service name*>, where <*service name*> indicates a service such as mysql, nfs, postfix and so on.

Already installed services can also be started and stopped using DirectMon's cmgui and cmsh tools.

An additional convenience that comes with the cluster manager tools is that some parameters useful for cluster management are very easily configured, whether on the head node, a regular node, or for a node category. These parameters are:

- `monitored`: checks periodically if a service is running. Information is displayed and logged the first time it starts or the first time it dies
- `autostart`: restarts a failed service that is being `monitored`.
  - If a service is removed using DirectMon, then the service is first made to stop if `autostart` is set to `on`.
  - A service that has not been stopped by CMDaemon undergoes an attempt to restart it even if `autostart` is off, if
    - CMDaemon is restarted
    - its configuration files are updated by CMDaemon, for example in other modes, as in the example on page 100.
- `runif`: (only honored for head nodes) whether the service should run with a state of:
  - `active`: run on the active node only
  - `passive`: run on the passive only
  - `always`: run both on the active and passive
  - `preferpassive`: preferentially run on the passive if it is available

The details of a service configuration remain part of the configuration methods of the service software itself. DirectMon configuration handles general services only at the generic service level to which all Unix services conform.

DirectMon can be used to keep a service working across a failover event with an appropriate `runif` setting and appropriate failover scripts such as the `Prefailover script` and the `Postfailover script` (section 15.4.6). The details of how to do this will depend on the service.

4.12.2 Managing And Configuring Services—Examples

If, for example, the CUPS software is installed ("`yum install cups`"), then the CUPS service can be managed in several ways:

Managing The Service From The Regular Shell, Outside Of CMDaemon

Standard Unix commands from the bash prompt work, as shown by this session:

```
[root@ddnmon61 ~]# chkconfig cups on
[root@ddnmon61 ~]# /etc/init.d/cups start
```

Managing The Service From `cmsh`

Starting the service in `cmsh`: The following session illustrates adding the CUPS service from within `device` mode and the `services` submode. The device in this case is a regular node, `node001`, but a head node can also be chosen. Monitoring and auto-starting are also set in the session:

```
[ddnmon61]% device services node001
[ddnmon61->device[node001]->services]% add cups
[ddnmon61->device*[node001*]->services*[cups*]]% show
Parameter                      Value
------------------------------ -----------------------------------
Autostart                      no
Belongs to role                no
Monitored                      no
Revision
Run if                         ALWAYS
Service                        cups
[ddnmon61->device*[node001*]->services*[cups*]]% set monitored on
[ddnmon61->device*[node001*]->services*[cups*]]% set autostart on
[ddnmon61->device*[node001*]->services*[cups*]]% commit
[ddnmon61->device[node001]->services[cups]]%
Tue Jul 26 16:42:17 2011 [notice] node001: Service cups was started
[ddnmon61->device[node001]->services[cups]]%
```

Other service options in `cmsh`: Within `cmsh`, the `start`, `stop`, `restart`, and `reload` options to the "`/etc/init.d/`*`<service name>`*" command can be used to manage the service at the `services` submode level. For example, continuing with the preceding session, stopping the CUPS service can be done by running the `cups` service command with the `stop` option as follows:

```
[ddnmon61->device[node001]->services[cups]]% stop
Wed Jul 27 14:42:41 2011 [notice] node001: Service cups was stopped
Successfully stopped service cups on: node001
[ddnmon61->device[node001]->services[cups]]%
```

The service is then in a `DOWN` state according to the `status` command.

```
[ddnmon61->device[node001]->services[cups]]% status
cups                [DOWN]
```

Details on how a state is used when monitoring a service are given in the section "Monitoring A Service With `cmsh` And `cmgui`" on page 139.

The CUPS service can also be added for a node category from `category` mode:

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% category services default
[ddnmon61->category[default]->services]% add cups
```

As before, after adding the service, the monitoring and autostart parameters can be set for the service. Also as before, the options to the "`init.d/`*`<service name>`*" command can be used to manage the service at the `services` submode level. The settings apply to the entire node category:

Example

```
[ddnmon61->category*[default*]->services*[cups*]]% show
[ddnmon61->category*[default*]->services*[cups*]]% set autostart yes
[ddnmon61->category*[default*]->services*[cups*]]% set monitored yes
[ddnmon61->category*[default*]->services*[cups*]]% commit
```

```
[ddnmon61->category[default]->services[cups]]%
Tue Aug 23 12:23:17 2011 [notice] node001: Service cups was started
Tue Aug 23 12:23:17 2011 [notice] node002: Service cups was started
Tue Aug 23 12:23:17 2011 [notice] node003: Service cups was started
Tue Aug 23 12:23:17 2011 [notice] node004: Service cups was started
[ddnmon61->category[default]->services[cups]]% status
cups             [   UP   ]
```

Managing The Service From `cmgui`

Using `cmgui`, a service can be managed from a `Services` tab. The tab is accessible from a "`Head Nodes`" item, from a "`Node Categories`" item, or from a `Nodes` item.

Figure 4.17 shows the `Services` tab accessed from the `default` software image, which is an item within the "`Node Categories`" folder.

The "`init.d/`*`<service name>`*" command options `start`, `stop`, `reload`, and so on, are displayed as buttons in the `Services` tab.

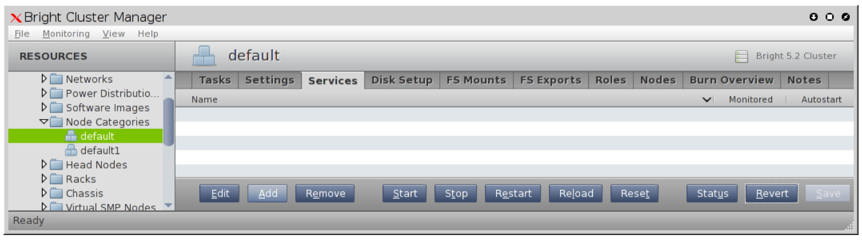
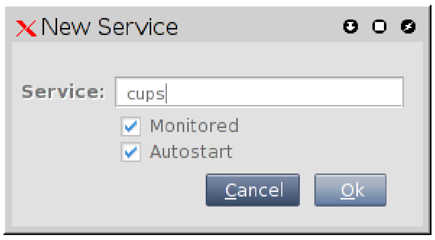


Figure 4.17: A `Services` Tab In `cmgui`

The existence of the service itself can be managed using the `Edit`, `Add`, and `Remove` buttons. The change can be saved or reverted with the `Save` and `Revert` buttons.

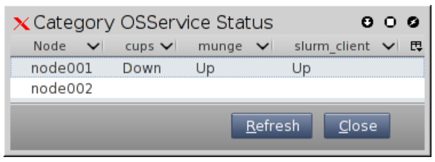
Figure 4.18 shows CUPS being set up from an `Add` dialog in the `Services` tab, accessed by clicking on the `Add` button in the `Services` tab of figure 4.17.



Service: cups
✓ Monitored
✓ Autostart

Figure 4.18: Setting Up A Service Using `cmgui`

For a service in the `Services` tab, clicking on the `Status` button in figure 4.17 displays a grid of the state of services on a running node as either `Up` or `Down` (figure 4.19). If the node is not running, it shows blank service entries.



Node	cups	munge	slurm_client
node001	Down	Up	Up
node002			

Figure 4.19: Displaying The Status Of Services Across Nodes Using `cmgui`

Monitoring A Service With `cmsh` And `cmgui`

The service is in a `DOWN` state if it is not running, and in a `FAILING` state if it is unable to run after 10 auto-starts in a row. Event messages are sent during these first 10 auto-starts. After the first 10 auto-starts, no more event messages are sent, but autostart attempts continue.

In case an autostart attempt has not yet restarted the service, the `reset` option may be used to attempt an immediate restart. The `reset` option is not a service option in the regular shell, but is used by CMDaemon (within `cmsh` and `cmgui`) to clear a `FAILING` state of a service, reset the attempted auto-starts count to zero, and attempt a restart of the service.

The monitoring system sets the `ManagedServicesOk` healthcheck (Appendix H.2.1) to a state of `FAIL` if any of the services it monitors is in the `FAILING` state. In `cmsh`, the statuses of the services are listed by running the `latesthealthdata` command (section 11.8.1) from `device` mode. In `cmgui` the failed services themselves are listed from the `Overview` tab, and the statuses of the services are listed in the `Services` tab.

Standard init.d script behavior is that the script return a non-zero exit code if the service is down, and a zero exit code if the service is up. A non-zero exit code makes DirectMon decide that the service is down, and should be restarted.

However, some scripts return a non-zero exit code even if the service is up. These services therefore have DirectMon attempt to start them repetitively, even though they are actually running.

This behavior is normally best fixed by setting a zero exit code for when the service is up, and a non-zero exit code for when the service is down.

Removing A Service From CMDaemon Control Without Shutting It Down

Removing a service from DirectMon control while `autostart` is set to `on` stops the service on the nodes:

```
[ddnmon61->category[default]->services]% add cups
[ddnmon61->category*[default*]->services*[cups*]]% set monitored on
[ddnmon61->category*[default*]->services*[cups*]]% set autostart on
[ddnmon61->category*[default*]->services*[cups*]]% commit; exit
[ddnmon61->category[default]->services]% remove cups; commit
Wed May 23 12:53:58 2012 [notice] node001: Service cups was stopped
```

In the preceding `cmsh` session, `cups` starts up when the `autostart` parameter is committed, if `cups` is not already up.

The behavior of having the service stop on removal is implemented because it is usually what is wanted.

However, sometimes the administrator would like to remove the service from CMDaemon control without it shutting down. To do this, autostart must be set to off first.

```
[ddnmon61->category[default]->services]% add cups
[ddnmon61->category*[default*]->services*[cups*]]% set monitored on
[ddnmon61->category*[default*]->services*[cups*]]% set autostart off
[ddnmon61->category*[default*]->services*[cups*]]% commit; exit
Wed May 23 12:54:40 2012 [notice] node001: Service cups was started
[ddnmon61->category[default]->services]% remove cups; commit
[ddnmon61->category[default]->services]% !# no change: cups stays up
```

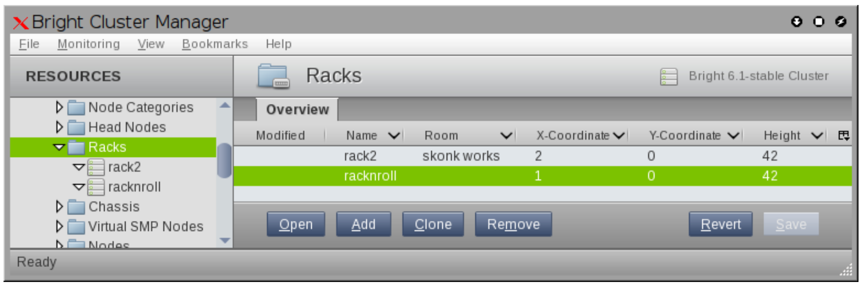
4.13 Managing And Configuring A Rack

4.13.1 Racks

A cluster may have local nodes grouped physically into racks. A rack is 42 units in height by default, and nodes normally take up one unit.

Racks Overview

Racks overview in cmgui: Selecting the Racks resource in cmgui opens up the Overview tabbed pane (figure 4.20). Racks can then be added, removed, or edited from that pane.



Modified	Name	Room	X-Coordinate	Y-Coordinate	Height
	rack2	skonk works	2	0	42
	racknroll		1	0	42

Figure 4.20: Racks Resource Overview Using cmgui

Within the Overview tabbed pane:

- a new rack item can be added with the Add button.
- an existing rack item can be edited by selecting it and clicking on the Open button, or by double clicking on the item itself in the tabbed pane.

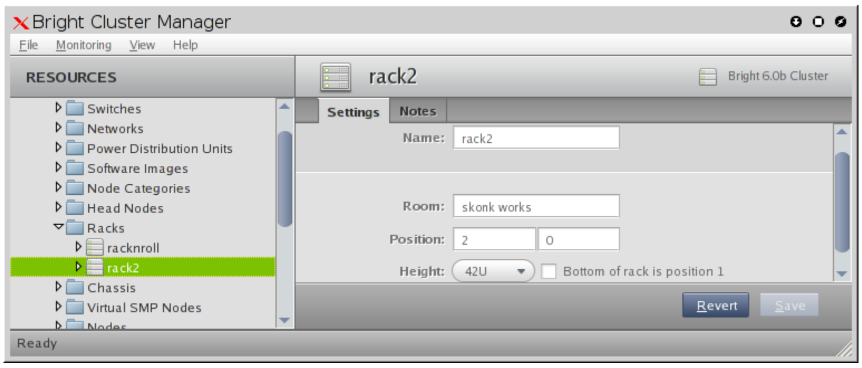
An existing rack item can also be edited by simply selecting its name in the resource tree pane. This brings up its Settings tabbed pane (figure 4.21) by default.

Racks overview in cmsh: The rack mode in cmsh allows racks defined in the cluster manager to be listed:

```
[ddnmon61->rack]% list
Name (key)       Room           x-Coordinate   y-Coordinate   Height
---------------  -------------- -------------- -------------- ------
rack2            skonk works    2              0              42
racknroll                       1              0              42
```

Rack Configuration Settings

Rack configuration settings in `cmgui`**:** The `Settings` tabbed pane for editing a rack item selected from the `Racks` resource is shown in figure 4.21.



Name: rack2
Room: skonk works
Position: 2 0
Height: 42U
Bottom of rack is position 1

Figure 4.21: Rack Configuration Settings Using `cmgui`

The rack item configuration settings are:

- `Name`: A unique name for the rack item. Names such as `rack001`, `rack002` are a sensible choice
- `Room`: A unique name for the room the rack is in.
- `Position`: The *x*- and *y*-coordinates of the rack in a room. In the rack view visualization (section 4.13.2), the racks are displayed in one row, with nodes ordered so that the lowest *x*-value is the leftmost node for nodes with the same *y*-value. These coordinates are meant to be a hint for the administrator about the positioning of the racks in the room, and as such are optional, and can be arbitrary. The `Notes` tabbed pane can be used as a supplement or as an alternative for hints.
- `Height`: by default this is the standard rack size of 42U.
- `Bottom of rack is position 1`: Normally, a rack uses the number 1 to mark the top and 42 to mark the bottom position for the places that a device can be positioned in a rack. However, some manufacturers use 1 to mark the bottom instead. Ticking the checkbox displays the numbering layout accordingly for all racks in `Rackview` (section 4.13.2), if the checkboxed rack is the first rack seen in `Rackview`.

Rack configuration settings in `cmsh`**:** In `cmsh`, tab-completion suggestions for the `set` command in `rack` mode display the racks available for configuration. On selecting a particular rack (for example, `rack2` as in the following example), tab-completion suggestions then display the configuration settings available for that rack:

Example

```
[ddnmon61->rack]% set rack
rack1   rack2   rack3
```

```
[ddnmon61->rack]% set rack2
height          name            revision        width           y-coordinate
inverted        notes           room            x-coordinate
```

The configuration settings for a particular rack obviously match with the parameters associated with and discussed in figure 4.21. The only slightly unobvious match is the Boolean parameter `inverted` in `cmsh`, which simply corresponds directly to "`Bottom of rack is position 1`" in `cmgui`.

Setting the values can be done as in this example:

Example

```
[ddnmon61->rack]% use rack2
[ddnmon61->rack[rack2]]% set room "skonk works"
[ddnmon61->rack*[rack2*]]% set x-coordinate 2
[ddnmon61->rack*[rack2*]]% set y-coordinate 0
[ddnmon61->rack*[rack2*]]% set inverted no
[ddnmon61->rack*[rack2*]]% commit
[ddnmon61->rack[rack2]]%
```

4.13.2 Rack View

The `Rackview` tab is available within `cmgui` after selecting the cluster resource item (figure 4.22).

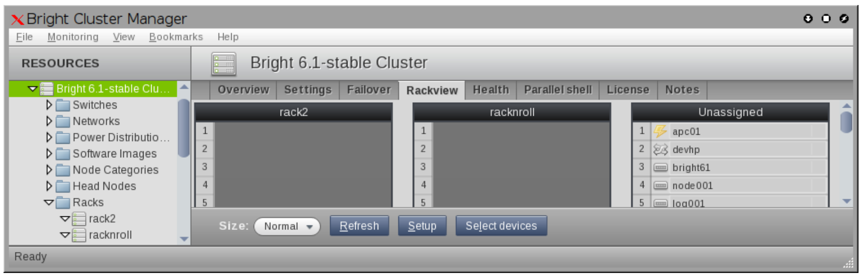


Figure 4.22: Rack View Using `cmgui`

Using Rack View

The `Rackview` pane of `cmgui` is a visualization of the layout of the racks, and also a visualization of the layout of devices such as nodes, switches, and chassis within a rack.

Any device can be visually located in the rack that it has been assigned to (section 4.13.3) by clicking on the "`Locate in rack`" button in the `Tasks` tabbed pane for that device. Clicking the button opens up the `Rackview` pane, with the device highlighted.

Some of the `Racks` configuration settings of figure 4.21 can also be visualized within rack view.

Visual metrics, in the form of bars that are color-calibrated with metric values, can be assigned via the `Setup` button from within the `Rackview` pane. The details for this are given on page 144.

In the `Rackview` pane, the color bar associated with a metric value is displayed within the device (a node or a switch), which is in turn within the rack. This can be useful in getting an idea of the spread of values across a cluster at a glance. In some cases patterns can become obvious when related to the physical positioning of devices in the rack, and related to the physical positioning of the racks.

For example, if ventilation is not working well in one part of the room, then metrics for the rack devices in that part may show a higher temperature compared to devices that are better ventilated. Mapped out as colors in the `Rackview` pane, the discrepancy would be noticed at a glance.

Rack View Configuration

In rack view:

- The rack name is displayed at the top of the rack.
- An item such as a node can be assigned a particular position within a rack space (section 4.13.3).
- A cloud region (chapter 7) is given a rack space, and takes the region name as its "rack" name. For example `us-east-1`.
- A cloud region can be visualized as a rack for cloud nodes. That is, within the `Rackview` pane, cloud nodes are assigned to their cloud region in the same way as nodes are assigned to racks.
- The remaining items that are not assigned to a rack are kept in a space labeled "'`Unassigned`".
- Tooltip text above a node shows useful information about that particular node, and any rack-view-associated cluster metrics configured using the "`Rack Setup`" window (figure 4.24, described on page 144).
- At the bottom of the `Rackview` tabbed pane, the `Size:` field button allows a drop-down selection of rack sizes to set four kinds of rack views. These options are `Large`, `Normal`, `Small`, and `Tiny`.
  - `Large` shows the rack names, the node names, the rows of the racks, visual metrics (colored bars) next to the nodes, and a color calibration legend for the visual metrics along the bottom.
  - `Normal` shows the same as `Large`, except that racks displayed are scaled down to half the `Large` size. (figure 4.23).

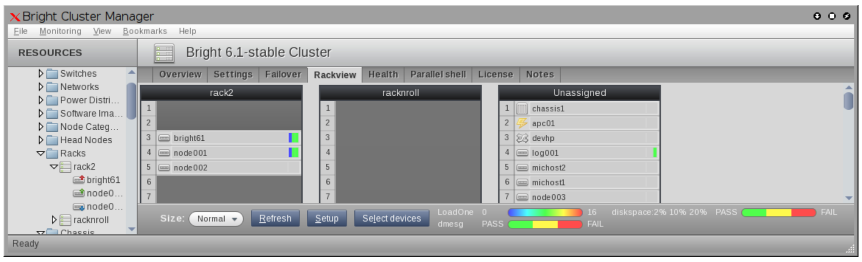
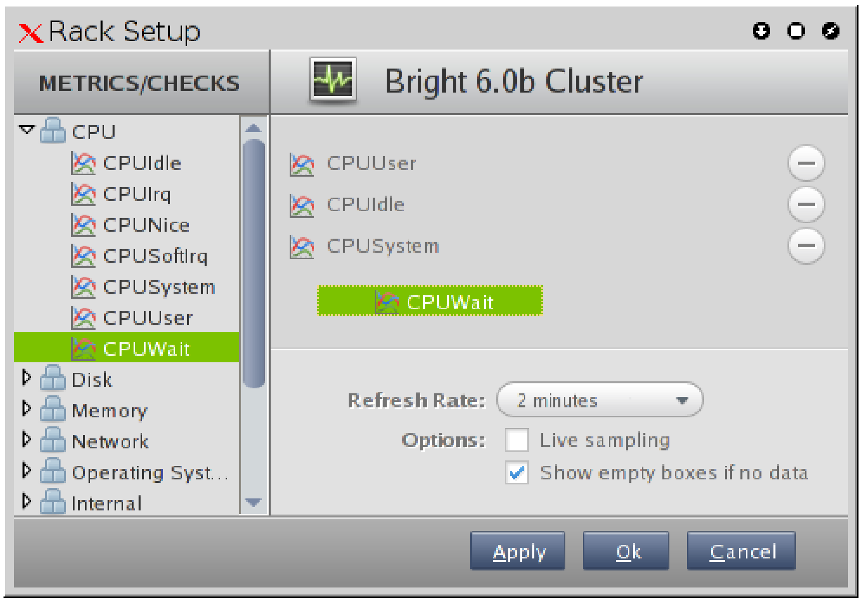


Figure 4.23: Rack View Normal Size

  - `Small` shows the same as `Normal`, except that the node names are dropped, and the racks are scaled down to half the `Normal` size.

- `Tiny` shows the same as `Small`, except that the racks are scaled down to half the `Small` size. This is really usually just to discern color patterns—the text associated with the racks is unreadable even with `cmgui` running in full-screen mode on a 22" monitor.

- The `Refresh` button refreshes the rack view.
- The `Setup` button opens up the `Rack Setup` window (figure 4.24).



Refresh Rate: 2 minutes
Options: ☐ Live sampling
✓ Show empty boxes if no data

Figure 4.24: Rack Setup, Showing Drag-And-Drop Of `CPUWait` Metric

The `Rack Setup` window allows the racks to be configured with metrics. Rack view then displays the metrics in the `Large`, `Normal`, `Small`, and `Tiny` rack views according to an automatically calibrated color scale.

In the `Rack Setup` window:

- The metrics can be configured by drag-and-drop.
- The `Apply` button applies a change.
- The `Ok` button applies any changes, and closes the "`Rack Setup`" screen.
- The `Refresh Rate` selection sets the time interval between metric measurement retrieval.
- The `Show empty boxes if no data` displays empty bars at the device if there are no data values, instead of the color that corresponds to a zero value.
- A check-mark in the `Live sampling` checkbox instructs that the selected metrics be measured and displayed at the refresh interval. Scripts that take longer than about a second to run can be used, but are best avoided because the display is not very "live" in that case.

- `Select devices` allows devices to be selected in groups according to a menu. Menu choices are
  - none
  - `hostname`
  - `MAC`
  - `tag`
  - `category`
  - `installmode`
  - `rack`
  - `position`
  - `running job`

  The selected menu items can then be filtered with regexes applied with Boolean `AND` or `OR`. The `Job filter` button allows selective filtering by clicking on jobs from a window.

  The resultant filtered items are devices such as nodes or switches, and these are then highlighted in the rack view display.

- The color scale is auto-calibrated by default. It can be user-calibrated via a dialog box, by clicking on the existing minimum (blue end) or maximum (red end) values.

4.13.3 Assigning Devices To A Rack

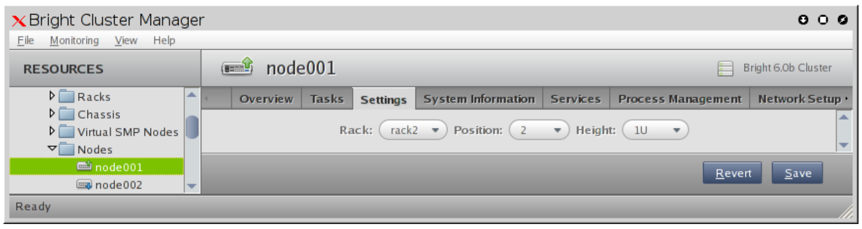
Devices such as nodes, switches, and chassis, can be assigned to racks.

By default, no such devices are assigned to a rack. All devices are thus originally shown as part of the "`Unassigned`" grouping in the `Rackview` tabbed pane.

Devices can be assigned to a particular rack and to a particular position within the rack as follows:

Assigning Devices To A Rack Using `cmgui`

Using `cmgui`, the assignments of a device such as a node to a rack can be done from the `Settings` tabbed pane of the device, within the `Rack` section (figure 4.25):



Rack: rack2 Position: 2 Height: 1U

Figure 4.25: Rack Assignment Using `cmgui`

Assigning Devices To A Rack Using `cmsh`

Using `cmsh`, the assignment can be done to a rack as follows:

```
[ddnmon61->device]% foreach -n node001..node003 (set deviceheight 1; se\
t deviceposition 2; set rack rack2)
[ddnmon61->device*]% commit
Successfully committed 3 Devices
[ddnmon61->device]%
```

The Convention Of The Top Of The Device Being Its Position

Since rack manufacturers usually number their racks from top to bottom, the position of a device in a rack (using the parameter `Position` in `cmgui`, and the parameter `deviceposition` in `cmsh`) is always taken to be where the top of the device is located. This is the convention followed even for the less usual case where the rack numbering is from bottom to top.

A position on a rack is 1U of space. Most devices have a height that fits in that 1U, so that the top of the device is located at the same position as the bottom of the device, and no confusion is possible. The administrator should however be aware that for any device that is greater than 1U in height such as, for example, a blade enclosure chassis (section 4.13.4), the convention means that it is the position that the top of the device is located at, that is noted as being the position of the device, rather than the position of the bottom.

4.13.4 Assigning Devices To A Chassis

A Chassis As A Physical Part Of A Cluster

In a cluster, several local nodes may be grouped together physically into a chassis. This is common for clusters using blade systems. Clusters made up of blade systems use less space, less hardware, and less electrical power than non-blade clusters with the same computing power. In blade systems, the blades are the nodes, and the chassis is the blade enclosure.

A blade enclosure chassis is typically 6 to 10U in size, and the node density is typically 2 blades per unit with current (2012) technology.

Chassis Configuration And Node Assignment

Basic chassis configuration and node assignment with `cmgui`: General chassis configuration in `cmgui` is done from the `Chassis` resource. Within the `Chassis` resource is the chassis `Members` tabbed pane, which allows assignment of nodes to a chassis, and which is described in detail soon (page 148).

If the `Chassis` resource is selected from the resource tree, then a new chassis item can be added, or an existing chassis item can be opened.

The chassis item can then be configured from the available tabs, some of which are:

- The `Tasks` tab (figure 4.26).

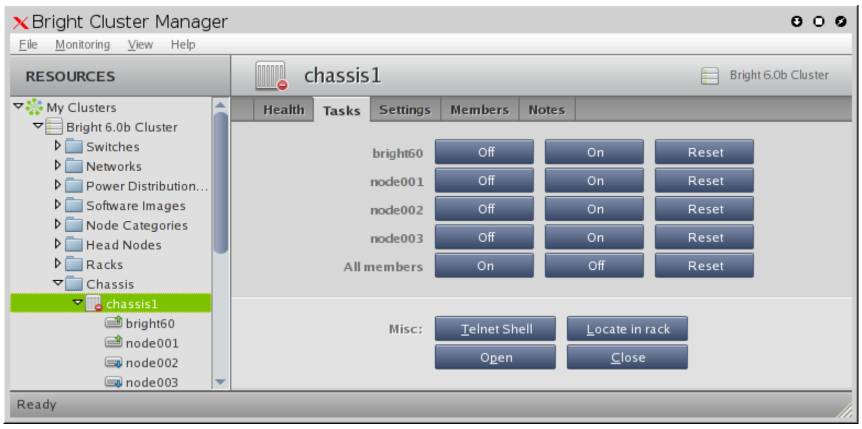


Figure 4.26: Chassis Tasks Tab

The `Tasks` tab for a chassis item allows:

– Individual member nodes of the chassis to be powered on and off, or reset.

– All the member nodes of the chassis to be powered on and off, or reset.

Miscellaneous tasks in the `Tasks` tab are:

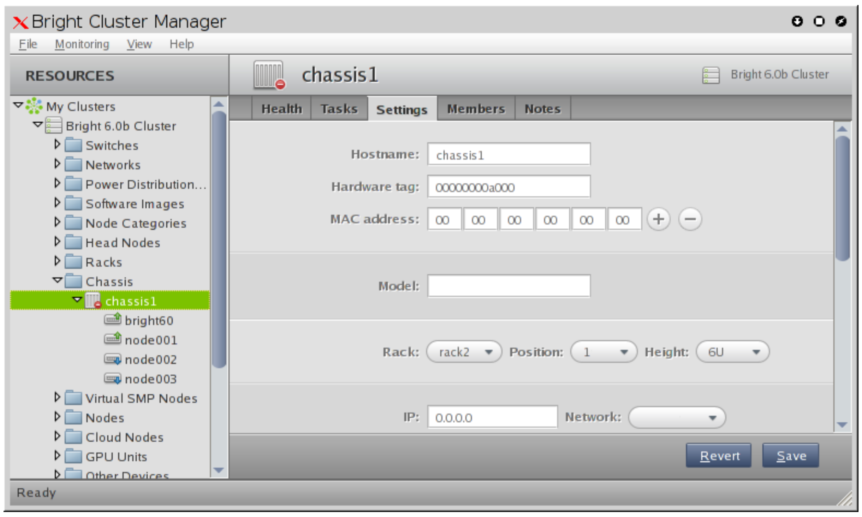
– The ability to start a telnet session, if possible, with the chassis operating system.

– The ability to start an SSH session, if possible, with the chassis operating system.

– The ability to locate the chassis in the visual representation of rack view.

– The ability to open or close the chassis state. States are described in section 6.5.

- The `Settings` tab (figure 4.27).



Hostname: chassis1
Hardware tag: 00000000a000
MAC address: 00 00 00 00 00 00
Model:
Rack: rack2 Position: 1 Height: 6U
IP: 0.0.0.0 Network:

Figure 4.27: Chassis Settings Tab

Amongst other options, the `Settings` tab for a chassis item allows selection of:

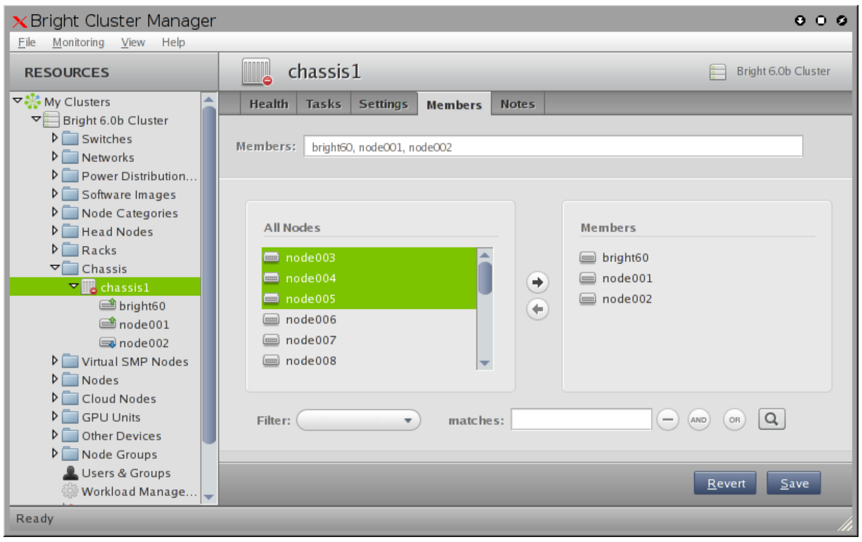
– A rack from the set of defined racks.

– A position within the rack (typically between 1 and 42 units).

– The height of the chassis (typically between 6 and 10 units). Because the typical numbering of a rack is from position 1 at the top to position 42 at the bottom, a chassis of height 6 at position 1 will take up the space from position 1 to 6. Effectively the base of the chassis is then at position 6, and the space from 1 to 6 can only be used by hardware that is to go inside the chassis.

The settings become active when the `Save` button is clicked.

• The `Members` tab (figure 4.28).



Members: bright60, node001, node002

Figure 4.28: Chassis Members Tab

The `Members` tab for a chassis item allows the following actions:

– Nodes can be selected, then added to or removed from the chassis.

– Multiple nodes can be selected at a time for addition or removal.

– The selection can be done with standard mouse and keyboard operations, or using the filter tool.

– The filter tool allows selection based on some useful node parameters such as the hostname, the MAC address, the category. Several rules can made to operate together.

Basic chassis configuration and node assignment with `cmsh`**:** The `chassis` mode in `cmsh` allows configuration related to a particular chassis. Tab-completion suggestions for a selected chassis with the `set` command show possible parameters that may be set:

Example

```
[ddnmon61->device[chassis1]]% set
custompingscript             ip          powercontrol
custompingscriptargument     mac         powerdistributionunits
custompowerscript            members     rack
custompowerscriptargument    model       slots
deviceheight                 network     tag
deviceposition               notes       userdefined1
ethernetswitch               partition   userdefined2
hostname                     password    username
```

Whether the suggested parameters are actually supported depends on the chassis hardware. For example, if the chassis has no network interface of its own, then the `ip` and `mac` address settings may be set, but cannot function.

The "positioning" parameters of the chassis within the rack can be set as follows with `cmsh`:

Example

```
[ddnmon61->device[chassis1]]% set rack rack2
[ddnmon61->device*[chassis1*]]% set deviceposition 1; set deviceheight 6
[ddnmon61->device*[chassis1*]]% commit
```

The members of the chassis can be set as follows with `cmsh`:

Example

```
[ddnmon61->device[chassis1]]% set members ddnmon61 node001..node005
[ddnmon61->device*[chassis1*]]% commit
```

Advanced Chassis Configuration And Node Assignment

This section can be ignored in a first reading. The example of section 4.13.5 is recommended before looking at these features.

Advanced chassis configuration features are *slots*, *container index*, and *layout*. These can be useful for some less common requirements.

Slots In `cmsh`, slots in the chassis can be assigned an arbitrary value. A chassis may physically have a series of slots, with a node per slot. The labels used can be in the node naming order, in which case setting labels of `1` to the first node, `2` to the second node, and so on, is probably a good start. If the numbering is not right, or if there is no numbering, then administrators can set their own arbitrary values:

Example

```
[ddnmon61->device[chassis1]]% set slots ddnmon61 "leftmost top"
[ddnmon61->device*[chassis1*]]% set slots node001 "leftmost bottom"
[ddnmon61->device*[chassis1*]]% set slots node002 "red cable tie top"
[ddnmon61->device*[chassis1*]]% set slots node003 "red cable tie bottom"
```

The values set are saved with the `commit` command. Looking up the slot values via `cmsh` is possible, but a more convienient way to use such labels is via the environment variable `$CMD_CHASSIS_SLOT` in scripts.

Container index The container index can be used to reposition nodes in the container (the chassis here) within the rack view. The nodes can then be made to align according to preference.

By default, nodes take up the first empty space in the container displayed by the rack view, from left to right, then on the next line, like the sequence followed when reading words in lines on a page. The position of each node in this sequence is given by the value of container index, which indicates the physical position in the sequence for a non-zero value. For a zero value the node simply takes up its default position (the first empty space in the container). So depending on the physical positioning, the nodes that are displaying incorrectly by default can be moved to a corrected location by setting their container index value. For example, the following layout is displayed for a chassis with a position specified as 5, a height specified as 10U, with 16 nodes as members, and with the default of `containerindex=0` for all members (figure 4.29):

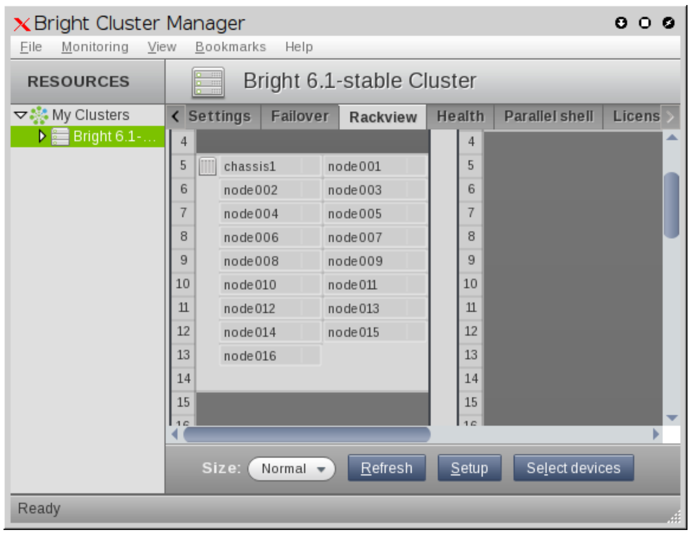


Figure 4.29: Default Positions For 16 Nodes In A 10U Chassis

The chassis icon object itself is represented as a node in the container. The icon takes up the first free position by default, which is position 1, and which would correspond to `containerindex=1` if the position were set explicitly. Similarly, node001 takes up position 2 by default, which would correspond to `containerindex=2` if set explicitly, node002 takes up position 3 (`containerindex=3`), and so on.

The last node can be shifted one position further as follows:

```
[ddnmon61->device]% set node016 containerindex 18; commit
Successfully committed 1 Devices
[ddnmon61->device]%
```

The display is then (figure 4.30):

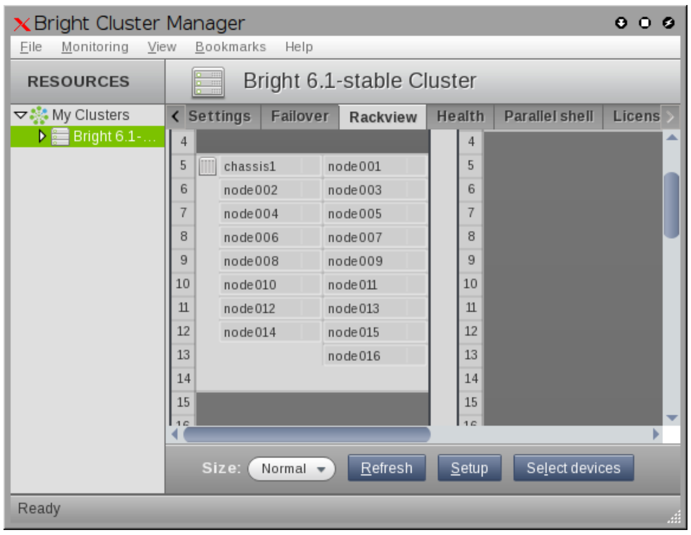


Figure 4.30: Display After Shifting Last Node By One Position

Moving node001 to the place vacated by node016 can be done with the following command:

```
[ddnmon61->device]% set node001 containerindex 17; commit
Successfully committed 1 Devices
[ddnmon61->device]%
```

The display is then (figure 4.31):

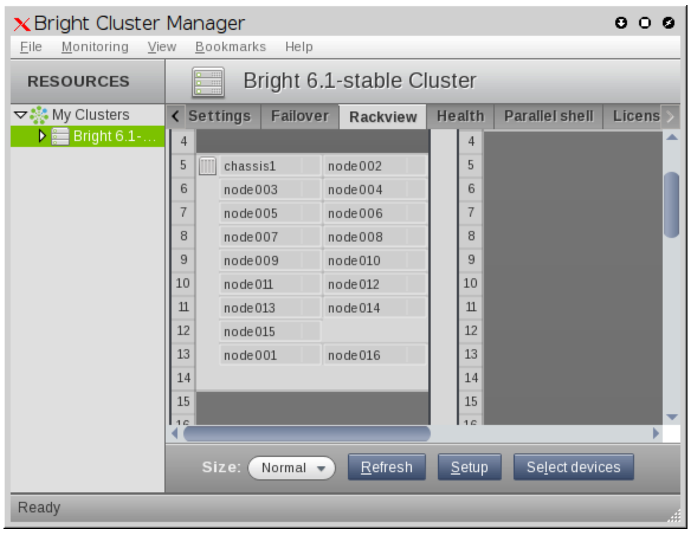


Figure 4.31: Display After Shifting First Node Near End

As can be seen, the remaining nodes all shift to take up the first empty space available in the container.

Layout The administrator can make the rackview display match the physical layout, by displaying nodes in a chassis in a custom vertical or horizontal layout, instead of in the default horizontal layout.

In this section, a way of getting to a particular custom vertical layout is first explained, via successive approximations. After that, an example of a custom horizontal layout is described.

- **Custom vertical layouts** can make sense for blades in a chassis. Thus, for the chassis in figure 4.29, with 16 nodes as members, a chassis with 8 blades across and 2 nodes vertically might be desired.

  – **First approximation.**

  A starting point to get to the desired layout can be specified by setting the `User defined 1` field of the container (the chassis) to:

  ```
  layout |8,2
  ```

  This can be read as:

  "layout, vertical text in the node box, space for 8 nodes along the *x*-axis, space for 2 nodes along the *y*-axis, for this container"

  This makes room in the chassis for 8 nodes across by 2 nodes up, although this space is only filled to the extent that members exist for the chassis.

  The "|", or vertical sign, makes the text associated with the member nodes orient vertically, or more accurately, rotates the text clockwise a quarter turn.

  Conversely, the "-", or horizontal sign, keeps the text oriented the standard way.

  * In `cmgui` the layout specification can be set via the `Chassis` resource, then selecting the `Settings` tab, and scrolling down to the `User defined 1` field.
  * In `cmsh`, this can be set with a session such as:

  ```
  [root@ddnmon61 ~]# cmsh
  [ddnmon61]% device use chassis1
  [ddnmon61->device[chassis1]]% set userdefined1 "layout |8,2"
  [ddnmon61->device*[chassis1*]]% commit
  ```

  Figure 4.32 shows the resulting first approximation of the desired vertical layout:

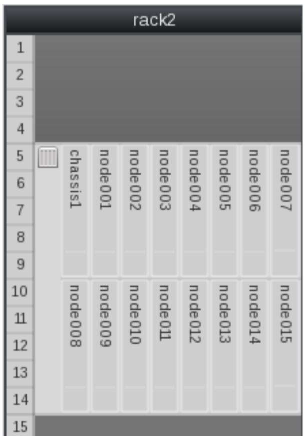


Figure 4.32: Vertical Blades Display, Approximation 1

  – **Second approximation.**

Since the chassis icon takes up 1 unit of space inside the chassis container in rackview, 1 unit of extra room should be specified either along the *x*- or *y*-axis. With this extra unit specified, for example as:

```
layout |9,2
```

an improved second approximation of the layout is reached (figure 4.33).

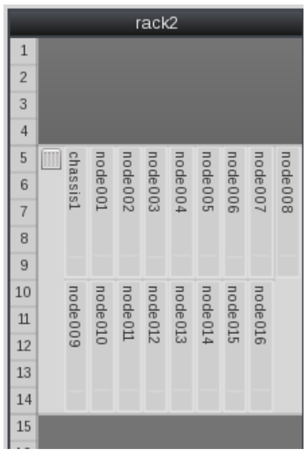


Figure 4.33: Vertical Blades Display, Approximation 2

– **Third approximation.**

Then, applying the principles of `containerindex` (page 150), a third adjustment of the layout can be carried out on the second row with the following short `bash` script:

```
[root@ddnmon61 ~]# for i in {009..016}
do j=$((10#$i+2))
cmsh -c "device use node$i; set containerindex $j; commit"
done
```

The `10#$i` bit in the `bash` script simply makes each zero-padded value that `$i` loops through get treated as a base 10 number instead of an octal number when assigned to `$j`. In other words, the script is like running (6 lines elided):

```
cmsh -c "device use node009; set containerindex 11; commit"
...
cmsh -c "device use node016; set containerindex 18; commit"
```

On commit, this third approximation shifts each node from `node009` onwards to a position further to the right. The layout of the third approximation is now a sufficiently human-friendly display for most system administrators, with the nodes aligned in pairs in the way they would typically be physically (figure 4.34):

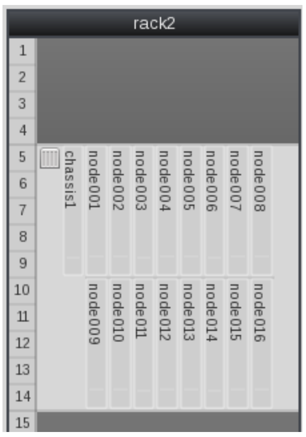


Figure 4.34: Vertical Blades Display, Approximation 3

- **Custom horizontal layouts** can be made, instead of a default horizontal layout of the kind shown in figure 4.29. A session can be continued from that layout, or from the layout reached at the end of figure 4.34, as follows:

  The container (which is the chassis) index values can first be cleared:

```
[bright61->device[chassis1]]% !
...
[root@bright61 cmsh]# for i in {001..016}
  do
    cmsh -c "device use node$i; clear containerindex; commit"
  done
[root@bright61 cmsh]# exit
```

  The `$userdefined1` value can then be set to: `layout -3,10`

```
[bright61->device[chassis1]]% set userdefined1 "layout -3,10"; commit
```

  This can be read as:

  "layout, horizontal text in the node box, space for 3 nodes along the x-axis, and 10 nodes along the y-axis, for this container".

  Finally, the value of `containerindex` can be offset by 3 for each node, in order to place the nodes in a position that starts in a fresh row, separate from the `chassis` text:

```
[bright61->device[chassis1]]% !
...
[root@bright61 cmsh]# for i in {001..016}
  do
    j=$((10#$i+3))
    cmsh -c "device use node$i; set containerindex $j; commit"
  done
```

  The resulting layout is shown in figure 4.35.

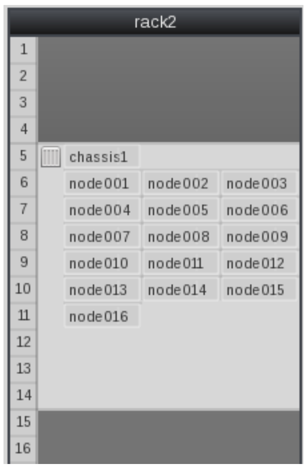


Figure 4.35: Horizontal Nodes Custom Display

4.13.5 An Example Of Assigning A Device To A Rack, And Of Assigning A Device To A Chassis

The following illustrative case explains how to set up a blade server in a cluster. This breaks down naturally into 2 steps: firstly configuring the chassis (the blade server device) for the rack, and secondly configuring the nodes (the blade devices) for the chassis.

1. **Assigning a chassis to a position within a rack:** A position within a rack of the cluster is first chosen for the chassis.

   To assign a chassis to a rack in the cluster manager, the rack must first exist according to the cluster manager. If the rack is not yet known to the cluster manager, it can be configured as described in section 4.13.

   Assuming racks called `rack1`, `rack2`, and `rack3` exist, it is decided to place the chassis in `rack2`.

   If the blade server is 10 units in height, with 20 nodes, then 10 units of room from the rack must be available. If positions 5 to 14 are free within `rack2`, then suitable positional settings are:

   - `rack2` for the rack value,
   - `5` for the position value,
   - `10` for the height value.

   These values can then be set as in the example of figure 4.27, and come into effect after saving them.

2. **Assigning nodes to the chassis:** The nodes of the chassis can then be made members of the chassis, as is done in the example of figure 4.28, and saving the members' settings.

   The rack, position within the rack, and height values of the chassis are inherited by nodes that are members of the chassis.

   After the nodes rack, position within the rack, and height values match those held by the chassis, they display correctly within the chassis space within rackview, as shown in figure 4.36.

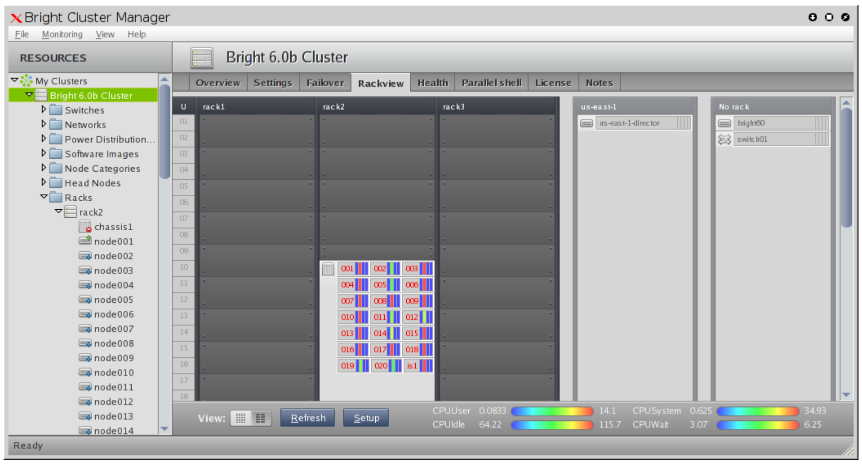


Figure 4.36: Nodes Displayed Inside The Chassis In Rack View

4.14 Configuring A GPU Unit, And Configuring GPU Settings

4.14.1 GPUs And GPU Units

GPUs (Graphics Processing Units) are processors that are heavily optimized for executing certain types of parallel processing tasks. GPUs were originally used for rendering graphics, and one GPU typically has hundreds of cores. When used for general processing, they are sometimes called General Processing GPUs, or GPGPUs. For convenience, the "GP" prefix is not used in this manual.

A GPU is typically placed on a PCIe card. GPUs can be physically inside the node that uses them, or they can be physically external to the node that uses them. As far as the operating system on the node making use of the physically external GPUs is concerned, the GPUs are internal to the node.

If the GPUs are physically external to the node, then they are typically in a *GPU unit*. A GPU unit is a chassis that hosts only GPUs. It can typically provide GPU access to several nodes, usually via PCIe extender connections.

4.14.2 GPU Unit Configuration Example: The Dell PowerEdge C410x

An example of a GPU unit is the Dell PowerEdge C410x, which comes in a 3U chassis size, has up to 16 Tesla M-series GPUs (in the form of cards in slots) inside it, and can allocate up to 4 GPUs per node.

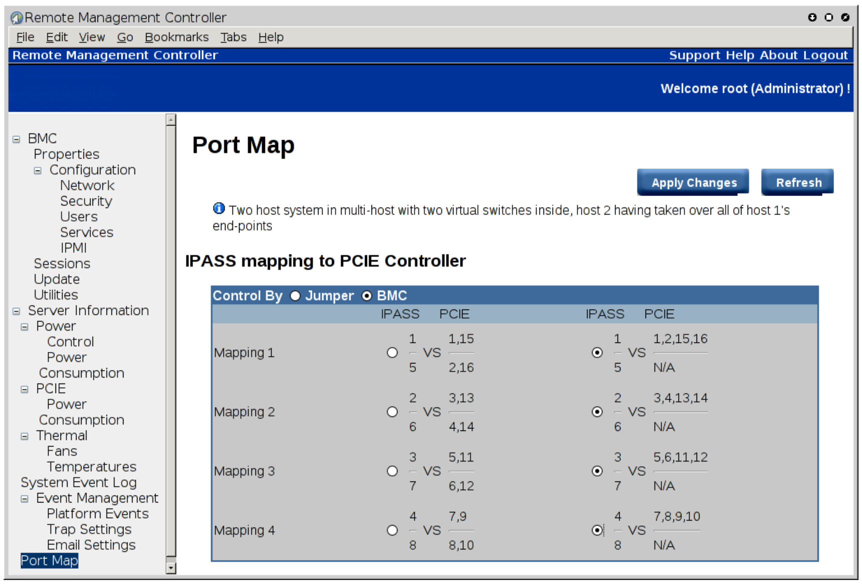
It can be configured to work with DirectMon as follows:

1. The GPUs in the unit are assigned to nodes using the direct web interface provided by the Baseboard Management Controller (BMC) of the C410x. This configuration is done outside of DirectMon. The assignment of the GPUs can be done only according to the fixed groupings permitted by the web interface.

   For example, if the unit has 16 GPUs in total (1, 2,..., 16), and there are 4 nodes in the cluster (node001, node002,..., node004), then an appropriate GPU indices mapping to the node may be:

- node001 is assigned 1, 2, 15, 16
- node002 is assigned 3, 4, 13, 14
- node003 is assigned 5, 6, 11, 12
- node004 is assigned 7, 8, 9, 10

This mapping is decided by accessing the C410x BMC with a browser, and making choices within the “`Port Map`” resource (figure 4.37). 4 mappings are displayed (Mapping 1, 2, 3, 4), with columns displaying the choices possible for each mapping. In the available mapping choices, the iPass value indicates a port, which corresponds to a node. Thus iPass 1 is linked to node001, and its corresponding PCIE values (1, 2, 15, 16) are the GPU indices here. Each iPass port can have 0 (N/A), 2, or 4 PCIE values associated with it, so that the GPU unit can supply up to 4 GPUs to each node.



Port Map

Two host system in multi-host with two virtual switches inside, host 2 having taken over all of host 1's end-points

IPASS mapping to PCIE Controller

Control By ○ Jumper [✓ ● BMC]

		IPASS		PCIE		IPASS		PCIE
Mapping 1	○	1 / 5	VS	1,15 / 2,16	[✓ ●]	1 / 5	VS	1,2,15,16 / N/A
Mapping 2	○	2 / 6	VS	3,13 / 4,14	[✓ ●]	2 / 6	VS	3,4,13,14 / N/A
Mapping 3	○	3 / 7	VS	5,11 / 6,12	[✓ ●]	3 / 7	VS	5,6,11,12 / N/A
Mapping 4	○	4 / 8	VS	7,9 / 8,10	[✓ ●]	4 / 8	VS	7,8,9,10 / N/A

Figure 4.37: Assigning GPUs To Nodes With The C410x web interface

The GPU indices (PCIE values) (that is, the numbers 1, 2,..., 16) are used to identify the card in DirectMon. The lowest GPU index associated with a particular node—for example, 5 (rather than 6) for node03—is used to make DirectMon aware in the next step that an association exists for that node, as well as aware which the first GPU card associated with it is.

2. The GPUs can be assigned to the nodes in DirectMon as follows:

   First, the GPU unit holding the GPUs can be assigned to the GPU unit resource. This can be carried out by adding the hostname of the GPU unit, for example in `cmsh` with:

   ```
   [root@ddnmon61 ~]# cmsh
   [ddnmon61]% device
   [ddnmon61->device]% add gpuunit schwartz
   ```

   This drops the administrator into the `gpuunit` object, which has the hostname `schwartz` here. The equivalent addition in `cmgui` can be done via the `GPU Units` resource folder, and clicking on the `Add` button in the `Overview` tabbed pane.

Next, the BMC user name, password, IP address, and network values can be set to appropriate values:

Example

```
[ddnmon61->device*[schwartz*]]% set bmcusername darkhelmet
[ddnmon61->device*[schwartz*]]% set bmcpassword 12345
[ddnmon61->device*[schwartz*]]% set ip 10.148.0.10
[ddnmon61->device*[schwartz*]]% set network bmcnet
```

Here, it is assumed that a BMC network that the BMC network interface is connected to has already been created appropriately. If it is not already set up, then the network can be configured as described in section 4.8.

The value of `owners` must be set to the list of nodes that have access to GPUs.

Also, the reserved variable, `userdefined1` must be set to a key=value pair, so that each key (the node in owner) has a value (the PCIE values), assigned to it:

Example

```
[ddnmon61...]% set owners node001 node002 node003 node004
[ddnmon61...]% set userdefined1 node001=1,2,15,16 node002=3,4,13,\
14 node003=5,6,11,12 node004=7,8,9,10
[ddnmon61->device*[schwartz*]]% commit
```

As a convenience, if the indices are all separated by 1, as in the case of node004 here, only the lowest index need be set for that node. The line setting the `userdefined1` value can thus equivalently be carried out with:

Example

```
[ddnmon61...]% set userdefined1 node001=1,2,15,16 node002=3,4,13,\
14 node003=5,6,11,12 node004=7
```

Once these values are committed, DirectMon is able to track GPU assignment consistently and automatically.

3. Finally, CUDA drivers should be installed on the relevant nodes so that they can make use of the GPUs. The details on how to do this are given in section 13.5.

4.14.3 Configuring GPU Settings

The `gpusettings` Submode In `cmsh`

In `cmsh`, GPUs can be configured for a specified node, via `device` mode.

Going into the `gpusettings` submode for that node then allows a range to be specified for a range of GPUs:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device use node001
[ddnmon61->device[node001]]% gpusettings
[ddnmon61->device[node001]->gpusettings]% add 1-3 ; commit
```

GPUs can also be configured for a specified category, via `category` mode. For example, using the category `default`, then entering into the `gpusettings` submode allows a range to be set for the range of GPUs:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% category use default
[ddnmon61->category[default]]% gpusettings
[ddnmon61->category[default]->gpusettings]% list
GPU range (key)  Power limit  ECC mode     Compute mode  Clock speeds
---------------- ------------ ------------ ------------- -------------
[ddnmon61->category[default]->gpusettings]% add 1-3 ; commit
[ddnmon61->category[default]->gpusettings[1-3]]% show
Parameter                      Value
------------------------------ ----------------------------------
Clock speeds
Compute mode
ECC mode
...
```

As usual, GPU settings for a node override those for a category (section 3.1.3).

After a range has been set, the following GPU settings may be specified, if supported, from within the `gpusettings` submode:

- `clockspeeds`: The clock speed (frequency in MHz) choices are:
  - `max#max`: maximum for both memory and graphics
  - `min#min`: minimum for both memory and graphics
  - *<number for memory>*#*<number for graphics>*: other possible combinations for memory and graphics, as suggested by tab-completion
- `computemode`: Contexts can be computed with the following values for this mode:
  - `Default`: Multiple contexts are allowed
  - `Exclusive thread`: Only one context is allowed per device, usable from one thread at a time
  - `Exclusive process`: Only one context is allowed per device, usable from multiple threads at a time. This mode option is valid for CUDA 4.0 and higher. Earlier CUDA versions ran only in this mode.
  - `Prohibited`: No contexts are allowed per device
- `eccmode`: Sets the ECC bit error check, with:

- `enabled`
- `disabled`

When ECC is enabled:

- Single bit errors are detected, using the `EccSBitGPU` metric (page 637), and corrected automatically.
- Double bit errors are also detected, using the `EccDBitGPU` metric (page 637), but cannot be corrected.

The `gpureset` and `gpuclearecc` commands (page 161) are relevant for this setting.

- `gpureset`: After changing the `eccmode` setting, the GPU must be reset to make the new value active.
- `gpuclearecc`: This is used to clear ECC counts for the GPU.

• `gpurange`: range values can be set as follows:

  - `all` or `*`: The GPU settings apply to all GPUs on the node.
  - <*number*>: The GPU settings apply to an individual GPU, for example: `1`
  - <*number range*>: The GPU settings apply to a range of GPUs, for example: `1,3-5`

• `operationmode`: The operation mode values for the GPU are:

  - `All on`: All of the GPU node is made available for calculations
  - `Compute`: Only the computing section is made available for calculations. Graphics section calculations are not possible.
  - `Low Double precision`: A mode that spares resources and power if little double precision use is expected

• `persistencemode`:

  - `Enabled`: If the mode is enabled, then the NVIDIA driver is kept loaded, even if nothing is actively using it. This reduces loading latency when running CUDA programs.

• `powerlimit`: The administrator-defined upper power limit for the GPU. Only valid if `powermode` is `Supported`.

  - `min`: The minimum upper power limit that the hardware supports.
  - `max`: The maximum upper power limit that the hardware supports.
  - <*number*>: An arbitrary upper power limit, specified as a number between `min` and `max`
  - `default`: Upper power limit based on the default hardware value.

• `powermode`:

- `Supported`: Allows power management functions.

If no value is specified for a GPU setting, then the hardware default is used.

Within `cmsh`, the `gpureset` and `gpuclearecc` commands can be executed at a node or category level, and also executed for specified nodes. The `cmsh` help text for these commands (`help gpureset` and `help gpuclearecc`) gives details, including node range specifications.

- The `gpureset` command carries out a power reset. It is executed only if there are no processes using the GPU. Possible processes using the GPU can be CUDA applications, X, monitoring applications, or other calculations. The device is all done with the reset in about 2 seconds.

  Example

  ```
  [ddnmon61->device]% gpureset -c default 1,3-6
  ```

  This resets GPUs 1, 3, 4, 5, and 6 in the default category.

  The command can be used to reset GPUs without rebooting the node, or to reset some hung GPUs.

- The `gpuclearecc` command clears the ECC counts for the GPU. The ECC counts are of two types:

  - `volatile`: Resets when the driver unloads. Resets on power cycle.
  - `aggregate`: Persists when the driver unloads. Resets on power cycle.

  By default, both types are cleared when the command is run, but each type can be cleared separately.

  Example

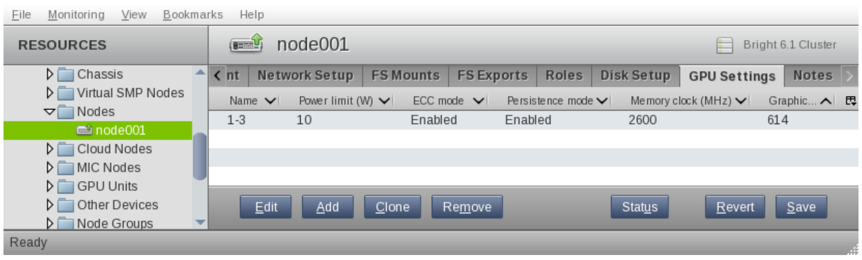
  ```
  [ddnmon61->device]% gpuclearecc -c default 1,3-6 -a
  ```

  This clears the aggregate ECC counts for the GPUs 1, 3, 4, 5, and 6 in the default category.

The `gpusettings` Submode In `cmgui`

In `cmgui` the GPU settings can be accessed by selecting a node from the `Nodes` resource, and then selecting the `GPU Settings` tabbed pane for that node (figure 4.38). Similarly, GPU settings can also be accessed within the `Category` resource, selecting a category item, and then selecting the `GPU Settings` tabbed pane.

Clicking on the `Status` button in the `GPU Settings` pane opens up a new window to show what properties and values are supported for the GPU.



Name	Power limit (W)	ECC mode	Persistence mode	Memory clock (MHz)	Graphic...
1-3	10	Enabled	Enabled	2600	614

Figure 4.38: GPU Settings Tabbed Pane For A Node

4.15 Configuring Custom Scripts

Some scripts are used for custom purposes. These are used as replacements for certain default scripts, for example, in the case of non-standard hardware where the default script does not do what is expected. The custom scripts that can be set, along with their associated arguments are:

- `custompowerscript` and `custompowerscriptargument`
- `custompingscript` and `custompingscriptargument`
- `customremoteconsolescript` and `customremoteconsolescriptargument`

The environment variables of CMDaemon (Appendix I.5) can be used in the scripts. Successful scripts, as is the norm, return 0 on exit.

4.15.1 `custompowerscript`

The use of custom power scripts is described in section 5.1.4.

4.15.2 `custompingscript`

The following example script:

Example

```
#!/bin/bash
/bin/ping -c1 $CMD_IP
```

can be defined and set for the cases where the default built-in ping script, cannot be used.

By default, the node device states are detected by the built-in ping script (section 6.5) using a type of PING called TCP SYN pings. This results in the statuses that can be seen on running the list command of `cmsh` in device mode. An example output, formatted for convenience, is:

Example

```
[root@ddnmon61]# cmsh -c "device; format hostname:15, status:15; list"
hostname (key)   status
---------------- ---------------
bright61         [    UP    ]
node001          [    UP    ]
node002          [    UP    ]
```

If some device is added to the cluster that blocks such pings, then the built-in ping can be replaced by the custom ping of the example, which relies on standard ICMP ping.

However, the replacement custom ping script need not actually use a variety of ping at all. It could be a script running web commands to query a chassis controller, asking if all its devices are up. The script simply has to provide an exit status compatible with expected ping behavior. Thus an exit status of 0 means all the devices are indeed up.

4.15.3 `customremoteconsolescript`

A custom remote console script can be used to replace the built-in remote console utility. This might be used, for example, to allow the administrator remote console access through a proprietary KVM switch client to its KVM switch.

4.16 Cluster Configuration Without Execution By CMDaemon

4.16.1 Cluster Configuration: The Bigger Picture

The configurations carried out in this chapter so far are based almost entirely on configuring nodes, via a CMDaemon front end (`cmsh` or `cmgui`), using CMDaemon to execute the change. Indeed, much of this manual is about this too because it is the preferred technique. It is preferred:

- because it is intended by design to be the easiest way to do common cluster tasks,
- and also generally keeps administration overhead minimal in the long run since it is CMDaemon rather than the system administrator that then takes care of tracking the cluster state.

There are however other cluster configuration techniques besides execution by CMDaemon. To get some perspective on these, it should be noted that cluster configuration techniques are always fundamentally about modifying a cluster so that it functions in a different way. The techniques can then for convenience be separated out into modification techniques that rely on CMDaemon execution and techniques that do not, as follows:

1. **Configuring nodes with execution by CMDaemon:** As explained, this is the preferred technique. The remaining techniques listed here should therefore usually only be considered if the task cannot be done with `cmgui` or `cmsh`.

2. **Replacing the node image:** The image on a node can be replaced by an entirely different one, so that the node can function in another way. This is covered in section 4.16.2. It can be claimed that since it is CMDaemon that selects the image, this technique should perhaps be classed as under item 1. However, since the execution of the change is really carried out by the changed image without CMDaemon running on the image, and because changing the entire image to implement a change of functionality is rather extreme, this technique can be given a special mention outside of CMDaemon execution.

3. **Using a `FrozenFile` directive:** Applied to a configuration file, this directive prevents CMDaemon from executing changes on that file for nodes. During updates, the frozen configuration may therefore

need to be changed manually. The prevention of CMDaemon acting on that file prevents the standard cluster functionality that would run based on a fully CMDaemon-controlled cluster configuration. The `FrozenFile` directive is introduced in 3.6.4, and covered in the configuration context in section 4.16.3.

4. **Using an `initialize` or `finalize` script:** This type of script is run during the initrd stage, much before CMDaemon on the regular node starts up. It is run if the functionality provided by the script is needed before CMDaemon starts up, or if the functionality that is needed cannot be made available later on when CMDaemon is started on the regular nodes. CMDaemon does not execute the functionality of the script itself, but the script is accessed and set on the initrd via a CMDaemon front end (Appendix E.2), and executed during the initrd stage. It is often convenient to carry out minor changes to configuration files inside a specific image in this way, as shown by the example in Appendix E.5. The `initialize` and `finalize` scripts are introduced in section 4.16.4

5. **A shared directory:** Nodes can be configured to access and execute a particular software stored on a shared directory of the cluster. CMDaemon does not execute the functionality of the software itself, but is able to mount and share directories, as covered in section 4.11.

Finally, outside the stricter scope of cluster configuration adjustment, but nonetheless a valid way to modify how a cluster functions, and therefore mentioned here for more completeness, is:

6. **Software management:** the installation, maintenance, and removal of software packages (sections 10.2–10.6).

4.16.2 Making Nodes Function Differently By Image

Making All Nodes Function Differently By Image

To change the name of the image used for an entire cluster, for example after cloning the image and modifying it (section 4.16.2), the following methods can be used:

- in `cmgui` from within the `Cluster` resource, in the `Settings` tab
- or in `cmsh` from within the `base` object of `partition` mode

A system administrator more commonly sets the software image on a per-category or per-node basis (section 4.16.2).

Making Some Nodes Function Differently By Image

For minor changes, adjustments can often be made to node settings via initialize and finalize scripts so that nodes or node categories function differently (section 4.16.4).

For major changes, it is usually more appropriate to have nodes function differently from each other by simply setting different images per node category with CMDaemon.

This can be done as follows with `cmsh`:

1. The image on which the new one will be based is cloned. The cloning operation not only copies all the settings of the original (apart from the name), but also the data of the image:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% softwareimage
[ddnmon61->softwareimage]% clone default-image imagetwo
[ddnmon61->softwareimage*[imagetwo*]]% commit
Thu Aug 11 15:44:44 2011 [notice] ddnmon61: Started to copy: /cm/\
images/default-image -> /cm/images/imagetwo
[ddnmon61->softwareimage*[imagetwo*]]%
Thu Aug 11 15:53:22 2011 [notice] ddnmon61: Copied: /cm/images/de\
fault-image -> /cm/images/imagetwo
[ddnmon61->softwareimage[imagetwo]]%
```

2. After cloning, the settings can be modified in the new object. For example, if the kernel needs to be changed to suit nodes with different hardware, kernel modules settings are changed (section 6.3.2) and committed. This creates a new image with a new ramdisk.

   Other ways of modifying and committing the image for the nodes are also possible, as discussed in sections 10.2–10.6 of this chapter.

3. The modified image that is to be used by the differently functioning nodes is placed in a new category in order to have the nodes be able to choose the image. To create a new category easily, it can simply be cloned. The image that the category uses is then set:

```
[ddnmon61->softwareimage[imagetwo]]% category
[ddnmon61->category]% clone default categorytwo
[ddnmon61->category*[categorytwo*]]% set softwareimage imagetwo
[ddnmon61->category*[categorytwo*]]% commit
[ddnmon61->category[categorytwo]]%
```

4. • For just one node, or a few nodes, the node can be set from `device` mode to the new category (which has the new image):

```
[ddnmon61->category[categorytwo]]% device
[ddnmon61->device]% use node099
[ddnmon61->device[node099]]% set category categorytwo
[ddnmon61->device*[node099*]]% commit; exit
```

   • If there are many nodes, for example node100 sequentially up to node200, they can be set to that category using a `foreach` loop like this:

   Example

```
[ddnmon61->device]% foreach -n node100..node200 (set category\
 categorytwo)
[ddnmon61->device*]% commit
```

5. Rebooting restarts the nodes that are assigned to the new category with the new image.

Similarly, using `cmgui`, a default image can be cloned with the "`Clone`" button in the "`Software Images`" resource operating on an existing image. The new image is modified from the "`Software Images`" tab (section 10.3.4) or using the other image modifying methods in this chapter, to make nodes function in the way required, and any appropriate node or node category is assigned this image.

4.16.3 Making All Nodes Function Differently From Normal Cluster Behavior With `FrozenFile`

Configuration changes carried out by `cmgui` or `cmsh` often generate, restore, or modify configuration files (Appendix A).

However, sometimes an administrator may need to make a direct change (without using `cmgui` or `cmsh`) to a configuration file to set up a special configuration that cannot otherwise be done.

The `FrozenFile` directive to CMDaemon (Appendix C) applied to such a configuration file stops CMDaemon from altering the file. The frozen configuration file is generally applicable to all nodes and is therefore a possible way of making all nodes function differently from their standard behavior.

Freezing files is however best avoided, if possible, in favor of a CMDaemon-based method of configuring nodes, for the sake of administrative maintainability.

4.16.4 Adding Functionality To Nodes Via An `initialize` Or `finalize` Script

CMDaemon can normally be used to allocate different images per node or node category as explained in section 4.16.2. However, some configuration files do not survive a reboot (Appendix A), sometimes hardware issues can prevent a consistent end configuration, and sometimes drivers need to be initialized before provisioning of an image can happen. In such cases, an `initialize` or `finalize` script (Appendix E.5) can be used to initialize or configure nodes or node categories.

These scripts are also useful because they can be used to implement minor changes across nodes:

Example

Supposing that some nodes with a particular network interface have a problem auto-negotiating their network speed, and default to 100Mbps instead of the maximum speed of 1000Mbps. Such nodes can be set to ignore auto-negotiation and be forced to use the 1000Mbps speed by using the `ETHTOOL_OPTS` configuration parameter in their network interface configuration file: `/etc/sysconfig/network-scripts/ifcfg-eth0` (or `/etc/sysconfig/network/ifcfg-eth0` in SUSE).

The `ETHTOOL_OPTS` parameter takes the options to the "`ethtool -s` *<device>*" command as options. The value of *<device>* (for example `eth0`) is specified by the filename that is used by the configuration file itself (for example `/etc/sysconfig/network-scripts/ifcfg-eth0`). The `ethtool` package is installed by default with DirectMon. Running the command:

```
ethtool -s autoneg off speed 1000 duplex full
```

turns out after some testing to be enough to reliably get the network card up and running at 1000Mbps on the problem hardware.

However, since the network configuration file is overwritten by node-installer settings during reboot, a way to bring persistence to the file setting is needed. One way to ensure persistence is to append the configuration setting to the file with a `finalize` script, so that it gets tagged onto the end of the configuration setting that the node-installer places for the file, just before the network interfaces are taken down again in preparation for `init`.

The script may thus look something like this for a Red Hat system:

```
#!/bin/bash

## node010..node014 get forced to 1000 duplex
if [[ $CMD_HOSTNAME = node01[0-4] ]]
then
echo 'ETHTOOL_OPTS="speed 1000 duplex full"'>>/localdisk/etc/sysconfig/\
network-scripts/ifcfg-eth0
fi
```

4.16.5 Examples Of Configuring Nodes With Or Without CMDaemon

A node or node category can often have its software configured in CMDaemon via `cmgui` or `cmsh`:

Example

Configuring a software for nodes using `cmgui` or `cmsh`: If the software under consideration is CUPS, then a node or node category can manage it from the `Services` tab with `cmgui` or `cmsh` as outlined in section 4.12.2.

A counterexample to this is:

Example

Configuring a software for nodes without using `cmgui` or `cmsh`[3], using an image: Software images can be created with and without CUPS configured. Setting up nodes to load one of these two images via a node category is an alternative way of letting nodes run CUPS.

Whether node configuration for a particular functionality is done with CMDaemon, or directly with the software, depends on what an administrator prefers. In the preceding two examples, the first example with `cmgui` or `cmsh` setting the CUPS service is likely to be preferred over the second example where an entire separate image must be maintained. A new category must also be created in the second case.

Generally, sometimes configuring the node via DirectMon, and not having to manage images is better, sometimes configuring the software and making various images to be managed out of it is better, and sometimes only one of these techniques is possible anyway.

Configuring Nodes Using `cmgui` Or `cmsh`: Category Settings

When configuring nodes using `cmgui` or `cmsh`, configuring particular nodes from a node category to overrule the state of the rest of its category (as explained in section 3.1.3) is sensible for a small number of nodes. For larger numbers it may not be organizationally practical to do this, and another category can instead be created to handle nodes with the changes conveniently.

[3]except to link nodes to their appropriate image via the associated category

The CUPS service in the next two examples is carried out by implementing the changes via `cmgui` or `cmsh` acting on CMDaemon.

Example

Setting a few nodes in a category: If only a few nodes in a category are to run CUPS, then it can be done by those few nodes having the CUPS service enabled in the `Nodes` resource, thereby overriding (section 3.1.3) the category settings.

Example

Setting many nodes to a category: If there are many nodes that are to be set to run CUPS, then a separate, new category can be created (cloning it from the existing one is easiest) and those many nodes are moved into that category, while the image is kept unchanged. The CUPS service setting is then set at category level to the appropriate value for the new category.

In contrast to these two examples, the software image method used in section 4.16.2 to implement a functionality such as CUPS would load up CUPS as configured in an image, and would not handle it via CMDaemon[3]. So, in section 4.16.2, software images prepared by the administrator are set for a node category. Since, by design, images are only selected for a category, a node cannot override the image used by the category other than by creating a new category, and using it with the new image. The administrative overhead of this can be inconvenient.

Administrators would therefore normally prefer letting CMDaemon track software functionality across nodes as in the last two examples, rather than having to deal with tracking software images manually. Indeed, the `roles` assignment option (section 3.1.5) is just a special preconfigured functionality toggle that allows CMDaemon to set categories or regular nodes to provide certain functions, typically by enabling services.

5 Power Management

Aspects of power management in DirectMon include:

- managing the main power supply to nodes through the use of power distribution units, baseboard management controllers, or CMDaemon
- monitoring power consumption over time
- setting CPU scaling governors for power-saving
- setting power-saving options in workload managers
- ensuring the passive head node can safely take over from the active head during failover (Chapter 15)
- allowing cluster burn tests to be carried out (appendix N)

The ability to control power inside a cluster is therefore important for cluster administration, and also creates opportunities for power savings. This chapter describes the DirectMon power management features.

In section 5.1 the configuration of the methods used for power operations is described.

Section 5.2 then describes the way the power operations commands themselves are used to allow the administrator turn power on or off, reset the power, and retrieve the power status. It explains how these operations can be applied to devices in various ways.

Section 5.3 briefly covers monitoring power.

Section 5.4 describes how CMDaemon can set CPU defaults to control some of the CPU-dependent power consumption.

The integration of power saving with workload management systems is covered in the chapter on Workload Management (section 9.9).

5.1 Configuring Power Parameters

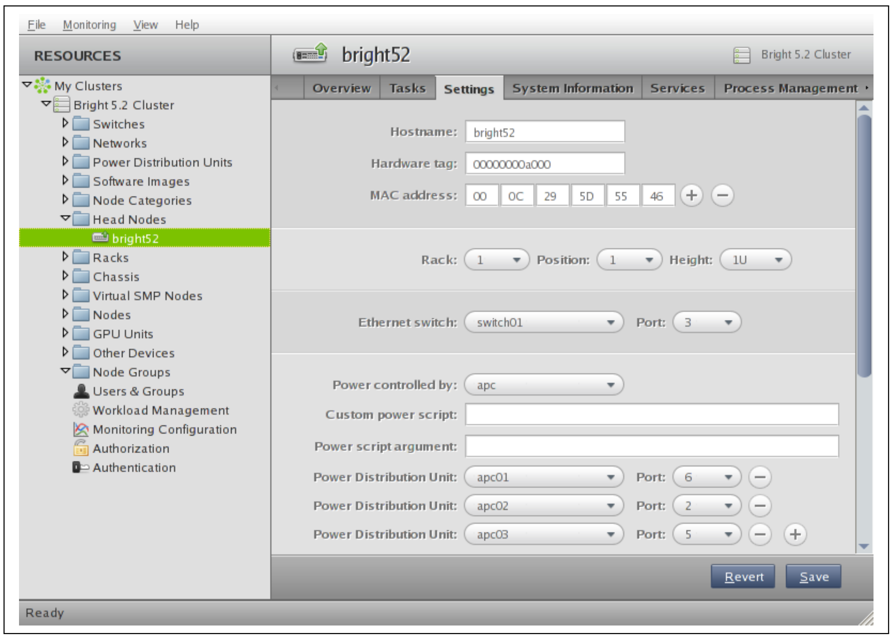
Several methods exist to control power to devices:

- Power Distribution Unit (PDU) based power control
- IPMI-based power control (for node devices only)
- Custom power control
- HP iLO-based power control (for node devices only)

5.1.1 PDU-Based Power Control

For PDU-based power control, the power supply of a device is plugged into a port on a PDU. The device can be a node, but also anything else with a power supply, such as a switch. The device can then be turned on or off by changing the state of the PDU port.

To use PDU-based power control, the PDU itself must be a device in the cluster and be reachable over the network. The `Settings` tab of each device object plugged into the PDU is then used to configure the PDU ports that control the device. Figure 5.1 shows the `Settings` tab for a head node.



Hostname: bright52
Hardware tag: 00000000a000
MAC address: 00 0C 29 5D 55 46
Rack: 1 Position: 1 Height: 1U
Ethernet switch: switch01 Port: 3
Power controlled by: apc
Custom power script:
Power script argument:
Power Distribution Unit: apc01 Port: 6
Power Distribution Unit: apc02 Port: 2
Power Distribution Unit: apc03 Port: 5

Figure 5.1: Head Node Settings

Each device plugged into the PDU can have PDU ports added and removed with the ⊕ and ⊖ buttons in their `Settings` tab. For the APC brand of PDUs, the "`Power controlled by`" property in the `Settings` tab should be set to `apc`, or the list of PDU ports is ignored by default. Overriding the default is described in section 5.1.3.

Since nodes may have multiple power feeds, there may be multiple PDU ports defined for a single device. The cluster management infrastructure takes care of operating all ports of a device in the correct order when a power operation is done on the device.

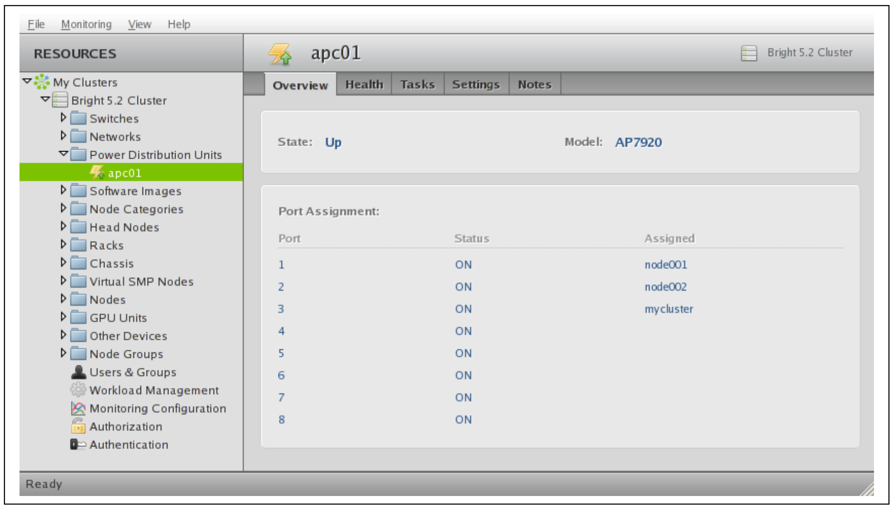
It is also possible for multiple devices to share the same PDU port. This is the case for example when *twin nodes* are used (i.e. two nodes sharing a single power supply). In this case, all power operations on one device apply to all nodes sharing the same PDU port.

If the PDUs defined for a node are not manageable, then the node's baseboard management controllers (that is, IPMI/iLO and similar) are assumed to be inoperative and are therefore assigned an unknown state. This means that dumb PDUs, which cannot be managed remotely, are best not assigned to nodes in DirectMon. Administrators wishing to use DirectMon to record that a dumb PDU is assigned to a node can deal with it as follows:

- in `cmgui` the `Notes` tab, or the "`User defined 1`"/"`User defined 2`" options in the `Settings` tab for that node can be used.
- in `cmsh` the equivalent is accessible when using the node from device mode, and running "`set notes`", "`set userdefined1`", or "`set userdefined2`".

For PDUs that are manageable:

- In `cmgui`, the `Overview` tab of a PDU (figure 5.2) provides an overview of the state of PDU ports and devices that have been associated with each port.



State: Up Model: AP7920

Port Assignment:

Port	Status	Assigned
1	ON	node001
2	ON	node002
3	ON	mycluster
4	ON	
5	ON	
6	ON	
7	ON	
8	ON	

Figure 5.2: PDU Overview

The power status of a node can be seen by selecting the node from the `Nodes` resource, then selecting the `Overview` tabbed pane. The first tile of the `Overview` tab displays the power state and assignment of any PDUs for the node.

- In `cmsh`, power-related options can be accessed from `device` mode, after selecting a device:

Example

```
[ddnmon61]% device use node001
[ddnmon61->device[node001]]% show | grep -i power
Custom power script argument
Ipmi/iLO power reset delay              0
Power control                           apc
PowerDistributionUnits                  apc01:6 apc01:7
```

The power status of a node can be accessed with:

Example

```
[ddnmon61->device[node001]]% power status
```

If the node is up and has one or more PDUs assigned to it, then the power status is one of ON, OFF, RESET, FAILED, or UNKNOWN:

Power Status	Description
ON	Power is on
OFF	Power is off
RESET	Shows during the short time the power is off during a power reset. The reset is a hard power off for PDUs, but can be a soft or hard reset for other power control devices.
FAILED	Power status script communication failure.
UNKNOWN	Power status script timeout

5.1.2 IPMI-Based Power Control

IPMI-based power control relies on the baseboard management controller (BMC) inside a node. It is therefore only available for node devices. Blades inside a blade chassis typically use IPMI for power management. For details on setting up networking and authentication for IPMI interfaces, see section 4.8.

To carry out IPMI-based power control operations, the "`Power controlled by`" property in figure 5.1 must be set to the IPMI interface through which power operations should be relayed. Normally this IPMI interface is configured to be `ipmi0`. Any list of configured APC PDU ports displayed in the GUI is ignored by default when the "`Power controlled by`" property is not `apc`.

Example

Configuring power parameters settings for a node using `cmsh`:

```
[mycluster]% device use node001
[...device[node001]]% set powerdistributionunits apc01:6 apc01:7 apc01:8
[...device*[node001*]]% get powerdistributionunits
apc01:6 apc01:7 apc01:8
[...device*[node001*]]% removefrom powerdistributionunits apc01:7
[...device*[node001*]]% get powerdistributionunits
apc01:6 apc01:8
[...device*[node001*]]% set powercontrol apc
[...device*[node001*]]% get powercontrol
apc
[...device*[node001*]]% commit
```

5.1.3 Combining PDU- and IPMI-Based Power Control

By default when nodes are configured for IPMI Based Power Control, any configured PDU ports are ignored. However, it is sometimes useful to change this behavior.

For example, in the CMDaemon configuration file directives in `/cm/local/apps/cmd/etc/cmd.conf` (introduced in section 3.6.2 and listed in Appendix C), the default value of `PowerOffPDUOutlet` is `false`. It can be set to `true` on the head node, and CMDaemon restarted to activate it.

With `PowerOffPDUOutlet` set to `true` it means that CMDaemon, after receiving an IPMI-based power off instruction for a node, and after powering off that node, also subsequently powers off the PDU port. Powering off the PDU port shuts down the BMC, which saves some additional power—typically a few watts per node. When multiple nodes share the same PDU port, the PDU port only powers off when all nodes served by that particular PDU port are powered off.

When a node has to be started up again the power is restored to the node. It is important that the node BIOS is configured to automatically power on the node when power is restored.

5.1.4 Custom Power Control

For a device which cannot be controlled through any of the standard existing power control options, it is possible to set a custom power management script. This is then invoked by the cluster management daemon on the head node whenever a power operation for the device is done.

Power operations are described further in section 5.2.

Using `custompowerscript`

To set a custom power management script for a device, the `powercontrol` attribute is set to `custom` using either `cmgui` or `cmsh`, and the value of `custompowerscript` is specified. The value for `custompowerscript` is the full path to an executable custom power management script on the head node(s) of a cluster.
A custom power script is invoked with the following mandatory arguments:

```
myscript <operation> <device>
```

where `<device>` is the name of the device on which the power operation is done, and `<operation>` is one of the following:

```
ON
OFF
RESET
STATUS
```

On success a custom power script exits with exit code 0. On failure, the script exits with a non-zero exit-code.

Using `custompowerscriptargument`

The mandatory argument values for `<operation>` and `<device>` are passed to a custom script for processing. For example, in `bash` the positional variables `$1` and `$2` are typically used for a custom power script. A custom power script can also be passed a further argument value by setting the value of `custompowerscriptargument` for the node via `cmsh` or `cmgui`. This further argument value would then be passed to the positional variable `$3` in `bash`.

An example custom power script is located at `/cm/local/examples/cmd/custompower`. In it, setting `$3` to a positive integer delays the script via a `sleep` command by `$3` seconds.

An example that is conceivably more useful than a "`sleep $3`" command is to have a "`wakeonlan $3`" command instead. If the `custompowerscriptargument` value is set to the MAC address of the node, that means the MAC value is passed on to `$3`. Using this technique, the power operation ON can then carry out a Wake On LAN operation on the node from the head node.

Setting the `custompowerscriptargument` can be done like this for all nodes:

```
#!/bin/bash
for nodename in $(cmsh -c "device; foreach * (get hostname)")
do
  macad=`cmsh -c "device use $nodename; get mac"`
  cmsh -c "device use $nodename; set customscriptargument $macad; commit"
done
```

The preceding material usefully illustrates how `custompowerscriptargument` can be used to pass on arbitrary parameters for execution to a custom script.

However, the goal of the task can be achieved in a simpler and quicker way using the environment variables available in the cluster management daemon environment (Appendix I.5). This is explained next.

Using Environment Variables With `custompowerscript`

Simplification of the steps needed for custom scripts in CMDaemon is often possible because there are values in the CMDaemon environment already available to the script. A line such as:

```
env > /tmp/env
```

added to the start of a custom script dumps the names and values of the environment variables to `/tmp/env` for viewing.

One of the names is `$CMD_MAC`, and it holds the MAC address string of the node being considered.

So, it is not necessary to retrieve a MAC value for `custompowerscriptargument` with a bash script as shown in the previous section, and then pass the argument via `$3` such as done in the command "`wakeonlan $3`". Instead, `custompowerscript` can simply call "`wakeonlan $CMD_MAC`" directly in the script when run as a power operation command from within CMDaemon.

5.1.5 Hewlett Packard iLO-Based Power Control

iLO Configuration During Installation

If "`Hewlett Packard`" is chosen as the node manufacturer during installation (section 2.3.5), and the nodes have an iLO management interface, then Hewlett-Packard's iLO management package, `hponcfg`, is installed by default on the nodes and head nodes.

iLO Configuration After Installation

If "`Hewlett Packard`" has not been specified as the node manufacturer during installation then it can be configured after installation as follows:

The `hponcfg` rpm package is normally obtained and upgraded for specific HP hardware from the HP website. Using an example of `hponcfg-3.1.1-0.noarch.rpm` as the package downloaded from the

HP website, and to be installed, the installation can then be done on the head node, the software image, and in the node-installer as follows:

```
rpm -iv hponcfg-3.1.1-0.noarch.rpm
rpm --root /cm/images/default-image -iv hponcfg-3.1.1-0.noarch.rpm
rpm --root /cm/node-installer -iv hponcfg-3.1.1-0.noarch.rpm
```

To use iLO over all nodes, the following steps are done:

1. The iLO interfaces of all nodes are set up like the IPMI interfaces as outlined in section 5.1.2, using "`set powercontrol ilo0`" instead of "`set powercontrol ipmi0`". DirectMon treats HP iLO interfaces just like regular IPMI interfaces, except that the interface names are `ilo0`, `ilo1`... instead of `ipmi0`, `ipmi1`...

2. The `ilo_power.pl` custom power script must be configured on all nodes. This can be done with a `cmsh` script. For example, for all nodes in the `default` category:

   Example

   ```
   [mycluster]% device foreach -c default (set custompowerscript /cm/lo\
   cal/apps/cmd/scripts/powerscripts/ilo_power.pl)
   [mycluster]% device foreach -c default (set powercontrol custom)
   [mycluster]% device commit
   ```

5.2 Power Operations

Power operations may be done on devices from either `cmgui` or `cmsh`. There are four main power operations:

- Power On: power on a device
- Power Off: power off a device
- Power Reset: power off a device and power it on again after a brief delay
- Power Status: check power status of a device

5.2.1 Power Operations With `cmgui`

In `cmgui`, buttons for executing On/Off/Reset operations are located under the `Tasks` tab of a device. Figure 5.3 shows the `Tasks` tab for a head node.

The `Overview` tab of a device can be used to check its power status information. In the display in figure 5.4, for a head node, the green LEDs indicate that all three PDU ports are turned on. Red LEDs would indicate power ports that have been turned off, while gray LEDs would indicate an unknown power status for the device.

Performing power operations on multiple devices at once is possible through the `Tasks` tabs of node categories and node groups.

It is also possible to do power operations on selected node through the `Nodes` folder in the resource tree: The nodes can be selected using the `Overview` tab. The selected group can then be operated on by a task chosen from the `Tasks` tab.

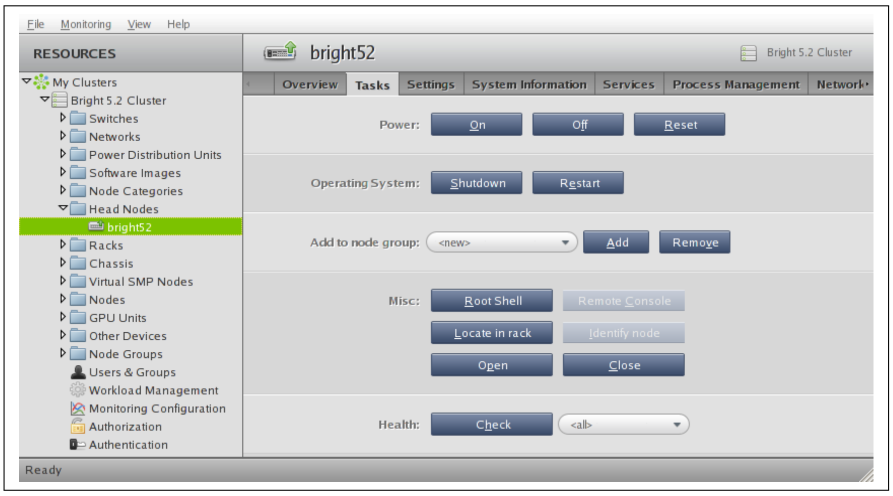
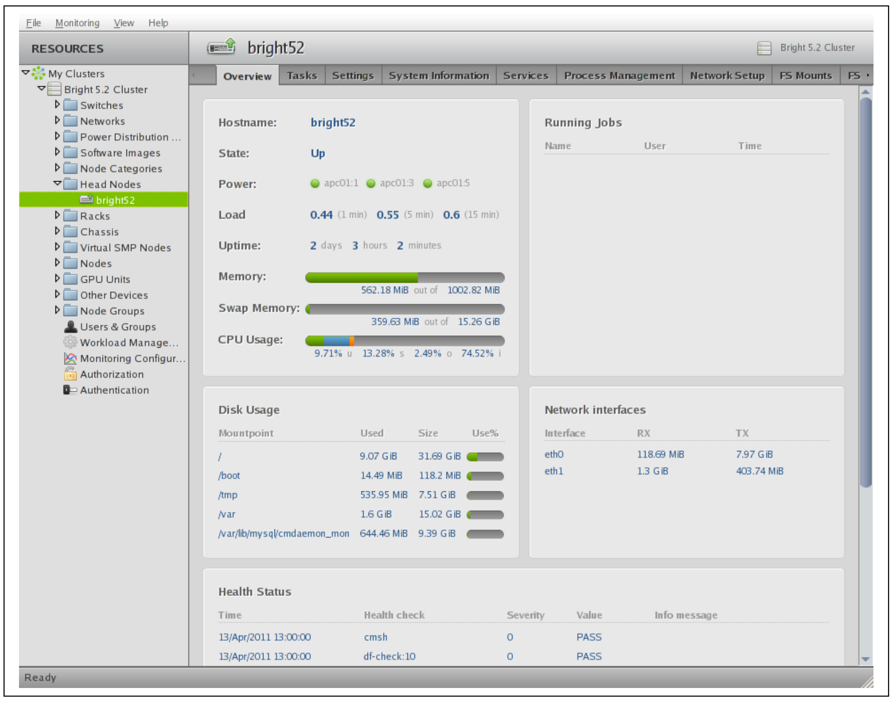


Figure 5.3: Head Node Tasks



Hostname: bright52

State: Up

Power: ● apc01:1 ● apc01:3 ● apc01:5

Load 0.44 (1 min) 0.55 (5 min) 0.6 (15 min)

Uptime: 2 days 3 hours 2 minutes

Memory: 562.18 MiB out of 1002.82 MiB

Swap Memory: 359.63 MiB out of 15.26 GiB

CPU Usage: 9.71% u 13.28% s 2.49% o 74.52% i

Running Jobs

Name	User	Time

Disk Usage

Mountpoint	Used	Size	Use%
/	9.07 GiB	31.69 GiB	
/boot	14.49 MiB	118.2 MiB	
/tmp	535.95 MiB	7.51 GiB	
/var	1.6 GiB	15.02 GiB	
/var/lib/mysql/cmdaemon_mon	644.46 MiB	9.39 GiB	

Network interfaces

Interface	RX	TX
eth0	118.69 MiB	7.97 GiB
eth1	1.3 GiB	403.74 MiB

Health Status

Time	Health check	Severity	Value	Info message
13/Apr/2011 13:00:00	cmsh	0	PASS	
13/Apr/2011 13:00:00	df-check:10	0	PASS	

Figure 5.4: Head Node Overview

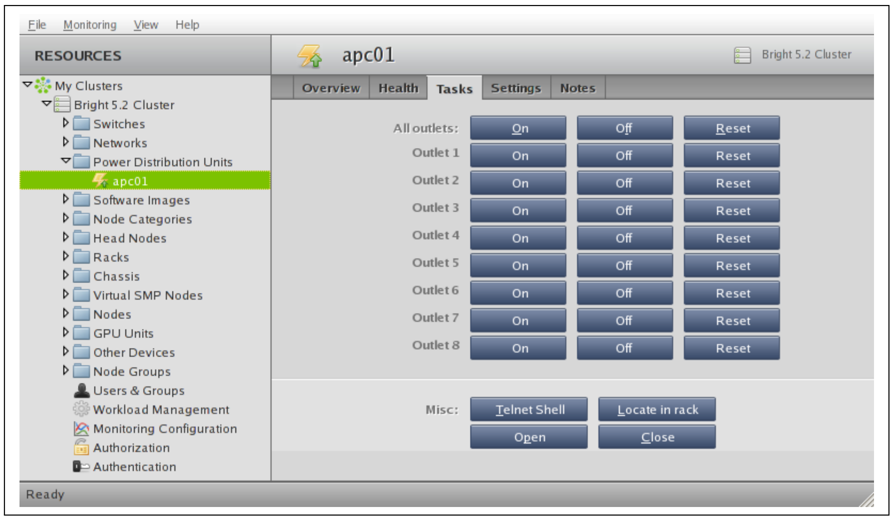


Figure 5.5: PDU Tasks

When doing a power operation on multiple devices, CMDaemon ensures a 1 second delay occurs by default between successive devices, to avoid power surges on the infrastructure. The delay period may be altered using cmsh's "-d|--delay" flag.

The Overview tab of a PDU object (figure 5.5), allows power operations on PDU ports by the administrator directly. All ports on a particular PDU can have their power state changed, or a specific PDU port can have its state changed.

5.2.2 Power Operations Through cmsh

All power operations in cmsh are done using the power command in device mode. Some examples of usage are now given:

- Powering on node001, and nodes from node018 to node033 (output truncated):

  Example

```
[mycluster]% device power -n node001,node018..node033 on
apc01:1 ............. [    ON    ] node001
apc02:8 ............. [    ON    ] node018
apc02:9 ............. [    ON    ] node019
...
```

- Powering off all nodes in the default category with a 100ms delay between nodes (some output elided):

  Example

```
[mycluster]% device power -c default -d 0.1 off
apc01:1 ............. [    OFF   ] node001
apc01:2 ............. [    OFF   ] node002
...
apc23:8 ............. [    OFF   ] node953
```

- Retrieving power status information for a group of nodes:

Example

```
[mycluster]% device power -g mygroup status
apc01:3 ............. [   ON    ] node003
apc01:4 ............. [   OFF   ] node004
```

Figure 5.6 shows usage information for the `power` command.

5.3 Monitoring Power

Monitoring power consumption is important since electrical power is an important component of the total cost of ownership for a cluster. The monitoring system of DirectMon collects power-related data from PDUs in the following metrics:

- `PDUBankLoad`: Phase load (in amperes) for one (specified) bank in a PDU
- `PDULoad`: Total phase load (in amperes) for one PDU

Chapter 11 on cluster monitoring has more on metrics and how they can be visualized.

5.4 CPU Scaling Governors

A cluster with CPU cores that run at a higher frequency consumes more power. A cluster administrator may therefore wish to implement schemes to control the CPU core frequencies, so that cluster power consumption is controlled.

5.4.1 The Linux Kernel And CPU Scaling Governors

In technology, a governor is the term used for a speed regulator. In computing, *CPU Scaling Governors* are power schemes that regulate what is commonly known as the CPU speed, or more precisely known as the CPU core clock frequency. The Linux kernel uses the *CPUFreq Governors* interface to implement these power schemes. The following governor values are currently commonly available to the interface:

Governor	Policy Description For CPU Frequency
`performance`	set to the highest allowed
`userspace`	set to that determined by root
`ondemand`	set according to demand, aggressively
`conservative`	set according to demand, non-aggressively
`powersave`	set to the lowest allowed

5.4.2 The Governor List According To `sysinfo`

Not all governors may be available for a particular CPU. Also, new governors may appear as new hardware is released. However, the hardware-defined list of governors that the kernel does detect as being available to a node <*hostname*> can always be seen in DirectMon:

```
Name:
   power - Manipulate or retrieve power state of devices

Usage:

   power [OPTIONS] status
   power [OPTIONS] on
   power [OPTIONS] off
   power [OPTIONS] reset

Options:
   -n, --nodes node(list)
     List of nodes, e.g. node001..node015,node20..node028,node030 or
     ^/some/file/containing/hostnames

   -g, --group group(list)
     Include all nodes that belong to the node group, e.g. testnodes
     or test01,test03

   -c, --category category(list)
     Include all nodes that belong to the category, e.g. default or
     default,gpu

   -r, --rack rack(list)
     Include all nodes that are located in the given rack, e.g rack01
     or rack01..rack04

   -h, --chassis chassis(list)
     Include all nodes that are located in the given chassis, e.g
     chassis01 or chassis03..chassis05

   -p, --powerdistributionunitport <pdu:port>(list)
     perform power operation directly on power distribution units. Use
     port '*' for all ports

   -b, --background
     Run in background, output will come as events

   -d, --delay <seconds>
     Wait <seconds> between executing two sequential power commands.
     This option is ignored for the status command

   -s, --status <states>
     Only run power command on nodes in specified states, e.g. UP,
     "CLOSED|DOWN", "INST.*"

   -f, --force
     Force power command on devices which have been closed

Examples:
   power status          Display power status for all devices or current device
   power on -n node001   Power on node001
```

Figure 5.6: `power` Command Help Text In `device` Mode

- in `cmgui`, by viewing it under <*hostname*>'s `System Information` tab
- in `cmsh`, by using the `sysinfo <hostname>` command, in `device` mode

The list can be displayed by CMDaemon in a form such as:

Example

```
[ddnmon61->device[ddnmon61]]% sysinfo
System Information
------------------ ----------------------------------------------------
...
Number of Cores    12
 Core 0            Six-Core AMD ... performance (ondemand, userspace)
 Core 1            Six-Core AMD ... performance (ondemand, userspace)
...
```

The governor in active operation for the cores is displayed outside the parentheses—in the example it is `performance`. The other two available, but inactive ones, are enclosed by parentheses—in the example they are `ondemand` and `userspace`. This list is essentially a hardware-defined list because its members depend only on the chip used. Which one of the members is active is however decided by a software setting for the governor value.

5.4.3 Setting The Governor

If no governor is set by the operating system or the application, then the value is decided by the BIOS setting for Cool 'n' Quiet (AMD) or SpeedStep (Intel).

On a running system, the cluster administrator can set the CPU governor value, in lower case, by using the CMDaemon front ends as follows:

- In `cmgui`:
  - if the `Node Categories` resource is chosen, and a particular category is selected, then a `Settings` tabbed pane for that category allows setting the `Scaling governor` to:
    - a single value.
    - multiple values. These can be set as a comma-separated list. This should not be confused with the hardware-defined list (section 5.4.2).
  - if the `Nodes` or `Head Nodes` resources is chosen, and a node selected from there, then its `Settings` tabbed pane allows its `Scaling governor` value to be set only to a single value.
- In `cmsh`:
  - if `category` mode, is used, and the category is chosen, then the `scalinggovernor` can be set to:
    - a single value.
    - multiple values. These can be set as a comma-separated list

  - if `device` mode is chosen, and a head or regular node selected from there, then its `scalinggovernor` value can be set only to a single value

If a list is set for scaling governors in DirectMon, at category level, then the first value in the list is applicable to all the nodes in that category. If that fails for any of the nodes, for example if a node does not support this value, then the next value is attempted, and so on, until the node is assigned a governor value. This allows configuration according to preferred order to take place at category level.

As usual, configuration of a value at node level overrides the category value. However, in contrast with the setting that is possible at the category level, only a single value is possible at node level. This is because, if a specific node comes to the attention of the administrator for a custom governor value, then that node can be assigned the value decided by the administrator.

Example

```
[ddnmon61->]% device use node001
[ddnmon61->device[node001]]% sysinfo
System Information
------------------ --------------------------------------------------
...
Number of Cores    12
 Core 0            Six-Core AMD ... performance (ondemand, userspace)
 Core 1            Six-Core AMD ... performance (ondemand, userspace)
...
 Core 11           Six-Core AMD ... performance (ondemand, userspace)

[ddnmon61->device[node001]]% category use default
[ddnmon61->category[default]]% set scalinggovernor ondemand\
,userspace,performance
[ddnmon61->category[default]]% device use node001
[ddnmon61->device[node001]]% sysinfo
System Information
------------------ --------------------------------------------------
...
Number of Cores    12
 Core 0            Six-Core AMD ... ondemand (userspace, performance)
 Core 1            Six-Core AMD ... ondemand (userspace, performance)
...
 Core 11           Six-Core AMD ... ondemand (userspace, performance)
```

6 Node Provisioning

This chapter covers *node provisioning*. Node provisioning is the process of how nodes obtain an image. Typically, this happens during their stages of progress from power-up to becoming active in a cluster, but node provisioning can also take place when updating a running node.

Section 6.1 describes the stages leading up to the loading of the kernel onto the node.

Section 6.2 covers configuration and behavior of the provisioning nodes that supply the software images.

Section 6.3 describes the configuration and loading of the kernel, the ramdisk, and kernel modules.

Section 6.4 elaborates on how the node-installer identifies and places the software image on the node in a 13-step process.

Section 6.5 explains node states during normal boot, as well node states that indicate boot problems.

Section 6.6 describes how running nodes can be updated, and modifications that can be done to the update process.

Section 6.7 explains how to add new nodes to a cluster so that node provisioning will work for these new nodes too. The `cmsh` and `cmsh` front ends for creating new node objects and properties in CMDaemon are described.

Section 6.8 describes troubleshooting the node provisioning process.

6.1 Before The Kernel Loads

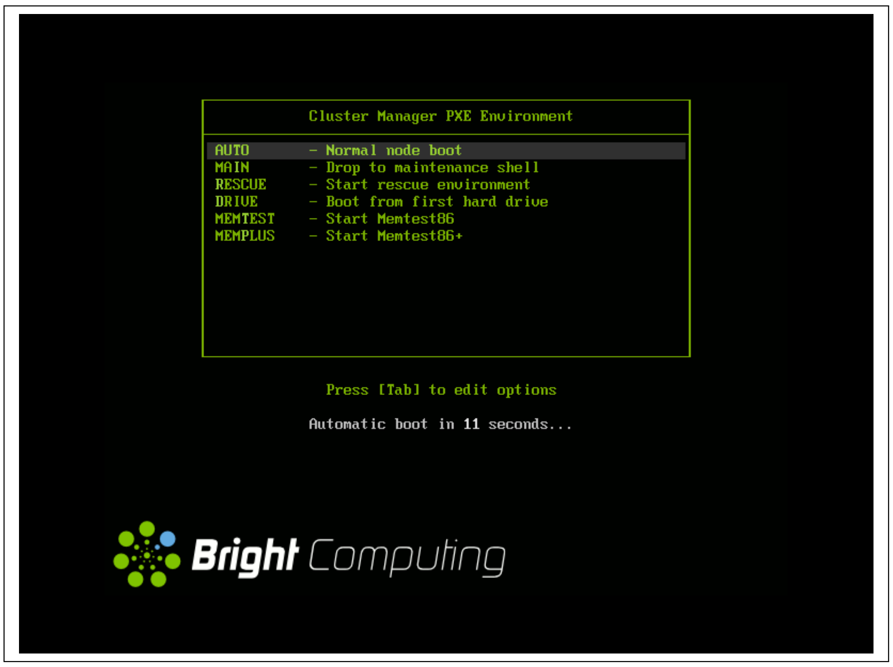
Immediately after powering up a node, and before it is able to load up the Linux kernel, a node starts its boot process in several possible ways:

6.1.1 PXE Booting

By default, nodes boot from the network when using DirectMon. This is called a *network boot*, or sometimes a *PXE boot*. It is recommended as a BIOS setting for nodes. The head node runs a tftpd server from within xinetd, which supplies the boot loader from within the default software image (section 3.1.2) offered to nodes.

The boot loader runs on the node and displays a menu (figure 6.1) based on loading a menu module within a configuration file. The configuration file is within the default software image *`<default-image>`* offered to nodes, and located at

/cm/images/<*default-image*>/boot/pxelinux.cfg/default
on the head node.



```
Cluster Manager PXE Environment

AUTO       - Normal node boot
MAIN       - Drop to maintenance shell
RESCUE     - Start rescue environment
DRIVE      - Boot from first hard drive
MEMTEST    - Start Memtest86
MEMPLUS    - Start Memtest86+

Press [Tab] to edit options

Automatic boot in 11 seconds...
```

Figure 6.1: PXE boot menu options

The default configuration file gives instructions to the menu module of PXElinux. The instruction set used is documented at http://www.syslinux.org/wiki/index.php/Comboot/menu.c32, and includes the TIMEOUT, LABEL, MENU LABEL, DEFAULT, and MENU DEFAULT instructions.

The PXE TIMEOUT Instruction

During the display of the PXE boot menu, a selection can be made within a timeout period to boot the node in a several ways. Among the options are some of the install mode options (section 6.4.4). If no selection is made by the user within the timeout period, then the AUTO install mode option is chosen by default.

In the PXE menu configuration file pxelinux.cfg/default, the default timeout of 5 seconds can be adjusted by changing the value of the "TIMEOUT 50" line. This value is specified in deciseconds.

Example

```
TIMEOUT 300   # changed timeout from 50 (=5 seconds)
```

The PXE LABEL And MENU LABEL Instructions

LABEL: The file pxelinux.cfg/default contains several multiline LABEL statements.

Each LABEL statement is associated with a kernel image that can be loaded from the PXE boot menu along with appropriate kernel options.

Each LABEL statement also has a text immediately following the LABEL tag. Typically the text is a description, such as linux, main, RESCUE, and so on. If the PXE menu module is not used, then tab completion prompting displays the list of possible text values at the PXE boot prompt so that the associated kernel image and options can be chosen by user intervention.

MENU LABEL: By default, the PXE menu module is used, and by default, each LABEL statement also contains a MENU LABEL instruction. Each MENU LABEL instruction also has a text immediately following the MENU LABEL tag. Typically the text is a description, such as AUTO, RESCUE and so on (figure 6.1). Using the PXE menu module means that the list of the MENU LABEL text values is displayed when the PXE boot menu is displayed, so that the associated kernel image and options can conveniently be selected by user intervention.

The PXE DEFAULT And MENU DEFAULT Instructions

DEFAULT: If the PXE menu module is not used and if no MENU instructions are used, and if there is no user intervention, then setting the same text that follows a LABEL tag immediately after the DEFAULT instruction, results in the associated kernel image and its options being run by default after the timeout.

By default, as already explained, the PXE menu module is used. In particular it uses the setting: DEFAULT menu.c32 to enable the menu.

MENU DEFAULT: If the PXE menu module is used and if MENU instructions are used, and if there is no user intervention, then setting a MENU DEFAULT tag as a line within the multiline LABEL statement results in the kernel image and options associated with that LABEL statement being loaded by default after the timeout.

The CMDaemon PXE Label Setting For Specific Nodes

The MENU DEFAULT value by default applies to every node using the software image that the PXE menu configuration file pxelinux.cfg/default is loaded from. To override its application on a per-node basis, the value of PXE Label can be set for each node. For example, using cmsh:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device use node001
[ddnmon61->device[node001]]% set pxelabel MEMTEST ; commit
```

Carrying it out for all nodes in the default category can be done, for example, with:

Example

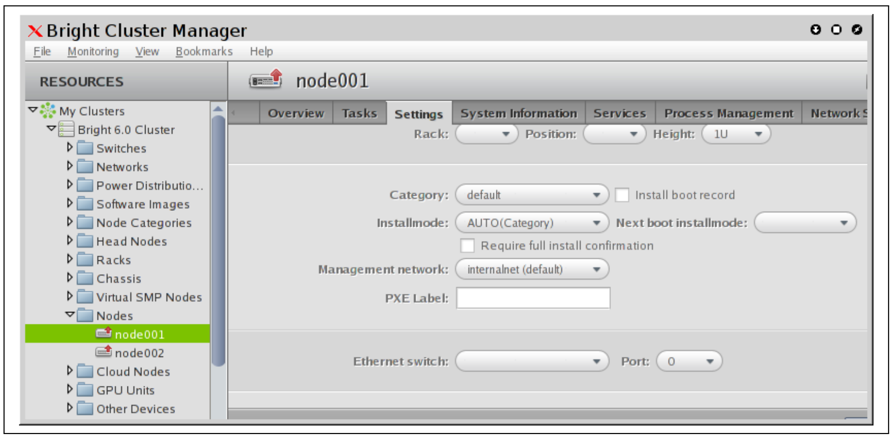
```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device
[ddnmon61->device]% foreach -c default (set pxelabel MEMTEST)
[ddnmon61->device*]% commit
```

The value of pxelabel can be cleared with:

Example

```
[root@ddnmon61 ~]# cmsh -c "device; foreach -c default (clear pxelabel)\
; commit"
```

In `cmgui`, the PXE label can be set from the `Settings` tab for a node (figure 6.2).



Rack:
Position:
Height: 1U
Category: default
Install boot record
Installmode: AUTO(Category)
Next boot installmode:
Require full install confirmation
Management network: internalnet (default)
PXE Label:
Ethernet switch:
Port: 0

Figure 6.2: `cmgui` PXE Label option

Changing The Install Mode Or Default Image Offered To Nodes

The selections offered by the PXE menu are pre-configured by default so that the `AUTO` menu option by default loads a kernel, runs the `AUTO` install mode, and eventually the `default-image` software image is provisioned.

Normally administrators should not be changing the install mode, kernel, or kernel options in the PXE menu configuration file `pxelinux.cfg/default`.

More on changing the install mode is given in section 6.4.4. More on changing software images, image package management, kernels, and kernel options, is to be found in Chapter 10.

6.1.2 iPXE Booting From A Disk Drive

Also by default, on disked nodes, iPXE software is placed on the drive during node installation. If the boot instructions from the BIOS for PXE booting fail, and if the BIOS instructions are that a boot attempt should then be made from the hard drive, it means that a PXE network boot attempt is done again, as instructed by the bootable hard drive. This can be a useful fallback option that works around certain BIOS features or problems.

6.1.3 iPXE Booting Using InfiniBand

On clusters that have InfiniBand hardware, it is normally used for data transfer as a service after the nodes have fully booted up (section 4.7). InfiniBand can also be used for PXE booting (described here) and used for node provisioning (section 6.3.3). However these uses are not necessary, even if InfiniBand is used for data transfer as a service later on, because booting and provisioning is available over Ethernet by default. This section (about boot over InfiniBand) may therefore safely be skipped when first configuring a cluster.

Booting over InfiniBand via PXE is enabled by carrying out these 3 steps:

1. Making the DirectMon aware that nodes are to be booted over InfiniBand. This can be done during the initial installation on the head node by marking the option "`Allow booting over InfiniBand`" (figure 2.11). Alternatively, if the cluster is already installed, then node booting (section 4.3.3, page 104) can be set from `cmsh` or `cmgui` as follows:

   (a) From `cmsh`'s `network` mode: If the InfiniBand network name is `ibnet`, then a `cmsh` command that will set it is:
   ```
   cmsh -c "network; set ibnet nodebooting yes;
   commit"
   ```

   (b) From `cmgui`'s "`Settings`" tab from the "`Networks`" resource (figure 4.6): The network item selected must be the InfiniBand network, "`ibnet`" by default, and the "`Allow node booting`" option is then set and saved.

   If the InfiniBand network does not yet exist, then it must be created (section 4.3.2). The recommended default values used are described in section 4.7.3.

   The administrator should also be aware that the interface from which a node boots, (conveniently labeled `BOOTIF`), must not be an interface that is already configured for that node in CMDaemon. For example, if `BOOTIF` is the device `ib0`, then `ib0` must not already be configured in CMDaemon . Either `BOOTIF` or the `ib0` configuration should be changed so that node installation can succeed.

2. Flashing iPXE onto the InfiniBand HCAs. (The ROM image is obtained from the HCA vendor).

3. Configuring the BIOS of the nodes to boot from the InfiniBand HCA.

Administrators who enable iPXE booting almost always wish to provision over InfiniBand too. Configuring provisioning over InfiniBand is described in section 6.3.3.

6.1.4 Booting From The Drive

Besides network boot, a node can also be configured to start booting and get to the stage of loading up its kernel entirely from its drive (section (6.4.4), just like a normal standalone machine.

6.1.5 The Boot Role

The action of providing a PXE boot image via DHCP and TFTP is known as providing *node booting*.

Roles in general are introduced in section 3.1.5. The *boot role* is one such role that can be assigned to a regular node. The boot role configures a regular node so that it can then provide node booting. The role cannot be assigned or removed from the head node—the head node always has a boot role.

The boot role is assigned by administrators to regular nodes if there is a need to cope with the scaling limitations of TFTP and DHCP. TFTP and DHCP services can be overwhelmed when there are large numbers of nodes making use of them during boot. An example of the scaling limitations may be observed, for example, when, during the powering up and PXE boot attempts of a large number of regular nodes from the head node, it turns out that random different regular nodes are unable to boot, apparently due to network effects.

One implementation of boot role assignment might therefore be, for example, to have a several groups of racks, with each rack in a subnet, and with one regular node in each subnet that is assigned the boot role. The boot role regular nodes would thus take the DHCP and TFTP load off the head node and onto themselves for all the nodes in their associated subnet, so that all nodes of the cluster are then able to boot without networking issues.

6.2 Provisioning Nodes

The action of transferring the software image to the nodes is called *node provisioning*, and is done by special nodes called the *provisioning nodes*. More complex clusters can have several provisioning nodes configured by the administrator, thereby distributing network traffic loads when many nodes are booting.

Creating provisioning nodes is done by assigning a *provisioning role* to a node or category of nodes. Similar to how the head node always has a boot role (section 6.1.5), the head node also always has a provisioning role.

6.2.1 Provisioning Nodes: Configuration Settings

The provisioning role has several parameters that can be set:

Property	Description
`allImages`	The following values decide what images the provisioning node provides: • `onlocaldisk`: all images on the local disk, regardless of any other parameters set • `onsharedstorage`: all images on the shared storage, regardless of any other parameters set • `no` (the default): only images listed in the `localimages` or `sharedimages` parameters, described next
`localimages`	A list of software images on the local disk that the provisioning node accesses and provides. The list is used only if `allImages` is "`no`".
`sharedimages`	A list of software images on the shared storage that the provisioning node accesses and provides. The list is used only if `allImages` is "`no`"
`maxProvisioningNodes`	The maximum number of nodes that can be provisioned in parallel by the provisioning node. The optimum number depends on the infrastructure. The default value is 10, which is safe for typical cluster setups. Setting it lower may sometimes be needed to prevent network and disk overload.
`nodegroups`	A list of node groups (section 3.1.4). If set, the provisioning node only provisions nodes in the listed groups. Conversely, nodes in one of these groups can only be provisioned by provisioning nodes that have that group set. Nodes without a group, or nodes in a group not listed in `nodegroups`, can only be provisioned by provisioning nodes that have no `nodegroups` values set. By default, the `nodegroups` list is unset in the provisioning nodes. The nodegroups setting is typically used to set up a convenient hierarchy of provisioning, for example based on grouping by rack and by groups of racks.

A provisioning node keeps a copy of all the images it provisions on its local drive, in the same directory as where the head node keeps such images. The local drive of a provisioning node must therefore have enough space available for these images, which may require changes in its disk layout.

6.2.2 Provisioning Nodes: Role Setup With cmsh

In the following `cmsh` example the administrator creates a new category called `misc`. The default category `default` already exists in a newly installed cluster.

The administrator then assigns the role called `provisioning` from the list of assignable roles to nodes in the `misc` category.

As an aside from the topic of provisioning, from an organizational perspective, other assignable roles include `monitoring`, `storage`, and `failover`. Tab-completion prompting after the `assign` command has been typed in lists all the possible roles.

The nodes in the `misc` category assigned the `provisioning` role then have `default-image` set as the image that they provision to other nodes, and have `20` set as the maximum number of other nodes to be provisioned simultaneously (some text is elided in the following example):

Example

```
[ddnmon61]% category add misc
[ddnmon61->category*[misc*]]% roles
[ddnmon61->category*[misc*]->roles]% assign provisioning
[ddnmon61...*]->roles*[provisioning*]]% set allimages false
[ddnmon61...*]->roles*[provisioning*]]% set localimages default-image
[ddnmon61...*]->roles*[provisioning*]]% set maxprovisioningnodes 20
[ddnmon61...*]->roles*[provisioning*]]% show
Parameter                Value
------------------------ ------------------------------------------
All Images               no
Local images             default-image
Name                     provisioning
Type                     ProvisioningRole
Nodegroups
maxProvisioningNodes     20
[ddnmon61->category*[misc*]->roles*[provisioning*]]% commit
[ddnmon61->category[misc]->roles[provisioning]]%
```

Assigning a `provisioning` role can also be done for an individual node instead, if using a category is deemed overkill:

Example

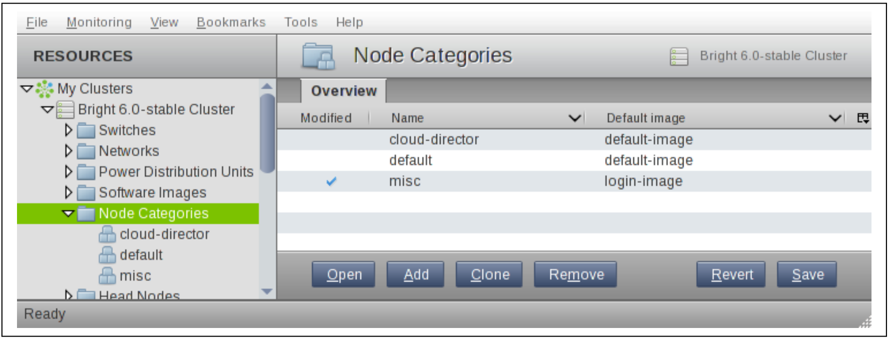
```
[ddnmon61]% device use node001
[ddnmon61->device[node001]]% roles
[ddnmon61->device[node001]->roles]% assign provisioning
[ddnmon61->device*[node001*]->roles*[provisioning*]]%
...
```

A role change configures a provisioning node, but does not directly update the provisioning node with images. After carrying out a role change, DirectMon runs the `updateprovisioners` command described in section 6.2.4 automatically, so that regular images are propagated to the provisioners. The propagation can be done by provisioners themselves if they have up-to-date images. CMDaemon tracks the provisioning nodes role changes, as well as which provisioning nodes have up-to-date images available, so that provisioning node configurations and regular node images propagate efficiently. Thus, for example, image update requests by provisioning nodes take priority over provisioning update requests from regular nodes.

6.2.3 Provisioning Nodes: Role Setup With `cmgui`

The provisioning configuration outlined in `cmsh` mode in section 6.2.2 can be done via `cmgui` too, as follows:

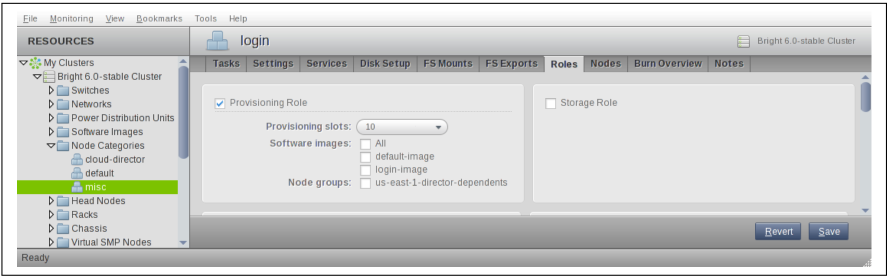
A `misc` category can be added by clicking on the `Add` button in the `Overview` tabbed pane in the `Node Categories` resource (figure 6.3).



Modified	Name	Default image
	cloud-director	default-image
	default	default-image
✓	misc	login-image

Figure 6.3: `cmgui`: Adding A `misc` Category

Clicking on the `misc` category in the resource tree on the left hand side (or alternatively, double-clicking on the `misc` category in the `Overview` tabbed pane of the `Node Categories` right hand side pane) opens the category up (figure 6.4).



✓ Provisioning Role

Provisioning slots: 10

Software images: ☐ All ☐ default-image ☐ login-image

Node groups: ☐ us-east-1-director-dependents

☐ Storage Role

Figure 6.4: `cmgui`: Configuring A `provisioning` Role

Selecting the `Roles` tab in this category displays roles that are part of the `misc` category. Ticking the checkbox of a role assigns the role to the category, and displays the settings that can be configured for this role. The `Provisioning slots` setting (`maxProvisioningNodes` in `cmsh`) decides how many images can be supplied simultaneously from the provisioning node, while the `Software images` settings (related to the `images` and `allimages` attributes of `cmsh`) decides what images the provisioning node supplies.

The `Software image` in the `Roles` tab should not be confused with the `Software image` selection possibility within the `Settings` tab, which is the image the provisioning node requests for itself.

6.2.4 Provisioning Nodes: Housekeeping

The head node does housekeeping tasks for the entire provisioning system. Provisioning is done on request for all non-head nodes on a first-come, first-serve basis. Since provisioning nodes themselves, too, need to be provisioned, it means that to cold boot an entire cluster up quickest, the head node should be booted and be up first, followed by provisioning nodes, and finally by all other non-head nodes. Following this start-up sequence ensures that all provisioning services are available when the other non-head nodes are started up.

Some aspects of provisioning housekeeping are discussed next:

Provisioning Node Selection

When a node requests provisioning, the head node allocates the task to a provisioning node. If there are several provisioning nodes that can provide the image required, then the task is allocated to the provisioning node with the lowest number of already-started provisioning tasks.

Limiting Provisioning Tasks With `MaxNumberOfProvisioningThreads`

Besides limiting how much simultaneous provisioning per provisioning node is allowed with `maxProvisioningNodes` (section 6.2.1), the head node also limits how many simultaneous provisioning tasks are allowed to run on the entire cluster. This is set using the `MaxNumberOfProvisioningThreads` directive in the head node's CMDaemon configuration file, `/etc/cmd.conf`, as described in Appendix C.

Provisioning Tasks Deferral and Failure

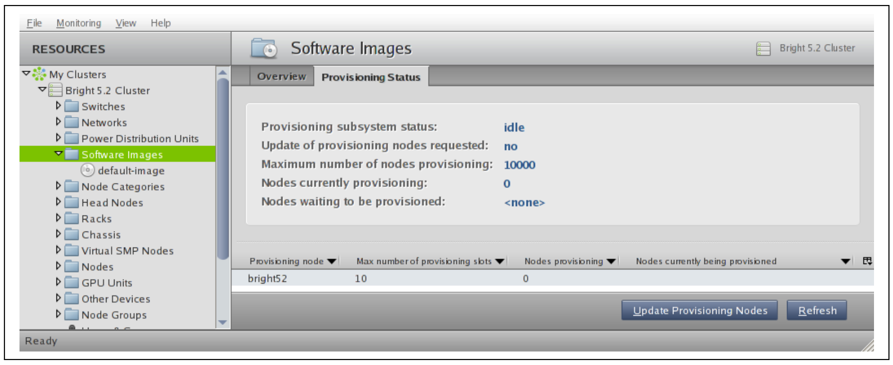
A provisioning request is *deferred* if the head node is not able to immediately allocate a provisioning node for the task. Whenever an ongoing provisioning task has finished, the head node tries to re-allocate deferred requests.

A provisioning request *fails* if an image is not transferred. 5 retry attempts at provisioning the image are made in case a provisioning request fails.

A provisioning node that is carrying out requests, and which loses connectivity, has its provisioning requests remain allocated to it for 180 seconds from the time that connectivity was lost. After this time the provisioning requests fail.

Provisioning Role Change Notification With `updateprovisioners`

The `updateprovisioners` command can be accessed from the `softwareimage` mode in `cmsh`. It can also be accessed from `cmgui` (figure 6.5):



```
Provisioning subsystem status:            idle
Update of provisioning nodes requested:   no
Maximum number of nodes provisioning:     10000
Nodes currently provisioning:             0
Nodes waiting to be provisioned:          <none>
```

Provisioning node	Max number of provisioning slots	Nodes provisioning	Nodes currently being provisioned
bright52	10	0	

Figure 6.5: `cmgui`: A Button To Update Provisioning Nodes

In the examples in section 6.2.2, changes were made to provisioning role attributes for an individual node as well as for a category of nodes. This automatically ran the `updateprovisioners` command.

The `updateprovisioners` command runs automatically if CMDaemon is involved during software image changes or during a provisioning request. If on the other hand, the software image is changed outside of the CMDaemon front ends (`cmgui` and `cmsh`), for example by an administrator adding a file by copying it into place from the bash prompt, then `updateprovisioners` should be run manually to update the provisioners.

In any case, if it is not run manually, there is also a scheduled time for it to run to ensure that it runs at least once every 24 hours.

When the default `updateprovisioners` is invoked manually, the provisioning system waits for all running provisioning tasks to end, and then updates all images located on any provisioning nodes by using the images on the head node. It also re-initializes its internal state with the updated provisioning role properties, i.e. keeps track of what nodes are provisioning nodes.

The default `updateprovisioners` command, run with no options, updates all images. If run from `cmsh` with a specified image as an option, then the command only does the updates for that particular image. A provisioning node undergoing an image update does not provision other nodes until the update is completed.

Example

```
[ddnmon61]% softwareimage updateprovisioners
Provisioning nodes will be updated in the background.

Sun Dec 12 13:45:09 2010 ddnmon61: Starting update of software image(s)\
 provisioning node(s). (user initiated).
[ddnmon61]% softwareimage updateprovisioners [ddnmon61]%
Sun Dec 12 13:45:41 2010 ddnmon61: Updating image default-image on prov\
isioning node node001.
[ddnmon61]%
Sun Dec 12 13:46:00 2010 ddnmon61: Updating image default-image on prov\
isioning node node001 completed.
Sun Dec 12 13:46:00 2010 ddnmon61: Provisioning node node001 was updated
Sun Dec 12 13:46:00 2010 ddnmon61: Finished updating software image(s) \
on provisioning node(s).
```

Provisioning Node Update Safeguards And `provisioningnodeautoupdatetimeout`

The `updateprovisioners` command is subject to safeguards that prevent it running too frequently. The minimum period between provisioning updates can be adjusted with the parameter `provisioningnodeautoupdatetimeout`, which has a default value of 300s.

When the head node receives a provisioning request, it checks if the last update of the provisioning nodes is more than the timeout period. If this is the case an update is triggered. The update is disabled if the timeout is set to zero (`false`).

The parameter can be accessed and set from `partition` mode:

Example

```
[root@ddnmon61 ]# cmsh
[ddnmon61]% partition use base
[ddnmon61->partition[base]]% get provisioningnodeautoupdatetimeout
[ddnmon61->partition[base]]% 300
[ddnmon61->partition[base]]% set provisioningnodeautoupdatetimeout 0
[ddnmon61->partition*[base*]]% commit
```

To prevent provisioning an image to the nodes, the `locked` state (section 6.4.7) can be used. A `locked` state stops the queued provisioning request from being executed, until the image is unlocked once more.

6.3 The Kernel Image, Ramdisk And Kernel Modules

A *software image* is a complete Linux file system that is to be installed on a non-head node. Chapter 10 describes images and their management in detail.

The head node holds the head copy of the software images. Whenever files in the head copy are changed using CMDaemon, the changes automatically propagate to all provisioning nodes via the `updateprovisioners` command (section 6.2.4).

6.3.1 Booting To A "Good State" Software Image

When nodes boot from the network in simple clusters, the head node supplies them with a *known good state* during node start up. The known good state is maintained by the administrator and is defined using a software image that is kept in a directory of the filesystem on the head node. Supplementary filesystems such as `/home` are served via NFS from the head node by default.

For a diskless node the known good state is copied over from the head node, after which the node becomes available to cluster users.

For a disked node, by default, the hard disk contents on specified local directories of the node are checked against the known good state on the head node. Content that differs on the node is changed to that of the known good state. After the changes are done, the node becomes available to cluster users.

Each software image contains a Linux kernel and a ramdisk. These are the first parts of the image that are loaded onto a node during early boot. The kernel is loaded first. The ramdisk is loaded next, and contains driver modules for the node's network card and local storage. The rest of the image is loaded after that, during the node-installer stage (section 6.4).

6.3.2 Selecting Kernel Driver Modules To Load Onto Nodes

Kernel Driver Modules With `cmsh`

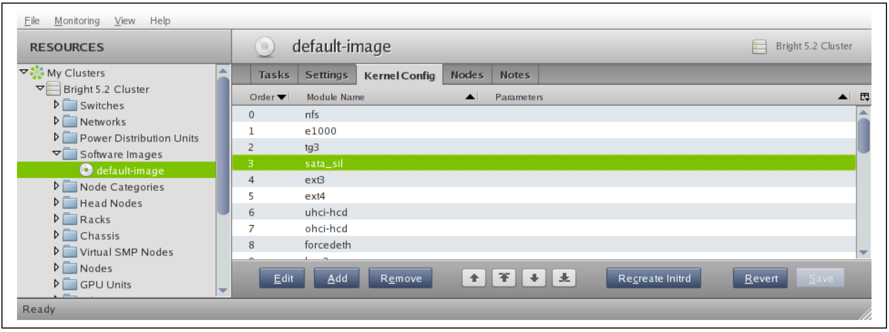
In `cmsh`, the modules that are to go on the ramdisk can be placed using the `kernelmodules` submode of the `softwareimage` mode. The order in which they are listed is the attempted load order.

Whenever a change is made via the `kernelmodules` submode to the kernel module selection of a software image, CMDaemon automatically runs the `createramdisk` command. The `createramdisk` command regenerates the ramdisk inside the initrd image and sends the updated image to all provisioning nodes, to the image directory, set by default to `/cm/images/default-image/boot/`. The original initrd image is saved as a file with suffix "`.orig`" in that directory. An attempt is made to generate the image for all software images that CMDaemon is aware of, regardless of category assignment, unless the image is protected from modification by CMDaemon with a `FrozenFile` directive (Appendix C).

The `createramdisk` command can also be run manually from within the `softwareimage` mode.

Kernel Driver Modules With `cmgui`

In `cmgui` the selection of kernel modules is done from by selecting the `Software Images` resource, and then choosing the "`Kernel Config`" tabbed pane (figure 6.6).



Order	Module Name	Parameters
0	nfs	
1	e1000	
2	tg3	
3	sata_sil	
4	ext3	
5	ext4	
6	uhci-hcd	
7	ohci-hcd	
8	forcedeth	

Figure 6.6: `cmgui`: Selecting Kernel Modules For Software Images

The order of module loading can be rearranged by selecting a module and clicking on the arrow keys. Clicking on the "`Recreate Initrd`" button runs the `createramdisk` command manually.

Manually Regenerating A Ramdisk

Regenerating a ramdisk manually via `cmsh` or `cmgui` is useful if the kernel or modules have changed without using CMDaemon. For example, after running a YUM update which has modified the kernel or modules of the nodes (section 10.3). In such a case, the distribution would normally update the ramdisk on the machine, but this is not done for the extended ramdisk for nodes in DirectMon. Not regenerating the DirectMon ramdisk for nodes after such an update means the nodes may fail on rebooting during the loading of the ramdisk (section 6.8.4).

An example of regenerating the ramdisk is seen in section 6.8.5.

Implementation Of Kernel Driver Via Ramdisk Or Kernel Parameter

Sometimes, testing or setting a kernel driver as a kernel parameter may be more convenient. How to do that is covered in section 10.3.4.

6.3.3 InfiniBand Provisioning

On clusters that have InfiniBand hardware, it is normally used for data transfer as a service after the nodes have fully booted up (section 4.7). It can also be used for PXE booting (section 6.1.3) and for node provisioning (described here), but these are not normally a requirement. This section (about InfiniBand node provisioning) may therefore safely be skipped in almost all cases when first configuring a cluster.

During node start-up on a setup for which InfiniBand networking has been enabled, the `init` process runs the `rdma` script. For SLES and distributions based on versions prior to Red Hat 6, the `openib` script is used instead of the `rdma` script. The script loads up InfiniBand modules into the kernel. When the cluster is finally fully up and running, the use of InfiniBand is thus available for all processes that request it. Enabling InfiniBand is normally set by configuring the InfiniBand network when installing the head node, during the `Additional Network Configuration` screen (figure 2.11).

Provisioning nodes over InfiniBand is not implemented by default, because the `init` process, which handles initialization scripts and daemons, takes place only after the node-provisioning stage launches. InfiniBand modules are therefore not available for use during provisioning, which is why, for default kernels, provisioning in DirectMon is done via Ethernet.

Provisioning at the faster InfiniBand speeds rather than Ethernet speeds is however a requirement for some clusters. To get the cluster to provision using InfiniBand requires both of the following two configuration changes to be carried out:

1. configuring InfiniBand drivers for the ramdisk image that the nodes first boot into, so that provisioning via InfiniBand is possible during this pre-`init` stage

2. defining the provisioning interface of nodes that are to be provisioned with InfiniBand. It is assumed that InfiniBand networking is already configured, as described in section 4.7.

   The administrator should be aware that the interface from which a node boots, (conveniently labeled `BOOTIF`), must not be an interface that is already configured for that node in CMDaemon. For example, if `BOOTIF` is the device `ib0`, then `ib0` must not already be configured in CMDaemon. Either `BOOTIF` or the `ib0` configuration should be changed so that node installation can succeed.

How these two changes are carried out is described next:

InfiniBand Provisioning: Ramdisk Image Configuration

An easy way to see what modules must be added to the ramdisk for a particular HCA can be found by running `rdma` (or `openibd`), and seeing what modules do load up on a fully booted system.

One way to do this is to run the following lines as root:

```
[root@ddnmon61 ~]# { service rdma stop; lsmod | cut -f1 -d" "; }>/tmp/a
[root@ddnmon61 ~]# { service rdma start; lsmod | cut -f1 -d" "; }>/tmp/b
```

The rdma service in the two steps should be replaced by openibd service instead when using SLES, or distributions based on versions of Red Hat prior to version 6.

The first step stops the InfiniBand service, just in case it is running, in order to unload its modules, and then lists the modules on the node. The second step then starts the service, so that the appropriate modules are loaded, and then lists the modules on the node again. The output of each step is stored in files a and b here.

Running diff on the output of these two steps then reveals the modules that get loaded. For rdma, the output may display something like:

Example

```
[root@ddnmon61 ~]# diff /tmp/a /tmp/b
1,3c1
< Unloading OpenIB kernel modules:
< Failed to unload ib_core
<                                                           [FAILED]
---
> Loading OpenIB kernel modules:                            [  OK  ]
4a3,14
> ib_ipoib
> rdma_ucm
> ib_ucm
> ib_uverbs
> ib_umad
> rdma_cm
> ib_cm
> iw_cm
> ib_addr
> ib_sa
> ib_mad
> ib_core
```

As suggested by the output, the modules ib_ipoib, rdma_ucm and so on are the modules loaded when rdma starts, and are therefore the modules that are needed for this particular HCA. Other HCAs may cause different modules to be loaded.

The modules should then be part of the initrd image in order to allow InfiniBand to be used during the node provisioning stage.

The initrd image for the nodes is created by adding the required InfiniBand kernel modules to it. How to load kernel modules into a ramdisk is covered more generally in section 6.3.2. A typical Mellanox HCA may have it created as follows (some text elided in the following example):

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% softwareimage use default-image
[ddnmon61->softwareimage[default-image]]% kernelmodules
[ddnmon61...age[default-image]->kernelmodules]% add mlx4_ib
[ddnmon61...age*[default-image*]->kernelmodules*[mlx4_ib*]]% add ib_ipoib
```

```
[ddnmon61...age*[default-image*]->kernelmodules*[ib_ipoib*]]% add ib_umad
[ddnmon61...age*[default-image*]->kernelmodules*[ib_umad*]]% commit
[ddnmon61->softwareimage[default-image]->kernelmodules[ib_umad]]%
Tue May 24 03:45:35 2011 ddnmon61: Initial ramdisk for image default-im\
age was regenerated successfully.
```

If the modules are put in another image instead of `default-image`, then the default image that nodes boot from should be set to the new image (section 4.16.2).

InfiniBand Provisioning: Network Configuration

It is assumed that the networking configuration for the final system for InfiniBand is configured following the general guidelines of section 4.7. If it is not, that should be checked first to see if all is well with the InfiniBand network.

The provisioning aspect is set by defining the provisioning interface. An example of how it may be set up for 150 nodes with a working InfiniBand interface `ib0` in `cmsh` is:

Example

```
[root@ddnmon61~]# cmsh
%[ddnmon61]% device
[ddnmon61->device]% foreach -n node001..node150 (set provisioninginter\
face ib0)
[ddnmon61->device*]% commit
```

6.4 Node-Installer

After the kernel has started up, and the ramdisk kernel modules are in place on the node, the node launches the node-installer.

The node-installer interacts with CMDaemon on the head node and takes care of the rest of the boot process.

As an aside, the node-installer modifies some files (section A.3) on the node it is installing to, so that they differ from the otherwise-expected pre-init stage Linux system. Such modifications can be prevented by a `frozenFilesPerNode` or `frozenFilesPerCategory` directive, as documented within `/cm/node-installer/scripts/node-installer.conf`.

Once the node-installer has completed its tasks, the local drive of the node has a complete Linux pre-init stage system. The node-installer ends by calling `/sbin/init` from the local drive and the boot process then proceeds as a normal Linux boot.

The steps the node-installer goes through for each node are:

1. requesting a node certificate (section 6.4.1)
2. deciding or selecting node configuration (section 6.4.2)
3. starting up all network interfaces (section 6.4.3)
4. determining install-mode type and execution mode (section 6.4.4)
5. running `initialize` scripts (section 6.4.5)
6. checking partitions, mounting filesystems (section 6.4.6)

7. synchronizing the local drive with the correct software image (section 6.4.7)
8. writing network configuration files to the local drive (section 6.4.8)
9. creating an `/etc/fstab` file on the local drive (section 6.4.9)
10. installing GRUB bootloader (section 6.4.10), if configured, and initializing SELinux (Appendix P), if configured
11. running `finalize` scripts (section 6.4.11)
12. unloading specific drivers no longer needed (section 6.4.12)
13. switching the root device to the local drive and calling `/sbin/init` (section 6.4.13)

These 13 node-installer steps and related matters are described in detail in the corresponding sections 6.4.1–6.4.13.

6.4.1 Requesting A Node Certificate

Each node communicates with the CMDaemon on the head node using a certificate. If no certificate is found, it automatically requests one from CMDaemon running on the head node (figure 6.7).

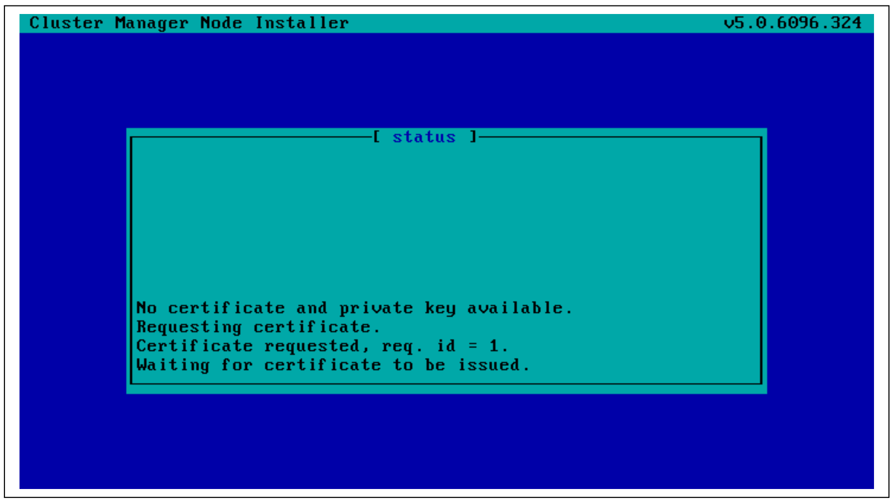

```
Cluster Manager Node Installer                                  v5.0.6096.324

[ status ]

No certificate and private key available.
Requesting certificate.
Certificate requested, req. id = 1.
Waiting for certificate to be issued.
```

Figure 6.7: Certificate Request

The certificate is stored on the head node in `/cm/node-installer/certificates/` by MAC address.

Null Cipher Certificates

By default, a null cipher is used on internal networks such as `internalnet`, to keep communications speedy. Using encryption on even these networks is sometimes a requirement in very unusual situations. In that case, setting the advanced configuration flag `AllowNullCipherNetwork=0` in `cmd.conf` (appendix C) forces encryption on after CMDaemon is restarted. By default, its value is `1`.

Certificate Auto-signing

By default, *certificate auto-signing* means the cluster management daemon automatically issues a certificate to any node that requests a certificate.

For untrusted networks it may be wiser to approve certificate requests manually to prevent new nodes being added automatically without getting noticed. Disabling certificate auto-signing can then be done by issuing the `autosign off` command from `cert` mode in `cmsh`.

Section 3.3 has more information on certificate management in general.

Example

Disabling certificate auto-sign mode:

```
[ddnmon61]% cert autosign
on
[ddnmon61]% cert autosign off
off
[ddnmon61]% cert autosign
off
[ddnmon61]%
```

Certificate Storage And Removal Implications

After receiving a valid certificate, the node-installer stores it in `/cm/node-installer/certificates/<node mac address>/` on the head node. This directory is NFS exported to the nodes, but can only be accessed by the `root` user. The node-installer does not request a new certificate if it finds a certificate in this directory, valid or invalid.

If an invalid certificate is received, the screen displays a communication error. Removing the node's corresponding certificate directory allows the node-installer to request a new certificate and proceed further.

6.4.2 Deciding Or Selecting Node Configuration

Once communication with the head node CMDaemon is established, the node-installer tries to identify the node it is running on so that it can select a configuration from CMDaemon's record for it, if any such record exists. It correlates any node configuration the node is expected to have according to network hardware detected. If there are issues during this correlation process then the administrator is prompted to select a node configuration until all nodes finally have a configuration.

Possible Node Configuration Scenarios

The correlations process and corresponding scenarios are now covered in more detail:

It starts with the node-installer sending a query to CMDaemon to check if the MAC address used for net booting the node is already associated with a node in the records of CMDaemon. In particular, it checks the MAC address for a match against the existing *node configuration* properties, and decides whether the node is *known* or *new*.

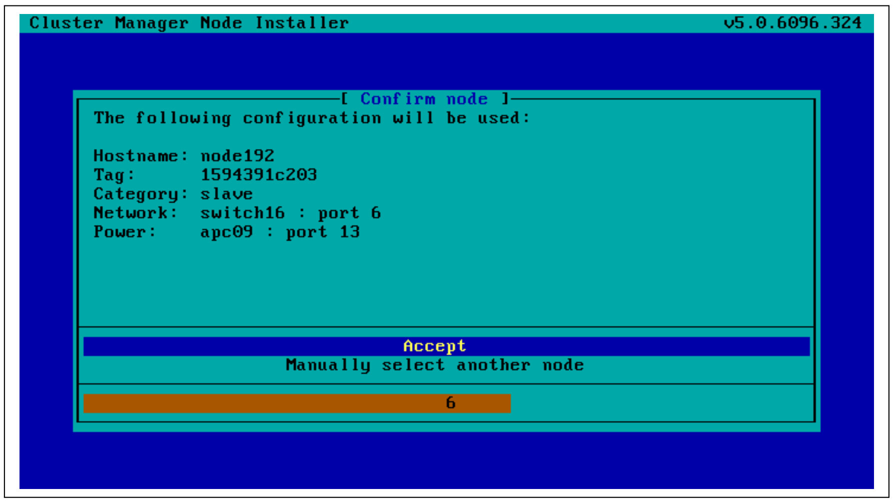
- the node is **known** if the query matches a node configuration. It means that node has been booted before.
- the node is **new** if no configuration is found.

In both cases the node-installer then asks CMDaemon to find out if the node is connected to an Ethernet switch, and if so, to which port. Setting up Ethernet switches for port detection is covered in section 4.9.

If a port is detected for the node, the node-installer queries CMDaemon for a node configuration associated with the detected Ethernet switch port. If a port is not detected for the node, then either the hardware involved with port detection needs checking, or a node configuration must be selected manually.

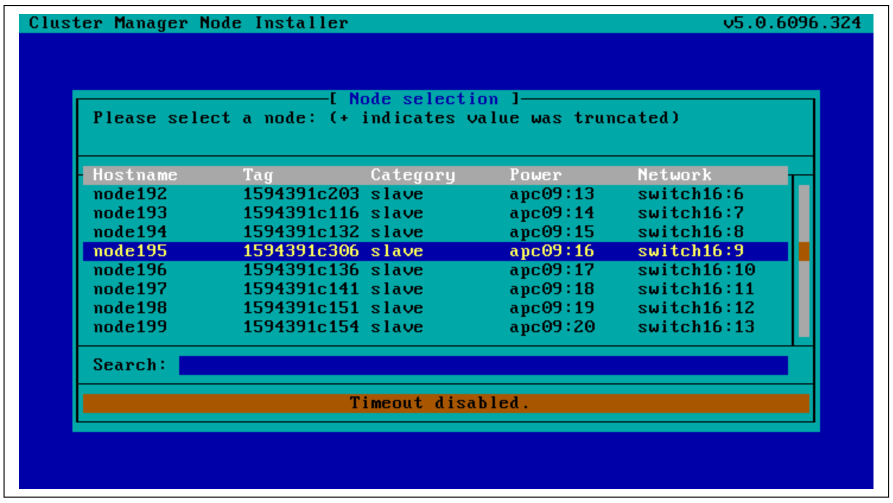
There are thus several scenarios:

1. The node is new, and an Ethernet switch port is detected. A node configuration associated with the port is found. The node-installer suggests to the administrator that the new node should use this configuration, and displays the configuration along with a confirmation dialog (figure 6.8). This suggestion can be interrupted, and other node configurations can be selected manually instead through a sub-dialog (figure 6.9). By default (in the main dialog), the original suggestion is accepted after a timeout.



```
Cluster Manager Node Installer                                    v5.0.6096.324

                         [ Confirm node ]
 The following configuration will be used:

 Hostname: node192
 Tag:      1594391c203
 Category: slave
 Network:  switch16 : port 6
 Power:    apc09 : port 13
```

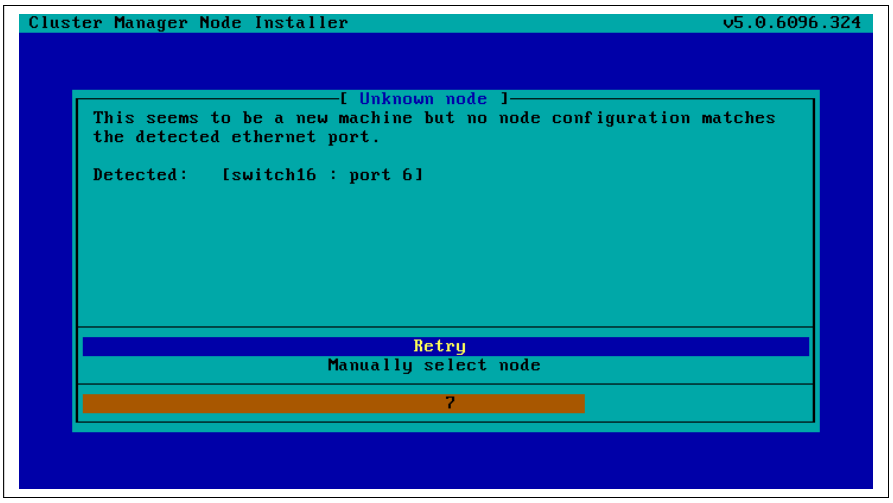
Figure 6.8: Scenarios: Configuration Found, Confirm Node Configuration



```
Cluster Manager Node Installer                                  v5.0.6096.324

              [ Node selection ]
  Please select a node: (+ indicates value was truncated)

  Hostname     Tag          Category    Power      Network
  node192      1594391c203  slave       apc09:13   switch16:6
  node193      1594391c116  slave       apc09:14   switch16:7
  node194      1594391c132  slave       apc09:15   switch16:8
  node195      1594391c306  slave       apc09:16   switch16:9
  node196      1594391c136  slave       apc09:17   switch16:10
  node197      1594391c141  slave       apc09:18   switch16:11
  node198      1594391c151  slave       apc09:19   switch16:12
  node199      1594391c154  slave       apc09:20   switch16:13

  Search:

                       Timeout disabled.
```

Figure 6.9: Scenarios: Node Selection Sub-Dialog

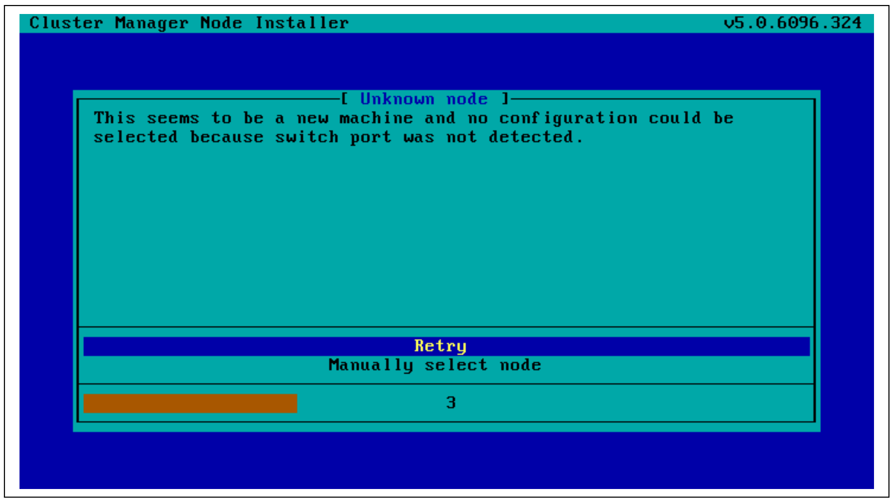
2. The node is new, and an Ethernet switch port is detected. A node configuration associated with the port is not found. The node-installer then displays a dialog that allows the administrator to either retry Ethernet switch port detection (figure 6.10) or to drop into a sub-dialog to manually select a node configuration (figure 6.9). By default, port detection is retried after a timeout.



```
Cluster Manager Node Installer                                  v5.0.6096.324

              [ Unknown node ]
  This seems to be a new machine but no node configuration matches
  the detected ethernet port.

  Detected:   [switch16 : port 6]

                            Retry
                     Manually select node

                              7
```

Figure 6.10: Scenarios: Unknown Node, Ethernet Port Detected

3. The node is new, and an Ethernet switch port is not detected. The node-installer then displays a dialog that allows the user to either retry Ethernet switch port detection (figure 6.11) or to drop into a sub-dialog to manually select a node configuration (figure 6.9). By default, port detection is retried after a timeout.

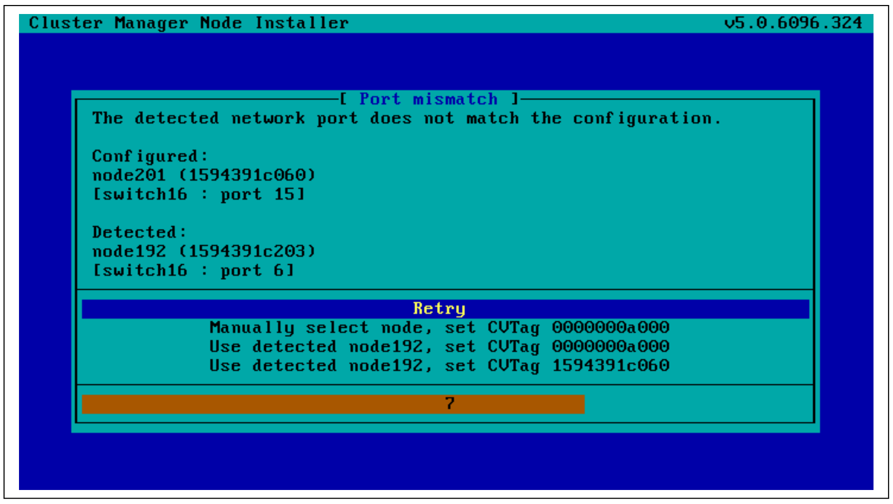


```
Cluster Manager Node Installer                                 v5.0.6096.324

              [ Unknown node ]
  This seems to be a new machine and no configuration could be
  selected because switch port was not detected.

                              Retry
                       Manually select node

                                3
```

Figure 6.11: Scenarios: Unknown Node, No Ethernet Port Detected

4. The node is known, and an Ethernet switch port is detected. The configuration associated with the port is the same as the configuration associated with the node's MAC address. The node-installer then displays the configuration as a suggestion along with a confirmation dialog (figure 6.8). The suggestion can be interrupted, and other node configurations can be selected manually instead through a sub-dialog (figure 6.9). By default (in the main dialog), the original suggestion is accepted after a timeout.

5. The node is known, and an Ethernet switch port is detected. However, the configuration associated with the port is not the same as the configuration associated with the node's MAC address. This is called a *port mismatch*. This type of port mismatch situation occurs typically during a mistaken *node swap*, when two nodes are taken out of the cluster and returned, but their positions are swapped by mistake (or equivalently, they are returned to the correct place in the cluster, but the switch ports they connect to are swapped by mistake). To prevent configuration mistakes, the node-installer displays a port mismatch dialog (figure 6.12) allowing the user to retry, accept a node configuration that is associated with the detected Ethernet port, or to manually select another node configuration via a sub-dialog (figure 6.9). By default (in the main port mismatch dialog), port detection is retried after a timeout.



```
Cluster Manager Node Installer                                   v5.0.6096.324

                  [ Port mismatch ]
 The detected network port does not match the configuration.

 Configured:
 node201 (1594391c060)
 [switch16 : port 15]

 Detected:
 node192 (1594391c203)
 [switch16 : port 6]

                              Retry
           Manually select node, set CVTag 0000000a000
           Use detected node192, set CVTag 0000000a000
           Use detected node192, set CVTag 1594391c060

                              ?
```

Figure 6.12: Scenarios: Port Mismatch Dialog

6. The node is known, and an Ethernet switch port is not detected. However, the configuration associated with the node's MAC address does have an Ethernet port associated with it. This is also considered a port mismatch. To prevent configuration mistakes, the node-installer displays a port mismatch dialog similar to figure 6.12, allowing the user to retry or to drop into a sub-dialog and manually select a node configuration that may work.

   However, a more likely solution in most cases is to:

   - either clear the switch port configuration in the cluster manager so that switch port detection is not attempted. For example, for `node001`, this can be done by running this `cmsh` command on the head node:

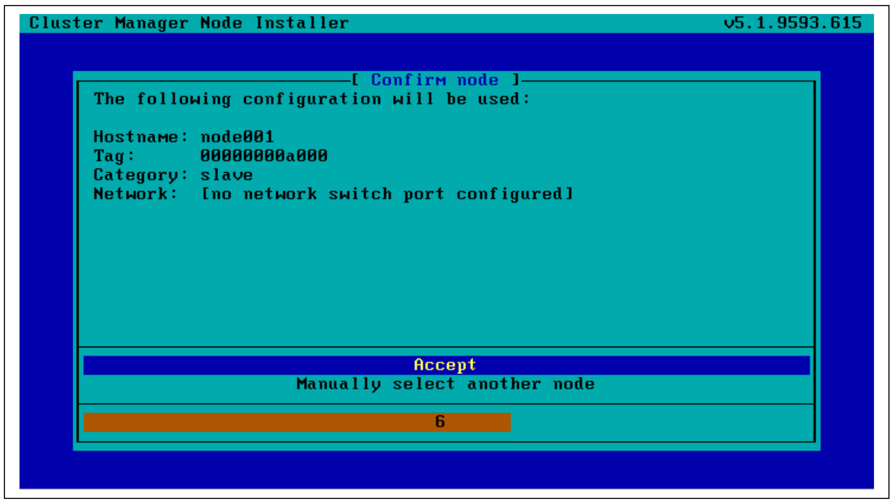
     ```
     cmsh -c "device clear node001 ethernetswitch"
     ```

   - or enable switch port detection on the switch. This is usually quite straightforward, but may require going through the manuals or software application that the switch manufacturer has provided.

   By default (in the port mismatch dialog), port detection is retried after a timeout. This means that if the administrator clears the switch port configuration or enables switch port detection, the node-installer is able to continue automatically with a consistent configuration.

7. The node is known, and an Ethernet switch port is detected. However, the configuration associated with the node's MAC address has no Ethernet switch port associated with it. This is not considered a port mismatch but an unset switch port configuration, and it typically occurs if switch port configuration has not been carried out, whether by mistake or deliberately. The node-installer displays the configuration as a suggestion along with a confirmation dialog (figure 6.13). The suggestion can be interrupted, and other node configurations can be selected manually instead using a sub-dialog. By

default (in the main dialog) the configuration is accepted after a timeout.



```
Cluster Manager Node Installer                                          v5.1.9593.615

                        [ Confirm node ]
  The following configuration will be used:

  Hostname: node001
  Tag:      00000000a000
  Category: slave
  Network:  [no network switch port configured]
```

Figure 6.13: Scenarios: Port Unset Dialog

A truth table summarizing the scenarios is helpful:

Scenario	Node known?	Switch port detected?	Switch port configuration found?	Switch port configuration conflicts with node configuration?
1	No	Yes	Yes	No
2	No	Yes	No	No
3	No	No	No	No
4	Yes	Yes	Yes	No
5	Yes	Yes	Yes	Yes (configurations differ)
6	Yes	No	Yes	Yes (port expected by MAC configuration not found)
7	Yes	Yes	No	No (port not expected by MAC configuration)

In these scenarios, whenever the user manually selects a node configuration in the prompt dialog, an attempt to detect an Ethernet switch port is repeated. If a port mismatch still occurs, it is handled by the system as if the user has not made a selection.

Summary Of Behavior During Hardware Changes

The logic of the scenarios means that an unpreconfigured node always boots to a dialog loop requiring manual intervention during a first install (scenarios 2 and 3). For subsequent boots the behavior is:

- If the node MAC hardware has changed (scenarios 1, 2, 3):
  - if the node is new and the detected port has a configuration, the node automatically boots to that configuration (scenario 1).
  - else manual intervention is needed (scenarios 2, 3)
- If the node MAC hardware has not changed (scenarios 4, 5, 6, 7):
  - if there is no port mismatch, the node automatically boots to its last configuration (scenarios 4, 7).
  - else manual intervention is needed (scenarios 5, 6).

The `newnodes` Command

`newnodes` **basic use:** New nodes that have not been configured yet can be detected using the `newnodes` command from within the `device` mode of `cmsh`. A new node is detected when it reaches the node-installer stage after booting, and contacts the head node.

Example

```
[ddnmon61->device]% newnodes
The following nodes (in order of appearance) are waiting to be assigned:
MAC                First appeared                  Detected on switch port
-----------------  -----------------------------  -----------------------
00:0C:29:01:0F:F8  Mon, 14 Feb 2011 10:16:00 CET   [no port detected]
```

At this point the node-installer is seen by the administrator to be looping, waiting for input on what node name is to be assigned to the new node.

The nodes can be uniquely identified by their MAC address or switch port address.

The port and switch to which a particular MAC address is connected can be discovered by using the `showport` command (section 4.9.4). After confirming that they are appropriate, the `ethernetswitch` property for the specified device can be `set` to the port and switch values.

Example

```
[ddnmon61->device]% showport 00:0C:29:01:0F:F8
switch01:8
[ddnmon61->device]% set node003 ethernetswitch switch01:8
[ddnmon61->device*]% commit
```

When the node name (`node003` in the preceding example) is assigned, the node-installer stops looping and goes ahead with the installation to the node.

The preceding basic use of `newnodes` is useful for small numbers of nodes. For larger number of nodes, the advanced options of `newnodes` may help carry out node-to-MAC assignment with less effort.

`newnodes` **advanced use—options:** The list of MAC addresses discovered by a `newnodes` command can be assigned in various ways to nodes specified by the administrator. Node objects should be created in advance to allow the assignment to take place. The easiest way to set up node objects is to use the `--clone` option of the `foreach` command (section 3.5.5).

The advanced options of `newnodes` are particularly useful for quickly assigning node names to specific physical nodes. All that is needed is to power the nodes up in the right order. For nodes with the same hardware, the node that is powered up first reaches the stage where it tries to connect with the node-installer first. So its MAC address is detected first, and arrives on the list generated by `newnodes` first. If some time after the first node is powered up, the second node is powered up, then its MAC address becomes the second MAC address on the list, and so on for the third, fourth, and further nodes.

When assigning node names to a physical node, on a cluster that has no such assignment already, the first node that arrived on the list gets assigned the name `node001`, the second node that arrived on the list gets assigned the name `node002` and so on.

The advanced options are shown in `device` mode by running the `help newnodes` command. The options can be introduced as being of three kinds: straightforward, grouping, and miscellaneous:

- The straightforward options:

  `-n|--nodes`

  `-w|--write`

  `-s|--save`

  Usually the most straightforward way to assign the nodes is to use the `-n` option, which accepts a list of nodes, together with a `-w` or `-s` option. The `-w` (`--write`) option sets the order of nodes to the corresponding order of listed MAC addresses, and is the same as setting an object in `cmsh`. The `-s` (`--save`) option is the same as setting and committing an object in `cmsh`, so `-s` implies a `-w` option is run at the same time.

  So, for example, if 8 new nodes are discovered by the node-installer on a cluster with no nodes so far, then:

  Example

  ```
  [ddnmon61->device]% newnodes -w -n node001..node008
  ```

  assigns (but does not commit) the sequence node001 to node008 the new MAC address according to the sequence of MAC addresses displaying on the list.

- The grouping options:

  `-g|--group`

  `-c|--category`

  `-h|--chassis`

  `-r|--rack`

The "`help newnodes`" command in `device` mode shows assignment options other than `-n` for a node range are possible. For example, the assignments can also be made for a group (`-g`), per category (`-c`), per chassis (`-h`), and per rack (`-r`).

- The miscellaneous options:

  `-f|--force`

  `-o|--offset`

  By default, the `newnodes` command fails when it attempts to set a node name that is already taken. The `-f` (`--force`) option forces the new MAC address to be associated with the old node name. When used with an assignment grouping, (node range, group, category, chassis, or rack) all the nodes in the grouping lose their node-to-MAC assignments and get new assignments. The `-f` option should therefore be used with care.

  The `-o` (`--offset`) option takes a number <*number*> and skips <*number*> nodes in the list of detected unknown nodes, before setting or saving values from the assignment grouping.

Examples of how to use the advanced options follow.

`newnodes` **advanced use—range assignment behavior example:** For example, supposing there is a cluster with nodes assigned all the way up to node022. That is, CMDaemon knows what node is assigned to what MAC address. For the discussion that follows, the three nodes node020, node021, node022 can be imagined as being physically in a rack of their own. This is simply to help to visualize a layout in the discussion and tables that follow and has no other significance. An additional 3 new, that is unassigned, nodes are placed in the rack, and allowed to boot and get to the node-installer stage.

The `newnodes` command discovers the new MAC addresses of the new nodes when they reach their node-installer stage, as before (the switch port column is omitted in the following text for convenience):

Example

```
[ddnmon61->device]% newnodes
MAC                 First appeared
------------------- ------------------------------
00:0C:29:EF:40:2A   Tue, 01 Nov 2011 11:42:31 CET
00:0C:29:95:D3:5B   Tue, 01 Nov 2011 11:46:25 CET
00:0C:29:65:9A:3C   Tue, 01 Nov 2011 11:47:13 CET
```

The assignment of MAC to node address could be carried out as follows:

Example

```
[ddnmon61->device]% newnodes -s -n node023..node025
MAC                  First appeared                 Hostname
-------------------- ------------------------------ --------
00:0C:29:EF:40:2A    Tue, 01 Nov 2011 11:42:31 CET  node023
00:0C:29:95:D3:5B    Tue, 01 Nov 2011 11:46:25 CET  node024
00:0C:29:65:9A:3C    Tue, 01 Nov 2011 11:47:13 CET  node025
```

Once this is done, the node-installer is able to stop looping, and to go ahead and install the new nodes with an image.

The physical layout in the rack may then look as indicated by this:

before	**after**	MAC
node020	node020	
node021	node021	
node022	node022	
	node023	...A
	node024	...B
	node025	...C

Here, `node023` is the node with the MAC address ending in `A`.

If instead of the previous `newnodes` command, an offset of 1 is used to skip assigning the first new node:

Example

```
[ddnmon61->device]% newnodes -s -o 1 node024..node025
```

then the rack layout looks like:

before	**after**	MAC
node020	node020	
node021	node021	
node022	node022	
	unassigned	...A
	node024	...B
	node025	...C

Here, *unassigned* is where `node023` of the previous example is physically located, that is, the node with the MAC address `...A`. The lack of assignment means there is actually no association of the name `node023` with that MAC address, due to the `newnodes` command having skipped over it with the `-o` option.

If instead the assignment is done with:

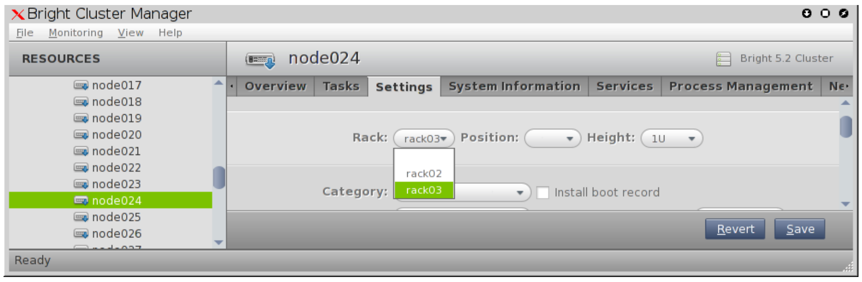
Example

```
[ddnmon61->device]% newnodes -s 1 node024..node026
```

then the `node023` name is unassigned, and the name `node024` is assigned instead to the node with the MAC address `...A`, so that the rack layout looks like:

before	after	MAC
node020	node020	
node021	node021	
node022	node022	
	node024	...A
	node025	...B
	node026	...C

`newnodes` **advanced use—assignment grouping example:** Node range assignments are one way of using `newnodes`. However assignments can also be made to a category, a rack, or a chassis. For example, with `cmgui`, assigning node names to a rack can be done from the `Nodes` resource and selecting the `Settings` tab. Within the tab, the `Rack` values can be set appropriately, and saved for each node (figure 6.14).



Rack: rack03 Position: Height: 1U

Category:

Figure 6.14: Assigning A Node To A Rack

In `cmsh`, the assignment of multiple node names to a rack can conveniently be done with a `foreach` loop from within `device` mode:

Example

```
[ddnmon61->device]% foreach -n node020..node029 (set rack rack02)
[ddnmon61->device*]% commit
[ddnmon61->device]% foreach -n node030..node039 (set rack rack03)
[ddnmon61->device*]% commit
```

The assignment of node names with the physical node in the rack can then be arranged as follows: If the nodes are identical hardware, and are powered up in numerical sequence, from `node020` to `node039`, with a few seconds in between, then the list that the basic `newnodes` command (without options) displays is arranged in the same numerical sequence. Assigning the list in the rack order can then be done by running:

Example

```
[ddnmon61->device]% newnodes -s -r rack02..rack03
```

If it turns out that the boot order was done very randomly and incorrectly for all of `rack02`, and that the assignment for `rack02` needs to be done again, then a simple way to deal with it is to clear out all of the `rack02` current MAC associations, and redo them according to the correct boot order:

Example

```
[ddnmon61->device]% foreach -r rack02 ( clear mac ) ; commit
  [...removes MAC association with nodes from CMDaemon...]

  [...now reboot nodes in rack02 in sequence...]

[ddnmon61->device]% newnodes

  [...shows sequence as the nodes come up..]

[ddnmon61->device]% newnodes -s -r rack02

  [...assigns sequence in boot order...]
```

`newnodes` **advanced use—assignment forcing example:** The `--force` option can be used in the following case: Supposing that `node022` fails, and a new node hardware comes in to replace it. The new regular node has a new MAC address. So, as explained by scenario 3 (section 6.4.2), if there is no switch port assignment in operation for the nodes, then the node-installer loops around, waiting for intervention.[1]

This situation can be dealt with from the command line by:

- accepting the node configuration at the regular node console, via a sub-dialog
- accepting the node configuration via `cmsh`, without needing to be at the regular node console:

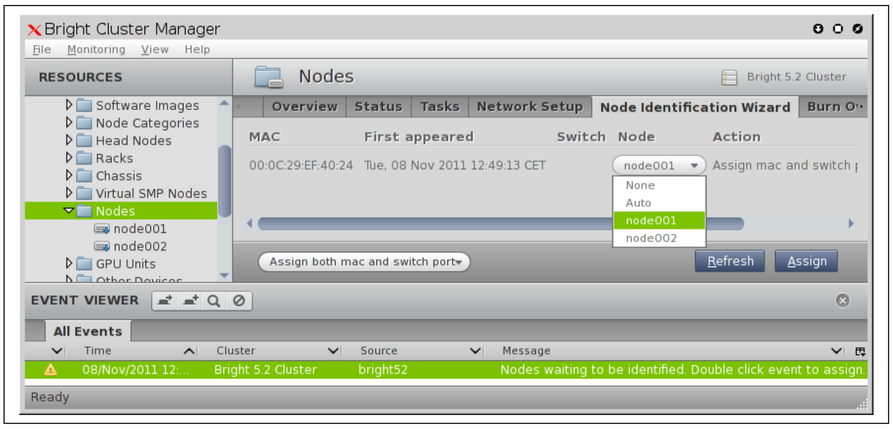
```
[ddnmon61->device]% newnodes -s -f -n node022
```

Node Identification Wizard

The *node identification wizard* can be accessed from the tabbed pane under the `Nodes` resource (figure 6.15). It can also be accessed by double-clicking, in the event viewer, on the event message "`Nodes waiting to be identified.  Double click event to assign`".

The node identification wizard is roughly the `cmgui` equivalent to the `newnodes` command of `cmsh`. Like `newnodes`, the wizard lists the MAC address and switch port of any unassigned node that the head node detects. Also, like `newnodes`, it can help assign a node name to the node, assuming the node object exists. After assignment is done, a prompt to save the new status appears.

[1]with switch port assignment in place, scenario 1 means the new node simply boots up by default and becomes the new `node022` without further intervention



MAC	First appeared	Switch	Node	Action
00:0C:29:EF:40:24	Tue, 08 Nov 2011 12:49:13 CET		node001	Assign mac and switch p

Time	Cluster	Source	Message
08/Nov/2011 12:...	Bright 5.2 Cluster	bright52	Nodes waiting to be identified. Double click event to assign

Figure 6.15: Node Identification Wizard

The most useful way of using the wizard is for node assignment in large clusters.

To do this, it is assumed that the node objects have already been created for the new nodes. The creation of the node objects means that the node names exist, and so assignment to the node names is able to take place. An easy way to create nodes, set their provisioning interface, and set their IP addresses is described in the section on the *node creation wizard* (section 6.7.2). Node objects can also be created by running `cmsh`'s `foreach` loop command on a node with a `--clone` option (section 3.5.5).

The nodes are also assumed to be set for net booting, typically set from a BIOS setting.

The physical nodes are then powered up in an arranged order. Because they are unknown new nodes, the node-installer keeps looping after a timeout. The head node in the meantime detects the new MAC addresses and switch ports in the sequence in which they first have come up and lists them in that order.

By default, all these newly detected nodes are set to `auto`, which means their numbering goes up sequentially from whatever number is assigned to the preceding node in the list. Thus, if there are 10 new unassigned nodes that are brought into the cluster, and the first node in the list is assigned to the first available number, say `node327`; then clicking on assign automatically assigns the remaining nodes to the next available numbers, say `node328-node337`.

After the assignment, the node-installer looping process on the new nodes notices that the nodes are now known. The node-installer then breaks out of the loop, and installation goes ahead without any intervention needed at the node console.

6.4.3 Starting Up All Network Interfaces

At the end of section 6.4.2, the node-installer knows which node it is running on, and has decided what its node configuration is.

It now gets on with setting up the IP addresses on the interfaces required for the node-installer, while taking care of matters that come up on the way:

Avoiding Duplicate IP Addresses

The node-installer brings up all the network interfaces configured for the node. Before starting each interface, the node-installer first checks if the IP address that is about to be used is not already in use by another device. If it is, then a warning and retry dialog is displayed until the IP address conflict is resolved.

Using `BOOTIF` To Specify The Boot Interface

`BOOTIF` is a special name for one of the possible interfaces. The node-installer automatically translates `BOOTIF` into the name of the device, such as eth0 or eth1, used for network booting. This is useful for a machine with multiple network interfaces where it can be unclear whether to specify, for example, eth0 or eth1 for the interface that was used for booting. Using the name `BOOTIF` instead means that the underlying device, eth0 or eth1 in this example, does not need to be specified in the first place.

Halting On Missing Kernel Modules For The Interface

For some interface types like VLAN and channel bonding, the node-installer halts if the required kernel modules are not loaded or are loaded with the wrong module options. In this case the kernel modules configuration for the relevant software image should be reviewed. Recreating the ramdisk and rebooting the node to get the interfaces up again may be necessary, as described in section 6.8.5.

Initializing BMC Interfaces

BMC interfaces, if present and set up in the node's configuration, are also initialized with correct IP address, netmask and user/password settings.

Restarting The Network Interfaces

At the end of this step (i.e. section 6.4.3) the network interfaces are up. When the node-installer has completed the remainder of its 13 steps (sections 6.4.4–6.4.13), control is handed over to the local `init` process running on the local drive. During this handover, the node-installer brings down all network devices. These are then brought back up again by `init` by the distribution's standard networking `init` scripts, which run from the local drive and expect networking devices to be down to begin with.

6.4.4 Determining Install-mode Type And Execution Mode

Stored *install-mode* values decide whether synchronization is to be applied fully to the local drive of the node, only for some parts of its filesystem, not at all, or even whether to drop into a maintenance mode instead.

Related to install-mode values are execution mode values that determine whether to apply the install-mode values to the next boot, to new nodes only, to individual nodes or to a category of nodes.

Related to execution mode values is the confirmation requirement toggle value in case of a full installation is to take place.

These values are merely determined at this stage; nothing is executed yet.

Install-mode Values

The install-mode can have one of four values: `AUTO`, `FULL`, `MAIN` and `NOSYNC`.

- If the install-mode is set to `FULL`, the node-installer re-partitions, creates new file systems and synchronizes a full image onto the local drive. This process wipes out all pre-boot drive content.
- If the install-mode is set to `AUTO`, the node-installer checks the partition table and file systems of the local drive against the node's stored configuration. If these do not match because, for example, the node is new, or if they are corrupted, then the node-installer recreates the partitions and file systems by carrying out a `FULL` install. If however the drive partitions and file systems are healthy, the node-installer only does an incremental software image synchronization. Synchronization tends to be quick because the software image and the local drive usually do not differ much.

  Synchronization also removes any extra local files that do not exist on the image, for the files and directories considered. Section 6.4.7 gives details on how it is decided what files and directories are considered.
- If the install-mode is set to `MAIN`, the node-installer halts in maintenance mode, allowing manual investigation of specific problems. The local drive is untouched.
- If the install-mode is set to `NOSYNC`, and the partition and filesystem check matches the stored XML configuration, then the node-installer skips synchronizing the image to the node, so that contents on the local drive persist from the previous boot. An exception to this is the node certificate and key, that is the files `/cm/local/apps/cmd/etc/cert.{pem|key}`. These are updated from the head node if missing.

  If however the partition or filesystem does not match the stored configuration, a `FULL` image sync is triggered. Thus, for example, a burn session (Appendix N), with the default burn configuration which destroys the existing partition and filesystem on a node, will trigger a `FULL` image sync on reboot after the burn session.

Install-mode Logging

The decision that is made is normally logged to the node-installer file, `/var/log/node-installer` on the head node.

Example

```
08:40:58 node001 node-installer: Installmode is: AUTO
08:40:58 node001 node-installer: Fetching disks setup.
08:40:58 node001 node-installer: Setting up environment for initialize \
scripts.
08:40:58 node001 node-installer: Initialize script for category default\
 is empty.
08:40:59 node001 node-installer: Checking partitions and filesystems.
08:40:59 node001 node-installer: Updating device status: checking disks
08:40:59 node001 node-installer: Detecting device '/dev/sda': found
```

```
08:41:00 node001 node-installer: Number of partitions on sda is ok.
08:41:00 node001 node-installer: Size for /dev/sda1 is ok.
08:41:00 node001 node-installer: Checking if /dev/sda1 contains ext3 fi\
lesystem.
08:41:01 node001 node-installer: fsck.ext3 -a /dev/sda1
08:41:01 node001 node-installer: /dev/sda1: recovering journal
08:41:02 node001 node-installer: /dev/sda1: clean, 129522/1250928 files\
, 886932/5000000 blocks
08:41:02 node001 node-installer: Filesystem check on /dev/sda1 is ok.
08:41:02 node001 node-installer: Size for /dev/sda2 is wrong.
08:41:02 node001 node-installer: Partitions and/or filesystems are miss\
ing/corrupt. (Exit code 18, signal 0)
08:41:03 node001 node-installer: Creating new disk layout.
```

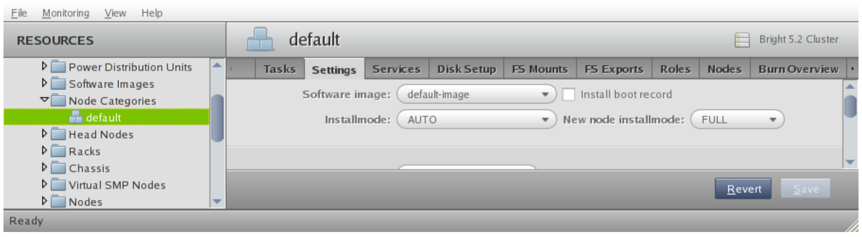
In this case the node-installer detects that the size of `/dev/sda2` on the disk no longer matches the stored configuration, and triggers a full re-install. For further detail beyond that given by the node-installer log, the `disks` script at `/cm/node-installer/scripts/disks` on the head node can be examined. The node-installer checks the disk by calling the `disks` script. Exit codes, such as the `18` reported in the log example, are defined near the top of the `disks` script.

Install-mode's Execution Modes

Execution of an install-mode setting is possible in several ways, both permanently or just temporarily for the next boot. Execution can be set to apply to categories or individual nodes. The node-installer looks for install-mode execution settings in this order:

1. The "`New node installmode`" property of the node's category. This decides the install mode for a node that is detected to be new.

   It can be set using `cmgui` (figure 6.16):



Software image: default-image
Install boot record
Installmode: AUTO
New node installmode: FULL

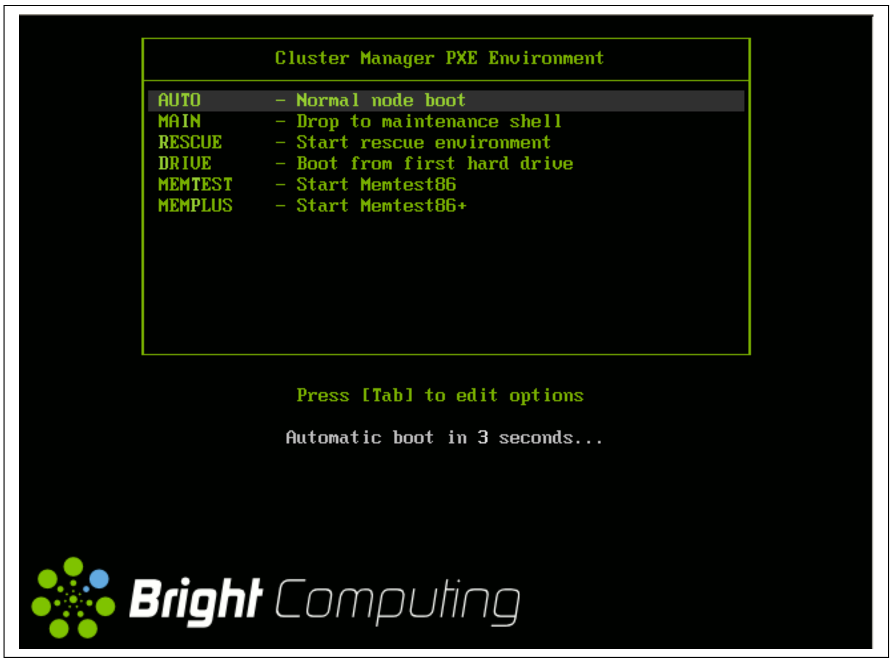
Figure 6.16: `cmgui` Install-mode Settings Under Node Category

   or using `cmsh` with a one-liner like:

```
cmsh -c "category use default; set newnodeinstallmode FULL; commit"
```

   By default, the "`New node installmode`" property is set to `FULL`.

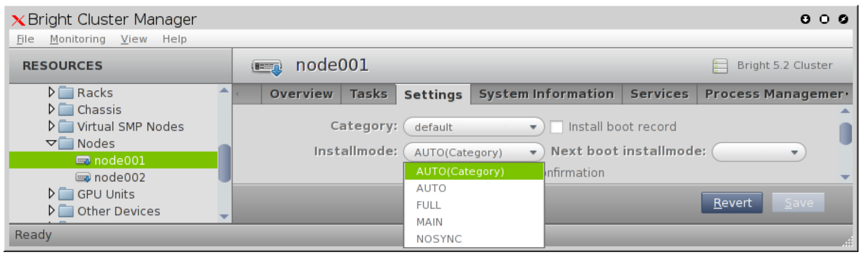
2. The Install-mode setting as set by choosing a PXE menu option on the console of the node before it loads the kernel and ramdisk (figure 6.17). This only affects the current boot. By default the PXE menu install mode option is set to `AUTO`.



```
Cluster Manager PXE Environment

AUTO      - Normal node boot
MAIN      - Drop to maintenance shell
RESCUE    - Start rescue environment
DRIVE     - Boot from first hard drive
MEMTEST   - Start Memtest86
MEMPLUS   - Start Memtest86+

Press [Tab] to edit options

Automatic boot in 3 seconds...
```

Figure 6.17: PXE Menu With Install-mode Set To `AUTO`

3. The "`Next boot install-mode`" property of the node configuration. This can be set using `cmgui` (figure 6.18):



Bright Cluster Manager

node001

Category: default

Install boot record

Installmode: AUTO(Category)

Next boot installmode:

Figure 6.18: `cmgui` Install-mode Settings For The Node

It can also be set using `cmsh` with a one-liner like:

```
cmsh -c "device use node001; set nextinstallmode FULL; commit"
```

The property is cleared when the node starts up again, after the node-installer finishes its installation tasks. So it is empty unless specifically set by the administrator during the current uptime for the node.

4. The `install-mode` property can be set in the node configuration using `cmgui` (figure 6.18), or using `cmsh` with a one-liner like:

```
cmsh -c "device use node001; set installmode FULL; commit"
```

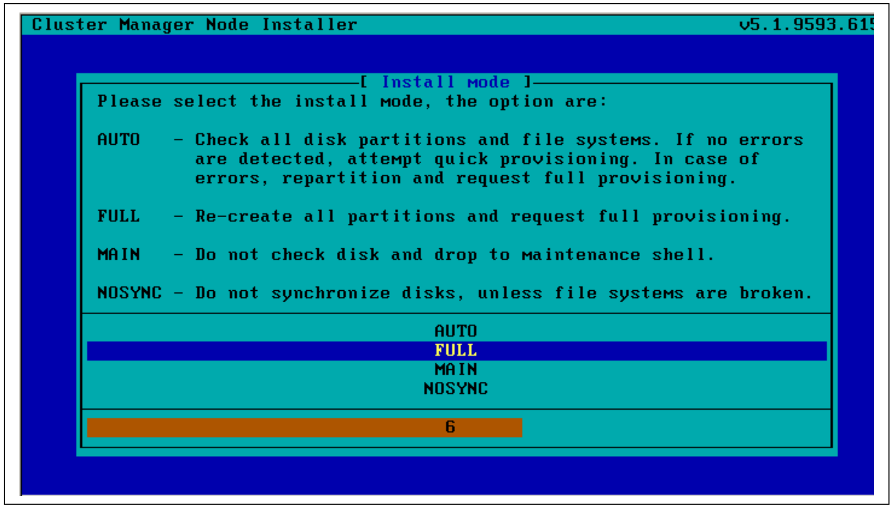
By default, the install-mode property is auto-linked to the property set for `install-mode` for that category of node. Since the property for that node's category defaults to `AUTO`, the property for the `install-mode` of the node configuration defaults to "`AUTO (Category)`".

5. The `install-mode` property of the node's category. This can be set using `cmgui` (figure 6.16), or using `cmsh` with a one-liner like:

   ```
   cmsh -c "category use default; set installmode FULL; commit"
   ```

   As already mentioned in a previous point, the install-mode is set by default to `AUTO`.

6. A dialog on the console of the node (figure 6.19) gives the user a last opportunity to overrule the install-mode value as determined by the node-installer. By default, it is set to `AUTO`:



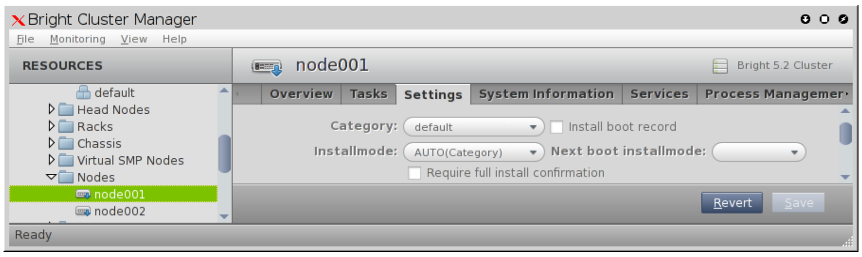
```
 Cluster Manager Node Installer                                    v5.1.9593.61

                          [ Install mode ]
   Please select the install mode, the option are:

   AUTO   - Check all disk partitions and file systems. If no errors
            are detected, attempt quick provisioning. In case of
            errors, repartition and request full provisioning.

   FULL   - Re-create all partitions and request full provisioning.

   MAIN   - Do not check disk and drop to maintenance shell.

   NOSYNC - Do not synchronize disks, unless file systems are broken.

                               AUTO
                               FULL
                               MAIN
                              NOSYNC

                                 6
```

Figure 6.19: Install-mode Setting Option During Node-Installer Run

Require Full Install Confirmation Toggle

Related to execution mode values is the "`Require full install confirmation`" value. This must be set in order to ask for a confirmation in case a FULL installation is about to take place. If set, the node-installer waits for a confirmation before going ahead with the FULL install.

The property can be set in the node category or node configuration with `cmgui` (figure 6.20):



Category: default
☐ Install boot record
Installmode: AUTO(Category)
Next boot installmode:
☐ Require full install confirmation

Figure 6.20: `cmgui` Require Full Install Confirmation Checkbox

Alternatively, it can be set using a `cmsh` one-liner like:

```
cmsh -c "device use node001; set requirefullinstallconfirmation yes;\
 commit"
```

The reason behind having such a setting available is that a FULL installation can be triggered by a change in disk/partition, or a change in the MAC address. If that happens, then:

- considering a drive, say, `/dev/sda` that fails, this means that any drive `/dev/sdb` would then normally become `/dev/sda` upon reboot. In that case an unwanted FULL install would not only be triggered by an install-mode settings of FULL, but also by the install-mode settings of AUTO or NOSYNC. Having the new, "accidental" `/dev/sda` have a FULL install is unlikely to be the intention, since it would probably contain useful data that the node-installer earlier left untouched.

- considering a node with a new MAC address, but with local storage containing useful data from earlier. In this case, too, an unwanted FULL install would not only be triggered by an install-mode setting of FULL, but also by the install-mode settings AUTO or NOSYNC.

Thus, in these two cases, having a confirmation required before proceeding with overwriting local storage contents is a good idea.

By default, no confirmation is required when a FULL installation is about to take place.

6.4.5 Running Initialize Scripts

An *initialize script* is used when custom commands need to be executed before checking partitions and mounting devices. For example, to initialize some not explicitly supported hardware, or to do a RAID configuration lookup for a particular node. In such cases the custom commands are added to an `initialize` script. How to edit an `initialize` script is described in Appendix E.2.

An `initialize` script can be added to both a node's category and the node configuration. The node-installer first runs an `initialize` script, if it exists, from the node's category, and then an `initialize` script, if it exists, from the node's configuration.

The node-installer sets several environment variables which can be used by the `initialize` script. Appendix E contains an example script documenting these variables.

Related to the `initialize` script is the `finalize` script (section 6.4.11). This may run after node provisioning is done, but just before the `init` process on the node run.

6.4.6 Checking Partitions, Mounting File Systems

In section 6.4.4 the node-installer determines the install-mode value, along with when to apply it to a node. The install-mode value defaults mostly to `AUTO`. If `AUTO` applies to the current node, it means the node-installer then checks the partitions of the local drive and its file systems and recreates them in case of errors. Partitions are checked by comparing the partition layout of the local drive(s) against the drive layout as configured in the node's category configuration and the node configuration.

After the node-installer checks the drive(s) and, if required, recreates the layout, it mounts all file systems to allow the drive contents to be synchronized with the contents of the software image.

If install-mode values of FULL or MAIN apply to the current node instead, then no partition checking or filesystem checking is done by the node-installer.

If the install-mode value of NOSYNC applies, then if the partition and filesystem checks both show no errors, the node starts up without getting an image synced to it from the provisioning node. If the partition or the filesystem check show errors, then the node partition is rewritten, and a known good image is synced across.

The node-installer is capable of creating advanced drive layouts, including software RAID and LVM setups. Drive layout examples and relevant documentation are in appendix D.

6.4.7 Synchronizing The Local Drive With The Software Image

After having mounted the local filesystems, these can be synchronized with the contents of the software image associated with the node (through its category). Synchronization is skipped if NOSYNC is set, and takes place if install-mode values of FULL or AUTO are set. Synchronization is delegated by the node-installer to the CMDaemon provisioning system. The node-installer just sends a provisioning request to CMDaemon on the head node.

For an install-mode of FULL, or for an install-mode of AUTO where the local filesystem is detected as being corrupted, full provisioning is done. For an install-mode of AUTO where the local filesystem is healthy and agrees with that of the software image, sync provisioning is done.

The software image that is requested is available to nodes by default. It can however be set to a locked state, which means that the request is held in the queue until it is no longer in a locked state. This is sometimes useful for making changes to an image when nodes are booting.

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% softwareimage use default-image
[ddnmon61->softwareimage[default-image]]% set locked yes
[ddnmon61->softwareimage*[default-image*]]% commit
```

For an unlocked image, on receiving the provisioning request, CMDaemon assigns the provisioning task to one of the provisioning nodes. The node-installer is notified when image synchronization starts, and also when the image synchronization task ends—whether it is completed successfully or not.

Exclude Lists: `excludelistsyncinstall` **And** `excludelistfullinstall`

What files are synchronized is decided by an *exclude list*. An exclude list is a property of the node category, and is a list of directories and files that are excluded from consideration during synchronization. The excluded list that is used is decided by the type of synchronization chosen: full or sync:

- A full type of synchronization rewrites the partition table of the node, then copies the filesystem from a software image to the node, using a list to specify files and directories to exclude from consideration when copying over the file system. The list of exclusions used is specified by the excludelistfullinstall property.

The intention of `full` synchronization is to allow a complete working filesystem to be copied over from a known good software image to the node. By default the `excludelistfullinstall` list contains `/proc/`, `/sys/`, and `lost+found/`, which have no content in DirectMon's default software image. The list can be modified to suit the requirements of a cluster, but it is recommended to have the list adhere to the principle of allowing a complete working node filesystem to be copied over from a known good software image.

- A `sync` type of synchronization uses the property `excludelistsyncinstall` to specify what files and directories to exclude from consideration when copying parts of the filesystem from a known good software image to the node. The `excludelistsyncinstall` property is in the form of a list of exclusions, or more accurately in the form of two sub-lists.

  The contents of the sub-lists specify the parts of the filesystem that should be retained or not copied over from the software image during `sync` synchronization when the node is booting. The intention behind this is to have the node boot up quickly, updating only the files from the image to the node that need updating due to the reboot of the node, and otherwise keeping files that are already on the node hard disk unchanged. The contents of the sub-lists are thus items such as the node log files, or items such as the `/proc` and `/sys` pseudo-filesystems which are generated during node boot.

  The administrator should be aware that nothing on a node hard drive can be regarded as persistent because a FULL sync takes place if any error is noticed during a partition or filesystem check.

  Anything already on the node that matches the content of these sub-lists is not overwritten by image content during an `excludelistsyncinstall` sync. However, image content that is not on the node is copied over to the node only for items matching the first sub-list. The remaining files and directories on the node, that is, the ones that are not in the sub-lists, lose their original contents, and are copied over from the software image.

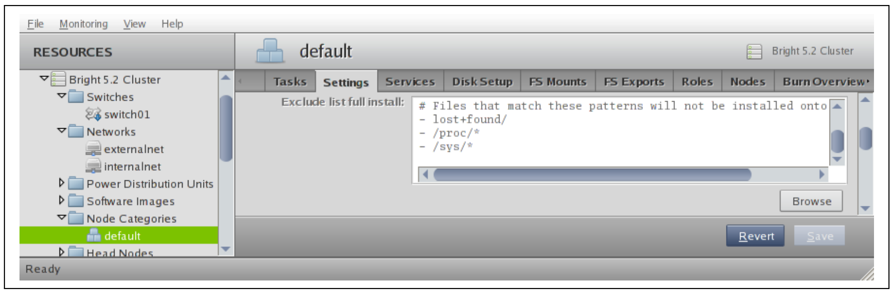
A `cmsh` one-liner to get an exclude list for a category is:

```
cmsh -c "category use default; get excludelistfullinstall"
```

Similarly, to set the list:

```
cmsh -c "category use default; set excludelistfullinstall; commit"
```

where a text-editor opens up to allow changes to be made to the list. Figure 6.21 illustrates how the setting can be modified via `cmgui`.



default

Exclude list full install:

```
# Files that match these patterns will not be installed onto
- lost+found/
- /proc/*
- /sys/*
```

Figure 6.21: Setting up exclude lists with `cmgui` for provisioning

Image synchronization is done using `rsync`, and the syntax of the items in the exclude lists conforms to the "`INCLUDE/EXCLUDE PATTERN RULES`" section of the `rsync(1)` man page, which includes patterns such as "`**`", "`?`", and "`[[:alpha:]]`".

The `excludelistfullinstall` and `excludelistsyncinstall` properties decide how a node synchronizes to an image during boot. For a node that is already fully up, the related `excludelistupdate` property decides how a running node synchronizes to an image without a reboot event, and is discussed in section 6.6.

Interface Used To Receive Image Data: `provisioninginterface`
For regular nodes with multiple interfaces, one interface may be faster than the others. If so, it can be convenient to receive the image data via the fastest interface. Setting the value of `provisioninginterface`, which is a property of the node configuration, allows this.

By default it is set to `BOOTIF` for regular nodes. Using `BOOTIF` is not recommended for node configurations with multiple interfaces.

When listing the network interfaces in `cmsh`, the provisioning interface has a `[prov]` flag appended to its name.

Example

```
[ddnmon61->device[node001]->interfaces]% list
Type         Network device name  IP               Network
------------ -------------------- ---------------- ----------------
physical     BOOTIF [prov]        10.141.0.1       internalnet
physical     eth1                 10.141.1.1       internalnet
physical     eth2                 10.141.2.1       internalnet
```

Head nodes and `provisioninginterface`: A head node in a single-head cluster does not use the `provisioninginterface` setting.

Head nodes in a failover configuration (Chapter 15), however, do have a value set for `provisioninginterface`, corresponding to the interface on the head that is being provisioned over `internalnet` by the other head (`eth0` in figure 15.1).

Transport Protocol Used For Image Data: `provisioningtransport`
The `provisioningtransport` property of the node sets whether the image data is sent encrypted or unencrypted to the node from the provisioner. The property value is set via the `device` mode for the receiving node to one of these values:

- `rsyncdaemon`, which sends the data unencrypted

- rsyncssh, which sends the data encrypted

The provisioningtransport value can be set for all nodes, including provisioning nodes, head nodes, and cloud-director (section 7.2.2) nodes. Because encryption severely increases the load on the provisioning node, using rsyncssh is only suggested if the users on the network cannot be trusted. By default, provisioningtransport is set to rsyncdaemon. If high availability (chapter 15) is set up with the head nodes exposed to the outside world on the external network, the administrator should consider setting up rsyncssh for the head nodes.

The rsyncssh transport requires passwordless root access via ssh from the provisioner to the node being provisioned. This is configured by default in the default DirectMon nodes. However, if a new image is created with the --exclude options for cm-create-image as explained in (section 10.6.2), the keys must be copied over from /root/.ssh/ on the existing nodes.

Tracking The Status Of Image Data Provisioning: provisioningstatus
The provisioningstatus command within the softwareimage mode of cmsh displays an updated state of the provisioning system. As a one-liner, it can be run as:

```
ddnmon61:~ # cmsh -c "softwareimage provisioningstatus"
Provisioning subsystem status:          idle, accepting requests
Update of provisioning nodes requested: no
Maximum number of nodes provisioning:   10000
Nodes currently provisioning:           0
Nodes waiting to be provisioned:        <none>
Provisioning node ddnmon61:
  Max number of provisioning nodes:     10
  Nodes provisioning:                   0
  Nodes currently being provisioned:    <none>
```

The provisioningstatus command has several options that allow the requests to be tracked. The -r option displays the basic status information on provisioning requests, while the -a option displays all status information on provisioning requests. Both of these options display the request IDs.

The cmgui equivalent to provisioningstatus is accessed from the "Provisioning Status" tabbed pane in the "Software Images" resource (figure 6.5). By default, it displays basic status information on provisioning requests, while the "Request Details" button allows details to be seen of provisioning requests.

Tracking The Provisioning Log Changes: synclog
For a closer look into the image file changes carried out during provisioning requests, the synclog command from device mode can be used (lines elided in the following output):

Example

```
[ddnmon61->device]% synclog node001
Tue, 11 Jan 2011 13:27:17 CET - Starting rsync daemon based provisionin\
g. Mode is SYNC.
```

```
sending incremental file list
./
...
deleting var/lib/ntp/etc/localtime
var/lib/ntp/var/run/ntp/
...
sent 2258383 bytes  received 6989 bytes  156232.55 bytes/sec
total size is 1797091769  speedup is 793.29

Tue, 11 Jan 2011 13:27:31 CET - Rsync completed.
```

In `cmgui`, the equivalent output to `cmsh`'s `synclog` is displayed by selecting a specific device or a specific category from the resource tree. Then, within the tasks tabbed pane that opens up, the "`Provisioning Log`" button at the bottom right is clicked (figure 6.22):

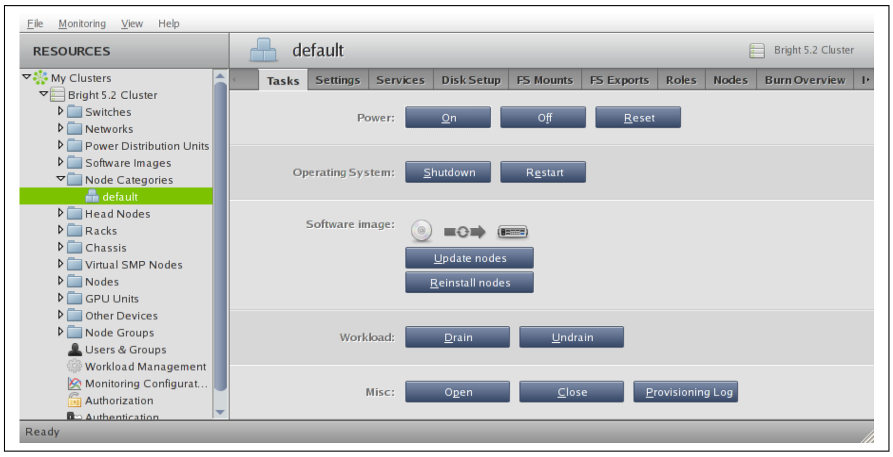


Figure 6.22: `cmgui`: Provisioning Log Button For A Device Resource

Aborting Provisioning With `cancelprovisioningrequest`

The `cancelprovisioningrequest` command cancels provisioning.

Its usage is:

```
cancelprovisioningrequest [OPTIONS] [<requestid> ...]
```

To cancel all provisioning requests, it can be run as:

```
ddnmon61:~ # cmsh -c "softwareimage cancelprovisioningrequest -a"
```

The `provisioningstatus` command of `cmsh`, or the "`Provisioning Status`" tab of `cmgui` can be used to find request IDs. Individual request IDs, for example `10` and `13`, can then be specified in the `cancelprovisioningrequest` command, as:

```
ddnmon61:~ # cmsh -c "softwareimage cancelprovisioningrequest 10 13"
```

The help page for `cancelprovisioningrequest` shows how to run the command on node ranges, groups, categories, racks, and chassis.

6.4.8 Writing Network Configuration Files

In the previous section, the local drive of the node is synchronized according to install-mode settings with the software image from the provisioning node. The node-installer now sets up configuration files for each configured network interface. These are files like:

```
/etc/sysconfig/network-scripts/ifcfg-eth0
```

for Red Hat, Scientific Linux, and CentOS, while SUSE would use:

```
/etc/sysconfig/network/ifcfg-eth0
```

These files are placed on the local drive.

When the node-installer finishes its remaining tasks (sections 6.4.9–6.4.13) it brings down all network devices and hands over control to the local `/sbin/init` process. Eventually a local `init` script uses the network configuration files to bring the interfaces back up.

6.4.9 Creating A Local `/etc/fstab` File

The `/etc/fstab` file on the local drive contains local partitions on which filesystems are mounted as the `init` process runs. The actual drive layout is configured in the category configuration or the node configuration, so the node-installer is able to generate and place a valid local `/etc/fstab` file. In addition to all the mount points defined in the drive layout, several extra mount points can be added. These extra mount points, such as NFS imports, `/proc`, `/sys` and `/dev/shm`, can be defined and managed in the node's category and in the specific configuration of the node configuration, using `cmgui` or `cmsh` (section 4.11.2).

6.4.10 Installing GRUB Bootloader

By default, a node-installer boots from the software image on the head node via the network.

Optionally, the node-installer installs a boot record on the local drive if the `installbootrecord` property of the node configuration or node category is set to `on`, so that the next boot can be from the local drive.

For a hard drive boot to work:

1. hard drive booting must be set to have a higher priority than network booting in the BIOS of the node. Otherwise regular PXE booting occurs, despite whatever value `installbootrecord` has.

2. the GRUB bootloader with a boot record must be installed in the MBR of the local drive, overwriting the default iPXE boot record.

   If the administrator is not using the default software image, but is using a custom software image (section 10.6.1), and if the image is based on a running node filessystem that has not been built directly from a parent distribution, then the GRUB boot configuration may not be appropriate for a standalone GRUB boot to work. This is because the parent distribution installers often use special logic for setting up the GRUB boot configuration. Carrying out this same special logic for all distributions using the custom software image creation tool `cm-create-image` (section 10.6.2) is impractical.

   Providing a custom working image from a standalone node that has been customized after direct installation from the parent distribution, ensures the GRUB boot configuration layout of the custom image is as expected by the parent distribution. This then allows a standalone GRUB boot on the node to run properly.

With a working custom software image, to set the GRUB bootloader in `cmgui`, the "Install boot record" checkbox must be ticked and saved in the node configuration or in the node category.

The `cmsh` equivalents are commands like:

```
cmsh -c "device use node001; set installbootrecord yes; commit"
```

or

```
cmsh -c "category use default; set installbootrecord yes; commit"
```

Arranging for the two items in the preceding list ensures that the next boot is from GRUB on the hard drive.

Simply unsetting "Install boot record" and rebooting the node does not restore its iPXE boot record and hence its ability to iPXE boot. To restore the iPXE boot record, the node can be booted from the default image copy on the head node via a network boot again. Typically this is done by manual intervention during node boot to select network booting from the BIOS of the node.

As suggested by the DirectMon iPXE boot prompt, setting network booting to work from the BIOS (regular "PXE" booting) is preferred to iPXE booting from the disk.

If configured, SELinux (Appendix P) is initialized at this point. For a boot from the hard drive, the initialization occurs if an SELinux filesystem has been saved to disk previously. For a PXE boot, the initialization takes place if the `SELinuxInitialize` directive is set to `true` in the `node-installer.conf` file.

6.4.11 Running Finalize Scripts

A *finalize script* is similar to an `initialize` script (section 6.4.5), only it runs a few stages later in the node-provisioning process.

It is used when custom commands need to be executed after the preceding mounting, provisioning, and housekeeping steps, but before handing over control to the node's local `init` process. For example, custom commands may be needed to:

- initialize some not explicitly supported hardware before `init` takes over
- supply a configuration file for the software image that cannot simply be added to the software image and used by `init` because it needs node-specific settings
- load a slightly altered standard software image on particular nodes, typically with the change depending on automatically detecting the hardware of the node it is being loaded onto. While this could also be done by creating a full new software image and loading it on to the nodes according to the hardware, it usually turns out to be better for simplicity's sake (future maintainability) to minimize the number of software images for the cluster

The custom commands used to implement such changes are then added to the `finalize` script. How to edit a `finalize` script is described in Appendix E.2.

A `finalize` script can be added to both a node's category and the node configuration. The node-installer first runs a `finalize` script, if it exists, from the node's category, and then a `finalize` script, if it exists, from the node's configuration.

The node-installer sets several environment variables which can be used by the `finalize` script. Appendix E contains an example script which documents these variables.

6.4.12 Unloading Specific Drivers

Many kernel drivers are only required during the installation of the node. After installation they are not needed and can degrade node performance.

Baseboard Management Controllers (BMCs, section 4.8) that use IPMI drivers are an egregious example of this. The IPMI drivers are required to have the node-installer configure the IP address of any IPMI cards. Once the node is configured, these drivers are no longer needed, but they continue to consume significant CPU cycles and power if they stay loaded, which can affect job performance.

To solve this, the node-installer can be configured to unload a specified set of drivers just before it hands over control to the local init process. This is done by editing the `removeModulesBeforeInit` setting in the node-installer configuration file `/cm/node-installer/scripts/node-installer.conf`. By default, the IPMI drivers are placed in the `removeModulesBeforeInit` setting.

To pick up IPMI-related data values, IPMI access is then carried out over the network without the drivers.

6.4.13 Switching To The Local `init` Process

At this point the node-installer is done. The node's local drive now contains a complete Linux installation and is ready to be started. The node-installer hands over control to the local `/sbin/init` process, which continues the boot process and starts all runlevel services. From here on the boot process continues as if the machine was started from the drive just like any other regular Linux machine.

6.5 Node States

During the boot process, several state change messages are sent to the head node CMDaemon or detected by polling from the head node CMDaemon. The most important node states for a cluster after boot up are introduced in section 3.1.1. These states are described again, along with some less common ones to give a more complete picture of node states.

6.5.1 Node States Icons In `cmgui`

In the node icons used by `cmgui` (figure 6.23):

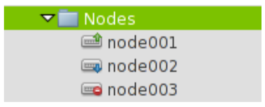


Figure 6.23: Icons Indicating Node States In `cmgui`

- Nodes in the `UP` state are indicated by an up-arrow.
  - If all health checks (section 11.2.4) for the node are successful, the up-arrow is green.
  - If there is a health check that fails or if the node requires a reboot, the up-arrow is red.
- Nodes in the `DOWN` state are indicated by a blue down-arrow.
- Nodes in all other states are indicated by a white horizontal dash inside a solid red circle.

6.5.2 Node States Shown In `cmsh`

In `cmsh`, the node state can be found using the `status` command from `device` mode for a node:

Example

```
[ddnmon61->device]% status -n node001..node002
node001 ............. [   UP   ] restart-required, health check failed
node002 ............. [  DOWN  ] (hostname changed) restart-required
```

Devices in general can have their states conveniently listed with the `list -f` (page 65) command:

Example

```
[ddnmon61->device]% list -f "hostname:10, status:48"
hostname ( status
---------- ------------------------------------------------
apc01      [   UP   ]
bright61   [   UP   ]
devhp      [   UP   ]
node001    [   UP   ] restart-required, health check failed
node002    [  DOWN  ] (hostname changed) restart-required
```

The reason for a red icon as shown in section 6.5.1 can be found within the parentheses. In this example it is `(hostname changed)`.

6.5.3 Node States Indicating Regular Start Up

During a successful boot process the node goes through the following states:

- `INSTALLING`. This state is normally entered as soon as the node-installer has determined on which node the node-installer is running. Within this state, information messages display indicating what is being done while the node is in the `INSTALLING` state. Possible messages under the status column for the node within `cmgui` and `cmsh` are normally, in sequence:
  1. `node-installer started`

2. Optionally, the following two messages:
   (a) `waiting for user input`
   (b) `installation was resumed`
3. `checking disks`
4. `recreating partitions and file systems`
5. `mounting disks`
6. One of these following two messages:
   (a) `waiting for FULL provisioning to start`
   (b) `waiting for SYNC provisioning to start`
7. `provisioning started, waiting for completion`
8. `provisioning complete`
9. `initializing SELinux`

Between steps 1 and 3 in the preceding, these optional messages can also show up:

– If `burn` mode is entered or left:

```
running burn-in tests
burn-in test completed successfully
```

– If `maintenance` mode is entered:

```
entered maintenance mode
```

• `INSTALLER_CALLINGINIT`. This state is entered as soon as the node-installer has handed over control to the local `init` process. The associated message normally seen with it in `cmsh` or `cmgui` is:

  – `switching to local root`

• `UP`. This state is entered as soon as the CMDaemon of the node connects to the head node CMDaemon.

6.5.4 Node States That May Indicate Problems

Other node states are often associated with problems in the boot process:

• `DOWN`. This state is registered as soon as the CMDaemon on the regular node is no longer detected by CMDaemon on the head node. In this state, the state of the regular node is still tracked, so that CMDaemon is aware if the node state changes.

• `CLOSED`. This state is appended to the `UP` or `DOWN` state of the regular node by the administrator, and causes most CMDaemon monitoring actions for the node to cease. The state of the node is however still tracked by default, so that CMDaemon is aware if the node state changes.

  The `CLOSED` state can be set from the `device` mode of `cmsh` using the `close` command. The `help` text for the command gives details on how it can be applied to categories, groups and so on. The `-m` option sets a message by the administrator for the closed node or nodes.

Example

```
root@b52 ~]# cmsh
[b52]% device
[b52->device]% close -m "fan dead" -n node001,node009,node020
Mon May 2 16:32:01 [notice] b52: node001 ...[ DOWN/CLOSED ] (fan dead)
Mon May 2 16:32:01 [notice] b52: node009 ...[ DOWN/CLOSED ] (fan dead)
Mon May 2 16:32:01 [notice] b52: node020 ...[ DOWN/CLOSED ] (fan dead)
```

The `--noping` option for the `close` command of `cmsh` causes the state ping tracking to cease. The additional text `state ping disabled` is then used alongside the `CLOSED` state message to indicate when the regular node state is no longer tracked by CMDaemon on the head node.

The `CLOSED` state can also be set from `cmgui`. This is done via the `Tasks` tab of the node item in the `Nodes` resource, or from the category item in the `Node Categories` resource (figure 6.24).

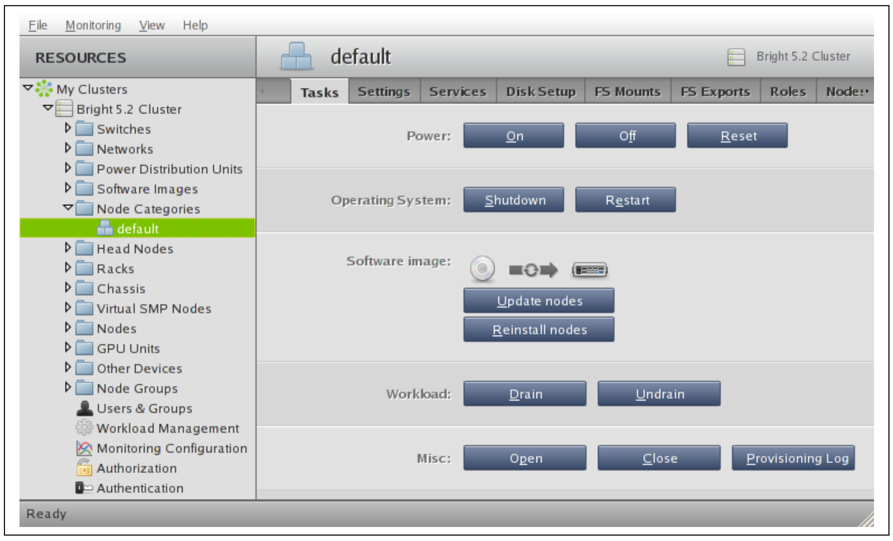


Figure 6.24: Acting On A CLOSED State From `cmgui`

When the `CLOSED` state is set for a device, CMDaemon commands can still attempt to act upon it. For example, in the `device` mode of `cmsh`:

- `open`: This is the converse to the `close` command. It has the same options, including the `-m` option that logs a message
- `drain` and `undrain` (Appendix H.3.1)
- For nodes that have power control[2]:
  - `power -f on`
  - `power -f off`
  - `power -f reset`

The `cmgui` equivalents in figure 6.24 are:

[2] power control mechanisms such as PDUs, BMCs using IPMI/HP iLO, and custom power scripts are described in Chapter 5

- The `Open` button from the `Watch` row
- The `Drain` and `Undrain` buttons from the `Workload` row
- The `On`, `Off`, and `Reset` buttons from the PDU `Power` row

CMDaemon on the head node only maintains device monitoring logs for a device that is in the `UP` state. If the device is in a state other than `UP`, then CMDaemon only tracks its state, and can display the state if queried. Executing the `close` command with the option `--noping` in `cmsh` disables even state tracking for the device by CMDaemon. Executing the `close` command without the `--noping` option then enables status tracking again.

For example: if a node displays the state `UP` when queried about its state, and is given a 'close' command, it then goes into a `CLOSED` state. Querying the node state then displays the state `UP/CLOSED`. It remains in that `CLOSED` state when the node is powered down. Querying the node state after being powered down displays `DOWN/CLOSED`. Next, powering up the node from that state, and having it go through the boot process, has the node displaying various `CLOSED` states during queries. Typically the query responses show it transitioning from `DOWN/CLOSED`, to `INSTALLING/CLOSED`, to `INSTALLER_CALLINGINIT/CLOSED`, and ending up displaying the `UP/CLOSED` state.

Thus, a node set to a `CLOSED` state remains in a `CLOSED` state regardless of whether the node is in an `UP` or `DOWN` state. The only way out of a `CLOSED` state is for the administrator to tell the node to open via the `cmsh` or `cmgui` "open" options discussed earlier. The node, as far as CMDaemon is concerned, then switches from the `CLOSED` state to the `OPEN` state. Whether the node listens or not does not matter—the head node records it as being in an `OPENING` state for a short time, and during this time the next `OPEN` state (`UP/OPEN`, `DOWN/OPEN`, etc) is agreed upon by the head node and the node.

When querying the state of a node, an `OPEN` tag is not displayed in the response, because it is the "standard" state. For example, `UP` is displayed rather than `UP/OPEN`. In contrast, a `CLOSED` tag is displayed when it is active, because it is a "special" state.

The `CLOSED` state is normally set to take a node that is unhealthy out of the cluster management system. The node can then still be in the `UP` state, displaying `UP/CLOSED`. It can even continue running workload jobs in this state, since workload managers run independent of CMDaemon. So, if the workload manager is still running, the jobs themselves are still handled by the workload manager, even if CMDaemon is no longer aware of the node state until the node is re-opened. For this reason, draining a node is often done before closing a node, although it is not obligatory.

- `OPENING`. This transitional state is entered as soon as the CMDaemon of the node rescinds the `CLOSED` state with an "open" command from `cmsh` or `cmgui`. The state usually lasts no more than about 5 seconds, and never more than 30 seconds in the default configuration settings of DirectMon. The `help` text for the `open` command of `cmsh` gives details on its options.

- INSTALLER_FAILED. This state is entered from the INSTALLING state when the node-installer has detected an unrecoverable problem during the boot process. For instance, it cannot find the local drive, or a network interface cannot be started. This state can also be entered from the INSTALLER_CALLINGINIT state when the node takes too long to enter the UP state. This could indicate that handing over control to the local init process failed, or the local init process was not able to start the CMDaemon on the node. Lastly, this state can be entered when the previous state was INSTALLER_REBOOTING and the reboot takes too long.
- INSTALLER_UNREACHABLE. This state is entered from the INSTALLING state when the head node CMDaemon can no longer ping the node. It could indicate the node has crashed while running the node-installer.
- INSTALLER_REBOOTING. In some cases the node-installer has to reboot the node to load the correct kernel. Before rebooting it sets this state. If the subsequent reboot takes too long, the head node CMDaemon sets the state to INSTALLER_FAILED.

6.6 Updating Running Nodes

Changes made to the contents of the software image for nodes, kept on the head node, become a part of any other provisioning nodes according to the housekeeping system on the head node (section 6.2.4).

Thus, when a regular node reboots, the latest image is installed from the provisioning system onto the regular node via a provisioning request (section 6.4.7).

However, updating a running node with the latest software image changes is also possible without rebooting it. Such an update can be requested using cmsh or cmgui, and is queued and delegated to a provisioning node, just like a regular provisioning request. The properties that apply to the regular provisioning an image also apply to such an update. For example, the value of the provisioninginterface setting (section 6.4.7) on the node being updated determines which interface is used to receive the image. In cmsh the request is submitted with the imageupdate option, while in cmgui, it is submitted using the "Update node" button. The imageupdate command and "Update node" button use a configuration file called excludelistupdate, which is, as its name suggests, a list of exclusions to the update.

The imageupdate command and "Update node" button update what is on a running node from a stored image. The converse, that is, to update a stored image from what is on a running node, can be also be carried out. This converse can be viewed as grabbing from a node and synchronizing what is grabbed to an image. It can be done using grabimage (cmsh), or Synchronize (cmgui), and involves further exclude lists excludelistgrab or excludelistgrabnew. The grabimage command and Synchronize button are covered in detail in section 10.5.2.

6.6.1 Updating Running Nodes: Configuration With `excludelistupdate`

The exclude list `excludelistupdate` used by the `imageupdate` command is defined as a property of the node's category. It has the same structure and `rsync` patterns syntax as that used by the exclude lists for provisioning the nodes during installation (section 6.4.7).

Distinguishing Between The Intention Behind The Various Exclude Lists

The administrator should note that it is the `excludelist`*update* list that is being discussed here, in contrast with the `excludelistsync`*install*/`excludelistfull`*install* lists which are discussed in section 6.4.7, and also in contrast with the `excludelist`*grab*/`excludelist`*grabnew* lists of section 10.5.2.

So, for the `imageupdate` command the `excludelistupdate` list concerns an *update* to a running system, while for installation `sync` or `full` provisioning, the corresponding exclude lists (`excludelistsyncinstall` and `excludelistfullinstall`) from section 6.4.7 are about an *install* during node start-up. Because the copying intention during updates is to be speedy, the the `imageupdate` command synchronizes files rather than unnecessarily overwriting unchanged files. Thus, the `excludelistupdate` exclusion list it uses is actually analogous to the `excludelistsyncinstall` exclusion list used in the `sync` case of section 6.4.7, rather than being analogous to the `excludelistfullinstall` list.

Similarly, the `excludelistgrab/excludelistgrabnew` lists of section 10.5.2 are about a *grab* from the running node to the image. The `excludelistgrab` list is here intended for synchronizing the existing image with the running node, and is thus analogous to the `excludelistsyncinstall` exclusion list, while the `excludelistgrabnew` list is intended for copying a full new image from the running node, and is thus analogous to the `excludelistfullinstall` list.

The following table summarizes this:

During:	Exclude list used is:	Copy intention:
update	`excludelistupdate`	sync, image to running node
install	`excludelistfullinstall`	full, image to starting node
	`excludelistsyncinstall`	sync, image to starting node
grab	`excludelistgrabnew`	full, running node to image
	`excludelistgrab`	sync, running node to image

The Exclude List Logic For `excludelistupdate`

During an `imageupdate` command, the synchronization process uses the `excludelistupdate` list, which is a list of files and directories. One of the cross checking actions that may run during the synchronization is that the items on the list are excluded when copying parts of the filesystem from a known good software image to the node. The detailed behavior is as follows:

The `exludelistupdate` list is in the form of two sublists. Both sublists are lists of paths, except that the second sublist is prefixed with the text "`no-new-files: `" (without the double quotes). For the node being updated, all of its files are looked at during an `imageupdate` synchronization run. During such a run, the logic that is followed is:

- if an excluded path from `excludelistupdate` exists on the node, then nothing from that path is copied over from the software image to the node
- if an excluded path from `excludelistupdate` does not exist on the node, then
  - if the path is on the first, non-prefixed list, then the path is copied over from the software image to the node.
  - if the path is on the second, prefixed list, then the path is not copied over from the software image to the node. That is, no new files are copied over, like the prefix text implies.

This is illustrated by figure 6.25.

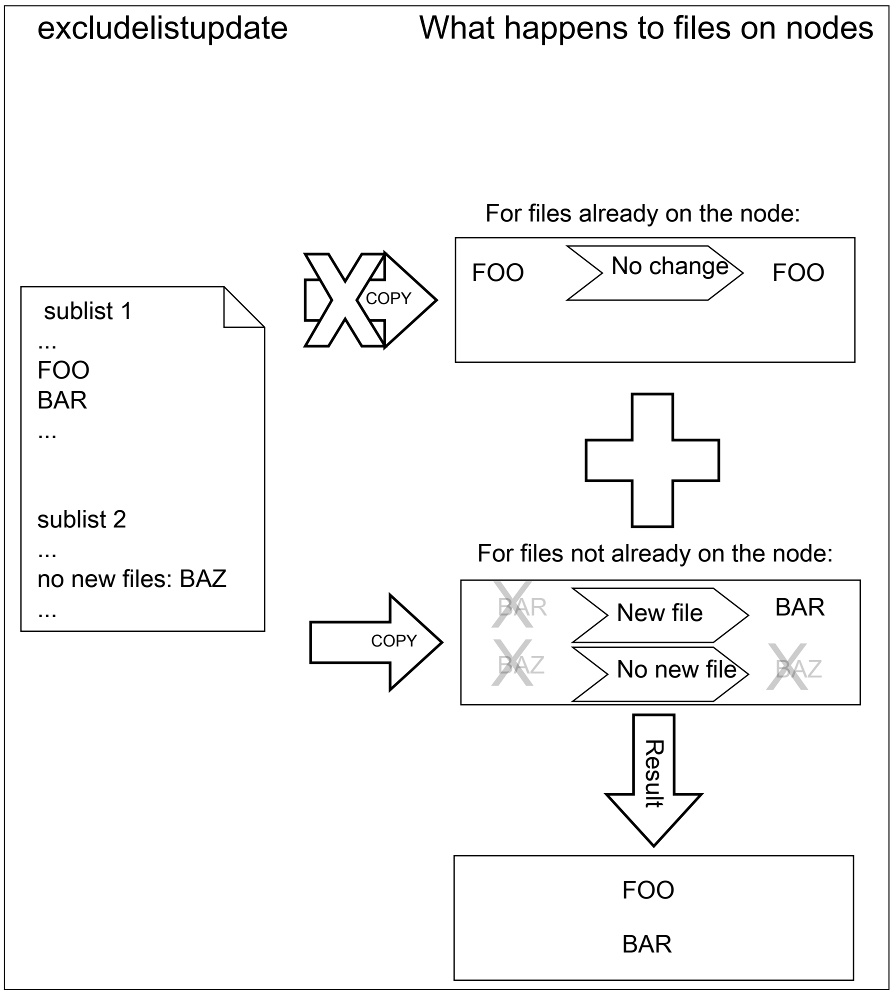


Figure 6.25: Exclude list logic

The files and directories on the node that are not in the sub-lists lose their original contents, and are copied over from the software image. So, content not covered by the sub-lists at that time is not normally protected from deletion.

In addition to the paths excluded using the `excludelistupdate` property, the provisioning system automatically adds any statically-imported NFS, Lustre, FUSE, CephFS, CIFS, PanFS, FhGFS, GlusterFS, and GPFS file systems that are on the node. If this were not done, all data on these filesystems would be wiped since they are not part of the software image. Statically-imported filesystems can also be excluded from being mounted on the nodes in the first place by removing them from the listed mounts within the `fsmounts` mode.

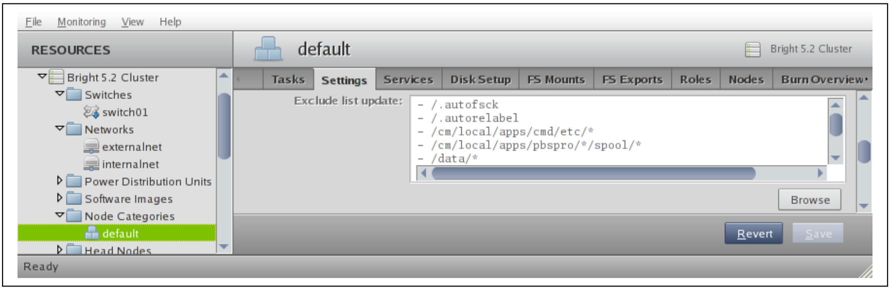
Filesystems mounted with an automounter cannot have their appearance or disappearance detected reliably. Any filesystem that may be imported via an automount operation must therefore explicitly be excluded by the administrator manually adding the filesystem to the exclude list. This is to prevent an incorrect execution of `imageupdate`. Neglecting to do this may wipe out the filesystem, if it happens to be mounted in the middle of an `imageupdate` operation.

Editing An Exclude List

A sample `cmsh` one-liner which opens up a text editor in a category so that the exclude list for updates can be edited is:

```
cmsh -c "category use default; set excludelistupdate; commit"
```

Similarly, the exclude list for updates can also be edited in `cmgui` as indicated in figure 6.26.



Exclude list update:

```
- /.autofsck
- /.autorelabel
- /cm/local/apps/cmd/etc/*
- /cm/local/apps/pbspro/*/spool/*
- /data/*
```

Figure 6.26: Setting up exclude lists with `cmgui` for node updates

6.6.2 Updating Running Nodes: With `cmsh` Using `imageupdate`

Using a defined `excludelistupdate` property (section 6.6.1), the `imageupdate` command of `cmsh` is used to start an update on a running node:

Example

```
[ddnmon61->device]% imageupdate -n node001
Performing dry run (use synclog command to review result, then pass -w \
to perform real update)...
Tue Jan 11 12:13:33 2011 ddnmon61: Provisioning started on node node001
```

```
[ddnmon61->device]% imageupdate -n node001: image update in progress ...
[ddnmon61->device]%
Tue Jan 11 12:13:44 2011 ddnmon61: Provisioning completed on node node0\
01
```

By default the `imageupdate` command performs a dry run, which means no data on the node is actually written. Before passing the "`-w`" switch, it is recommended to analyze the rsync output using the `synclog` command (section 6.4.7).

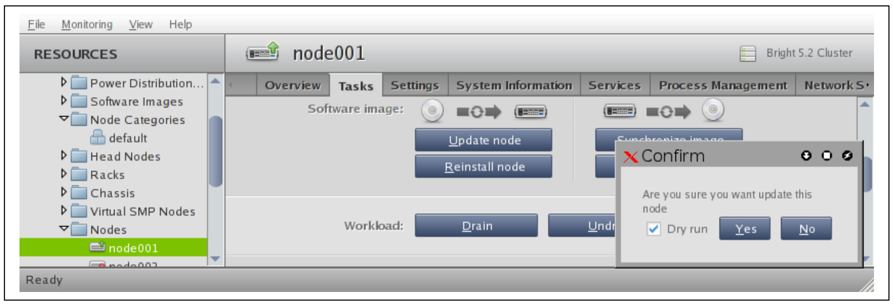
If the user is now satisfied with the changes that are to be made, the `imageupdate` command is invoked again with the "`-w`" switch to implement them:

Example

```
[ddnmon61->device]% imageupdate -n node001 -w
Provisioning started on node node001
node001: image update in progress ...
[ddnmon61->device]% Provisioning completed on node node001
```

6.6.3 Updating Running Nodes: With `cmgui` Using The "`Update node`" Button

In `cmgui` an image update can be carried out by selecting the specific node or specific category from the resource tree. Then, within the tasks tabbed pane that opens up, the "`Update node`" button is clicked (figure 6.27). This opens up a dialog which has a dry-run checkbox marked by default.



Are you sure you want update this node

✓ Dry run

Figure 6.27: Updating A Running Node With `cmgui`

The dry-run can be reviewed by clicking on the "`Provisioning Log`" button further down the same tabbed pane. The update can then be done again with the dry-run check mark off to actually implement the update.

6.6.4 Updating Running Nodes: Considerations

Updating an image via `cmsh` or `cmgui` automatically updates the provisioners first via the `updateprovisioners` command (section 6.2.4) if the provisioners have not been updated in the last 5 minutes. The conditional update period can be set with the `provisioningnodeautoupdatetimeout` parameter (section 6.2.4).

So, with the default setting of 5 minutes, if there has been an update within the last 5 minutes, then provisioners do not get an updated image when doing the updates. Running the `updateprovisioners` command just before running the imageupdate command therefore usually makes sense.

Also, when updating services, the services on the nodes may not restart since the init process may not notice the replacement.

For these reasons, especially for more extensive changes, it can be safer for the administrator to simply reboot the nodes instead of using imageupdate to provision the images to the nodes. A reboot by default ensures that a node places the latest image with an AUTO install (section 6.4.7), and restarts all services. The "Reinstall node" button (figure 6.27) also does the same as a reboot with default settings, except for that it unconditionally places the latest image with a FULL install, and so may take longer to complete.

6.7 Adding New Nodes

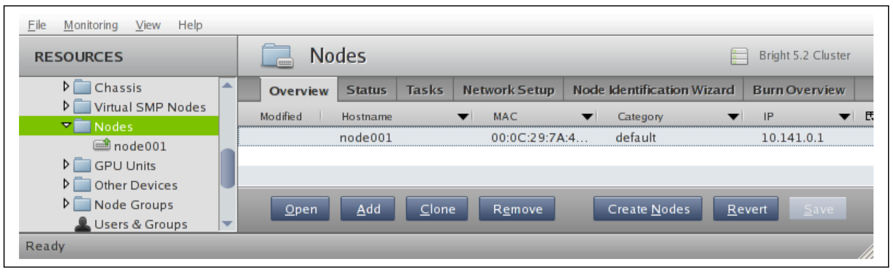
6.7.1 Adding New Nodes With cmsh And cmgui Add Functions

Node objects can be added from within the device mode of cmsh by running the add command:

Example

```
[ddnmon61->device]% add physicalnode node002
[ddnmon61->device*[node002*]% commit
```

The cmgui equivalent of this is to go within the Nodes resource, and after the Overview tabbed pane for the Nodes resource comes up, to click on the Add button (figure 6.28)



Modified	Hostname	MAC	Category	IP
	node001	00:0C:29:7A:4...	default	10.141.0.1

Figure 6.28: Buttons To Add, Remove And Set Up Nodes

When adding the node objects in cmsh and cmgui, some values (IP addresses for example) may need to be filled in before the object validates.

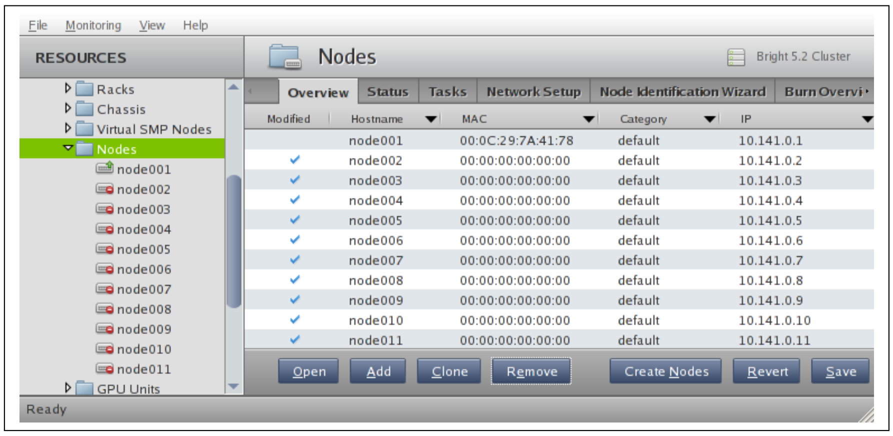
Adding new node objects as "placeholders" can also be done from cmsh or cmgui. By placeholders, here it is meant that an incomplete node object is set. For example, sometimes it is useful to create a node object with the MAC address setting unfilled because it is still unknown. Why this can be useful is covered shortly.

6.7.2 Adding New Nodes With The Node Creation Wizard

Besides adding nodes using the add command of cmsh or the Add button of cmgui as in the previous section, there is also a cmgui wizard that guides the administrator through the process—the *node creation wizard*. This is useful when adding many nodes at a time. It is available from the Nodes resource, by selecting the Overview tabbed pane and then the "Create Nodes" button (figure 6.28).

This wizard should not be confused with the closely related node *identification* wizard described earlier in section 6.4.2, which identifies unassigned MAC addresses and switch ports, and helps assign them node names.

The node *creation* wizard instead creates an object for nodes, assigns them node names, but it leaves the MAC address field for these nodes unfilled, keeping the node object as a “placeholder”(figure 6.29).

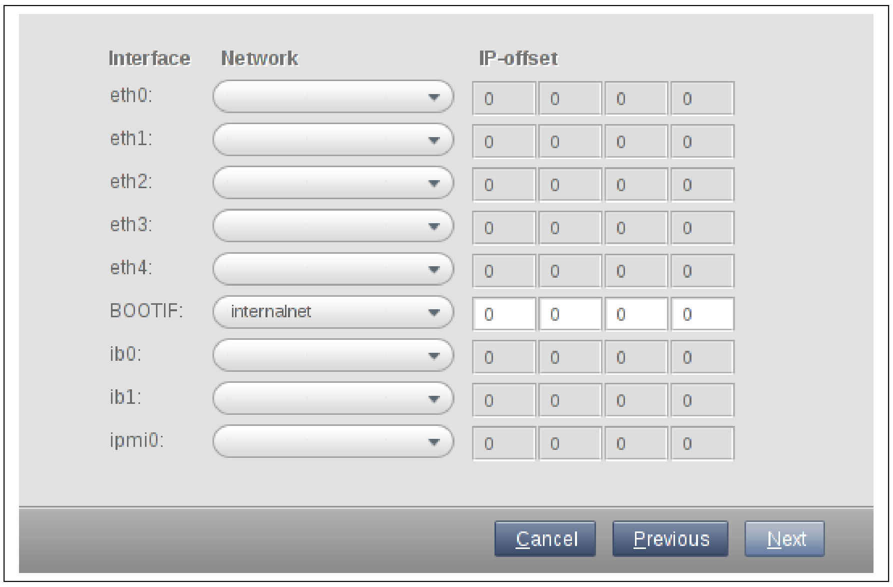


Modified	Hostname	MAC	Category	IP
	node001	00:0C:29:7A:41:78	default	10.141.0.1
✓	node002	00:00:00:00:00:00	default	10.141.0.2
✓	node003	00:00:00:00:00:00	default	10.141.0.3
✓	node004	00:00:00:00:00:00	default	10.141.0.4
✓	node005	00:00:00:00:00:00	default	10.141.0.5
✓	node006	00:00:00:00:00:00	default	10.141.0.6
✓	node007	00:00:00:00:00:00	default	10.141.0.7
✓	node008	00:00:00:00:00:00	default	10.141.0.8
✓	node009	00:00:00:00:00:00	default	10.141.0.9
✓	node010	00:00:00:00:00:00	default	10.141.0.10
✓	node011	00:00:00:00:00:00	default	10.141.0.11

Figure 6.29: Node Creation Wizard: 10 Placeholders Created

The MAC addresses can be assigned to a node via the node identification wizard. However, leaving nodes in a “placeholder” state, where the MAC address entry is left unfilled, means that any new node with an unassigned MAC address that is started up is offered a choice out of the created node names by the provisioning system at its console. This happens when the node-installer reaches the node configuration stage during node boot as described in section 6.4.2. This is sometimes preferable to associating the node name with a MAC address remotely.

The node creation wizard can set IP addresses for the nodes. At one point in the dialog a value for `IP-offset` can also be set (figure 6.30).



Interface	Network	IP-offset			
eth0:		0	0	0	0
eth1:		0	0	0	0
eth2:		0	0	0	0
eth3:		0	0	0	0
eth4:		0	0	0	0
BOOTIF:	internalnet	0	0	0	0
ib0:		0	0	0	0
ib1:		0	0	0	0
ipmi0:		0	0	0	0

Figure 6.30: Node Creation Wizard: Setting Interfaces

The default setting for `IP-offset` is `0.0.0.0`, and means the default IP address is suggested for assignment to each node in the range. The default IP address is based on the node name, with node001 having the value 10.141.0.1, and so on. An offset of x implies that the xth IP address after the default is suggested for assignment to each node in the range. Some care must be taken when setting IP addresses using the wizard, since no duplicate IP address checking is done.

Example

A node001 has its default IP address 10.141.0.1. The node005 is then added.

- If `IP-offset`=0.0.0.0, then 10.141.0.5 is suggested for assignment to node005, because, by default, the node name is parsed and its default IP address suggested.
- If `IP-offset`=0.0.0.2, then 10.141.0.7 is suggested for assignment to node005, because it is 2 IP addresses after the default.

The `cmsh` equivalent of the node creation wizard is the `foreach` loop with the `-clone` option acting on a node (section 3.5.5).

6.8 Troubleshooting The Node Boot Process

During the node boot process there are several common issues that can lead to an unsuccessful boot. This section describes some of these issues and their solutions. It also provides general hints on how to analyze boot problems.

6.8.1 Node Fails To PXE Boot

Possible reasons to consider if a node is not even starting to PXE boot in the first place:

- DHCP may not be running. A check can be done to confirm that DHCP is running on the internal network interface (usually eth0):

  ```
  [root@ddnmon61 ~]# ps u -C dhcpd
  USER PID %CPU %MEM VSZ  RSS TTY STAT START TIME COMMAND
  root 2448 0.0 0.0 11208 436 ?   Ss   Jan22 0:05 /usr/sbin/dhcpd eth0
  ```

  This may indicate that in `cmgui` the `Allow node booting` checkbox in figure4.6 (page 95) needs to be ticked. The equivalent in `cmsh` is to check if the response to:

  ```
  cmsh -c "network use internalnet; get nodebooting"
  ```

  needs to be set to `yes`.

- A rogue DHCP server may be running. If there are all sorts of other machines on the network the nodes are on, then it is possible that there is a rogue DHCP server active on it, perhaps on an IP address that the administrator has forgotten, and interfering with the expected PXE booting. Such stray DHCP servers should be eliminated.

  In such a case, removing all the connections and switches and just connecting the head node directly to a problem node, NIC-to-NIC, should allow a normal PXE boot to happen. If a normal PXE boot then does happen, it indicates the problem is indeed due to a rogue DHCP server on the more-connected network.

- The boot sequence may be set wrongly in the BIOS. The boot interface should normally be set to be the first boot item in the BIOS.

- There may a bad cable connection. This can be due to moving the machine, or heat creep, or another physical connection problem. Firmly inserting the cable into its slot may help. Replacing the cable or interface as appropriate may be required.

- There may a problem with the switch. Removing the switch and connecting a head node and a regular node directly with a cable can help troubleshoot this.

- The cable may be connected to the wrong interface. By default, eth0 is assigned the internal network interface, and eth1 the external network interface on the head node for a type 1 network. However:

  – The two interfaces can be confused when physically viewing them and a connection to the wrong interface can therefore be made.

  – It is also possible that the administrator has changed the default assignment.

  The connections should be checked to eliminate these possibilities.

- The TFTP server that sends out the image may have hung. During a normal run, an output similar to this appears when an image is in the process of being served:

```
[root@ddnmon61 ~]# ps ax | grep [t]ftp
 7512 ?        Ss     0:03 in.tftpd --maxthread 500 /tftpboot
```

If the TFTP server is in a zombie state, the head node should be rebooted. If the TFTP service hangs regularly, there is likely a networking hardware issue that requires resolution.

Incidentally, grepping the process list for a TFTP service returns nothing when the head node is listening for TFTP requests, but not actively serving a TFTP image. This is because the TFTP service runs under xinet.d and is called on demand. Running

```
[root@ddnmon61 ~]# chkconfig --list
```

should include in its output the line:

```
tftp:      on
```

if TFTP is running under xinet.d.

- Sometimes a manufacturer releases hardware with buggy drivers that have a variety of problems. For instance: Ethernet frames may be detected at the interface (for example, by `ethtool`), but TCP/IP packets may not be detected (for example, by `wireshark`). In that case, the manufacturer should be contacted to upgrade their driver.
- The interface may have a hardware failure. In that case, the interface should be replaced.

6.8.2 Node-installer Logging

If the node manages to get beyond the PXE stage to the node-installer stage, then the first place to look for hints on node boot failure is usually the node-installer log file. The node-installer runs on the node that is being provisioned, and sends logging output to the syslog daemon running on that node. This forwards all log data to the IP address from which the node received its DHCP lease, which is typically the IP address of the head node or failover node. In a default DirectMon setup, the `local5` facility of the `syslog` daemon is used on the node that is being provisioned to forward all node-installer messages to the log file `/var/log/node-installer` on the head node.

After the node-installer has finished running, its log is also stored in `/var/log/node-installer` on the regular nodes.

Optionally, extra log information can be written by enabling debug logging, which sets the syslog importance level at `LOG_DEBUG`. To enable debug logging, the debug field is changed in `/cm/node-installer/scripts/node-installer.conf`.

From the console of the booting node the log file is generally accessible by pressing `Alt+F7` on the keyboard. Debug logging is however excluded from being viewed in this way, due to the output volume making this impractical.

A booting node console can be accessed remotely if Serial Over LAN (SOL) is enabled (section 12.6), to allow the viewing of console messages directly. A further depth in logging can be achieved by setting the kernel option `loglevel=N`, where `N` is a number from `0` (`KERN_EMERG`) to `7` (`KERN_DEBUG`).

6.8.3 Provisioning Logging

The provisioning system sends log information to the CMDaemon log file. By default this is in `/var/log/cmdaemon` on the local host, that is, the provisioning host. The host this log runs on can be configured with the CMDaemon directive `SyslogHost` (Appendix C).

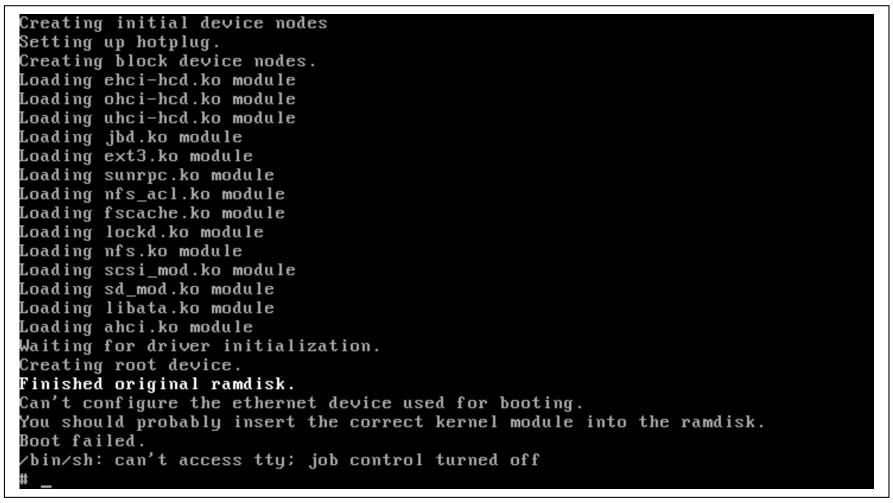
The image synchronization log file can be retrieved with the `synclog` command running from device mode in `cmsh` (section 6.4.7). Hints on provisioning problems are often found by looking at the tail end of the log.

6.8.4 Ramdisk Fails During Loading Or Sometime Later

One issue that may come up after a software image update via yum or zypper (section 10.4), is that the ramdisk stage may fail during loading or sometime later, for a node that is rebooted after the update. This occurs if there are instructions to modify the ramdisk by the update. In a normal machine the ramdisk would be regenerated. In a cluster, the extended ramdisk that is used requires an update, but DirectMon is not aware of this. Running the `createramdisk` command from `cmsh` or `cmgui` (section 6.3.2) generates an updated ramdisk for the cluster, and solves the failure for this case.

6.8.5 Ramdisk Cannot Start Network

The ramdisk must activate the node's network interface in order to fetch the node-installer. To activate the network device, the correct kernel module needs to be loaded. If this does not happen, booting fails, and the console of the node displays something similar to figure 6.31.



```
Creating initial device nodes
Setting up hotplug.
Creating block device nodes.
Loading ehci-hcd.ko module
Loading ohci-hcd.ko module
Loading uhci-hcd.ko module
Loading jbd.ko module
Loading ext3.ko module
Loading sunrpc.ko module
Loading nfs_acl.ko module
Loading fscache.ko module
Loading lockd.ko module
Loading nfs.ko module
Loading scsi_mod.ko module
Loading sd_mod.ko module
Loading libata.ko module
Loading ahci.ko module
Waiting for driver initialization.
Creating root device.
Finished original ramdisk.
Can't configure the ethernet device used for booting.
You should probably insert the correct kernel module into the ramdisk.
Boot failed.
/bin/sh: can't access tty; job control turned off
# _
```

Figure 6.31: No Network Interface

To solve this issue the correct kernel module should be added to the software image's kernel module configuration (section 6.3.2). For example, to add the e1000 module to the default image using `cmsh`:

Example

```
[mc]% softwareimage use default-image
[mc->softwareimage[default-image]]% kernelmodules
```

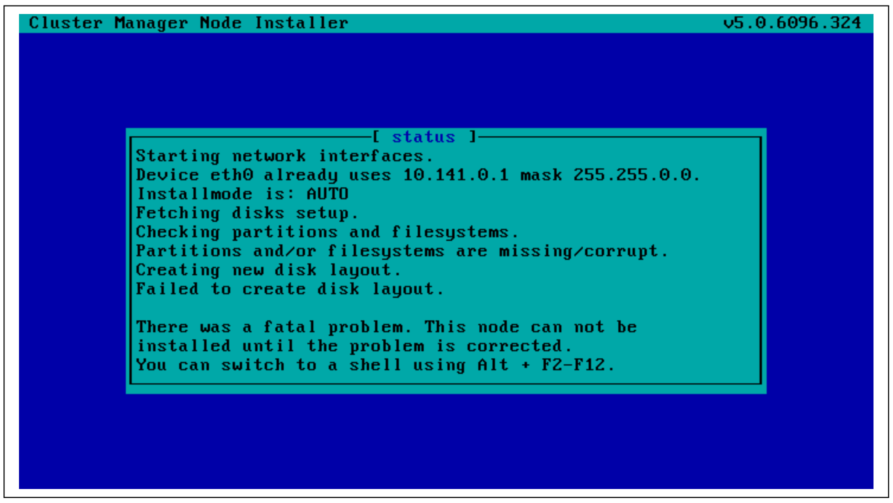
```
[mc->softwareimage[default-image]->kernelmodules]% add e1000
[mc->softwareimage[default-image]->kernelmodules[e1000]]% commit
Initial ramdisk for image default-image was regenerated successfully
[mc->softwareimage[default-image]->kernelmodules[e1000]]%
```

After committing the change it typically takes about a minute before the initial ramdisk creation is completed via a `mkinitrd` run by CMDaemon.

6.8.6 Node-Installer Cannot Create Disk Layout

When the node-installer is not able to create a drive layout it displays a message similar to figure 6.32. The node-installer log file (section 6.8.2) contains something like:

```
Mar 24 13:55:31 10.141.0.1 node-installer: Installmode is: AUTO
Mar 24 13:55:31 10.141.0.1 node-installer: Fetching disks setup.
Mar 24 13:55:31 10.141.0.1 node-installer: Checking partitions and
filesystems.
Mar 24 13:55:32 10.141.0.1 node-installer: Detecting device '/dev/sda':
not found
Mar 24 13:55:32 10.141.0.1 node-installer: Detecting device '/dev/hda':
not found
Mar 24 13:55:32 10.141.0.1 node-installer: Can not find device(s) (/dev\
/sda /dev/hda).
Mar 24 13:55:32 10.141.0.1 node-installer: Partitions and/or filesystems
are missing/corrupt. (Exit code 4, signal 0)
Mar 24 13:55:32 10.141.0.1 node-installer: Creating new disk layout.
Mar 24 13:55:32 10.141.0.1 node-installer: Detecting device '/dev/sda':
not found
Mar 24 13:55:32 10.141.0.1 node-installer: Detecting device '/dev/hda':
not found
Mar 24 13:55:32 10.141.0.1 node-installer: Can not find device(s) (/dev\
/sda /dev/hda).
Mar 24 13:55:32 10.141.0.1 node-installer: Failed to create disk layout.
(Exit code 4, signal 0)
Mar 24 13:55:32 10.141.0.1 node-installer: There was a fatal problem. T\
his node can not be installed until the problem is corrected.
```



```
Cluster Manager Node Installer                          v5.0.6096.324

[ status ]
Starting network interfaces.
Device eth0 already uses 10.141.0.1 mask 255.255.0.0.
Installmode is: AUTO
Fetching disks setup.
Checking partitions and filesystems.
Partitions and/or filesystems are missing/corrupt.
Creating new disk layout.
Failed to create disk layout.

There was a fatal problem. This node can not be
installed until the problem is corrected.
You can switch to a shell using Alt + F2-F12.
```

Figure 6.32: No Disk

It is likely that this issue is caused by the correct storage driver not being loaded. To solve this issue the correct kernel module should be added to the software image's kernel module configuration (section 6.3.2).

Experienced system administrators work out what drivers may be missing by checking the results of hardware probes. For example, the output of `lspci` provides a list of hardware detected in the PCI slots, giving the chipset name of the storage controller hardware in this case:

Example

```
[root@ddnmon61 ~]# lspci | grep SCSI
00:10.0 Serial Attached SCSI controller: LSI Logic / Symbios Logic SAS2\
008 PCI-Express Fusion-MPT SAS-2 [Falcon] (rev 03)
```

The next step is to Google with likely search strings based on that output.

The Linux Kernel Driver DataBase (LKDDb) is a hardware database built from kernel sources that lists driver availability for Linux. It is available at `http://cateee.net/lkddb/`. Using the Google search engine's "`site`" operator to restrict results to the `cateee.net` web site only, a likely string to try might be:

Example

```
SAS2008 site:cateee.net
```

The search result indicates that the `mpt2sas` kernel module needs to be added to the node kernels. A look in the modules directory of the software image shows if it is available:

Example

```
find /cm/images/default-image/lib/modules/ -name "*mpt2sas*"
```

If it is is not available, the driver module must then be obtained. If it is a source file, it will need to be compiled. By default, nodes run on standard distribution kernels, so that only standard procedures need to be followed to compile modules.

If the module is available, it can be added to the default image using `cmsh` in `softwareimage` mode:

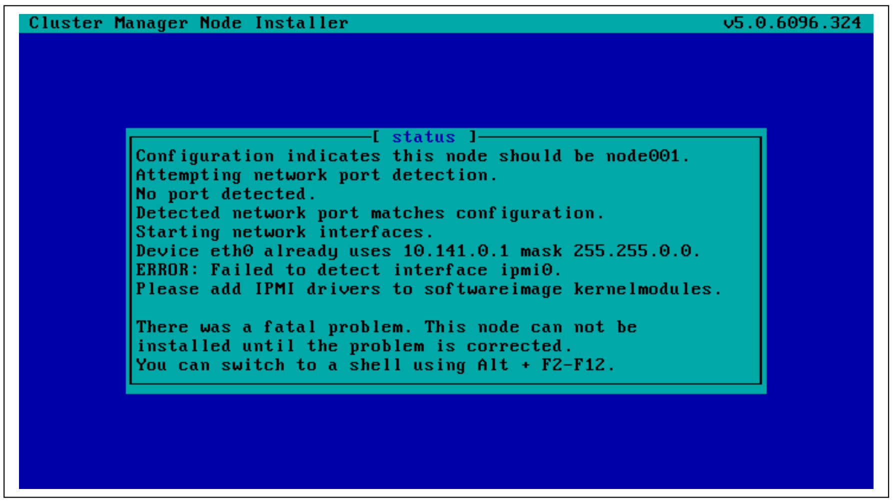
Example

```
[ddnmon61]% softwareimage use default-image
[ddnmon61->softwareimage[default-image]]% kernelmodules
[ddnmon61->softwareimage[default-image]->kernelmodules]% add mpt2sas
[ddnmon61->softwareimage[default-image]->kernelmodules*[mpt2sas*]]% com\
mit
[ddnmon61->softwareimage[default-image]->kernelmodules[mpt2sas]]%
Thu May 19 16:54:52 2011 [notice] ddnmon61: Initial ramdisk for image de\
fault-image is being generated
[ddnmon61->softwareimage[default-image]->kernelmodules[mpt2sas]]%
Thu May 19 16:55:43 2011 [notice] ddnmon61: Initial ramdisk for image de\
fault-image was regenerated successfully.
[ddnmon61->softwareimage[default-image]->kernelmodules[mpt2sas]]%
```

After committing the change it can take some time before ramdisk creation is completed—typically about a minute, as the example shows. On rebooting the node, it should now continue past the disk layout stage.

6.8.7 Node-Installer Cannot Start BMC (IPMI/iLO) Interface

In some cases the node-installer is not able to configure a node's BMC interface, and displays an error message similar to figure 6.33.



```
Cluster Manager Node Installer                                      v5.0.6096.324

                    [ status ]
Configuration indicates this node should be node001.
Attempting network port detection.
No port detected.
Detected network port matches configuration.
Starting network interfaces.
Device eth0 already uses 10.141.0.1 mask 255.255.0.0.
ERROR: Failed to detect interface ipmi0.
Please add IPMI drivers to softwareimage kernelmodules.

There was a fatal problem. This node can not be
installed until the problem is corrected.
You can switch to a shell using Alt + F2-F12.
```

Figure 6.33: No BMC Interface

Usually the issue can be solved by adding the correct BMC (IPMI/iLO) kernel modules to the software image's kernel module configuration. However, in some cases the node-installer is still not able to configure the BMC interface. If this is the case the BMC probably does not support one of the commands the node-installer uses to set specific settings.

The `setupBmc` Node-Installer Configuration Setting

To solve this issue, setting up BMC interfaces can be disabled globally by setting the `setupBmc` field in the node-installer configuration file `/cm/node-installer/scripts/node-installer.conf` to `false`. Doing this disables configuration of all BMC interfaces by the node-installer. A custom `finalize` script (appendix E) can then be used to run the required commands instead.

The `setupBmc` field in the node-installer should not be confused with the `SetupBMC` directive in `cmd.conf` (Appendix C). The former is about enabling the BMC interface, while the latter is about enabling automated passwords to the BMC interface (an interface that must of course be enabled in the first place to work).

The `failOnMissingBmc` Node-Installer Configuration Setting

If the kernel modules for the BMC are loaded up correctly, and the BMC is configured, but it is not detected by the node-installer, then the node-installer halts by default. This corresponds to the setting `failOnMissingBmc = true` in the node-installer configuration file `/cm/node-installer/scripts/node-installer.conf`. Toggling this to `false` skips BMC network device detection, and lets the node-installer continue past the BMC detection and configuration stage. This can be convenient, for example, if the BMC is not yet configured and the aim is to get on with setting up the rest of the cluster.

7 Cloudbursting

This chapter covers *cloudbursting*. In weather, a cloudburst is used to convey the idea that a sudden flood of cloud contents takes place. In cluster computing, the term cloudbursting conveys the idea that a flood of extra cluster capacity is made available when needed from a cloud computing services provider such as Amazon.

DirectMon implements cloudbursting for two scenarios:

1. A "Cluster-On-Demand", or a "pure" cloud cluster (section 7.1). In this scenario, the entire cluster can be started up on demand from a state of non-existence. All nodes, including the head node, are instances running in a coordinated manner entirely inside the cloud computing service.

2. A "Cluster Extension", or a "hybrid" cloud cluster (section 7.2). In this scenario, the head node is kept outside the cloud. Zero or more regular nodes are also run outside the cloud. When additional capacity is required, the cluster is extended via cloudbursting to make additional nodes available from within the cloud.

7.1 Cluster-On-Demand Cloudbursting

Requirements If the cloud provider is Amazon, then Cluster-On-Demand cloudbursting (the case of starting up a "pure" cloud cluster) requires:

- an Amazon account
- registration on the DDN Customer Portal website at `http://www.ddn.com/Customer-Login.php`
- a DirectMon product key. The key is obtained at the Customer Portal website specifically for a Cluster-On-Demand setup, from the `Burst!` menu. This key is later activated when the license is installed (section 7.1.2) on the head node. The head node and regular nodes in this case are in the cloud.

Steps The following steps are then carried out to start up the head node and regular nodes of the cloud cluster:

- a head node instance is launched from a browser, using the Amazon management console (section 7.1.1)
- the head node instance is logged into via `ssh` and the cluster is configured (section 7.1.2)
- the regular nodes are started up from the head node using `cmsh` or `cmgui` to power them up (section 7.1.4)

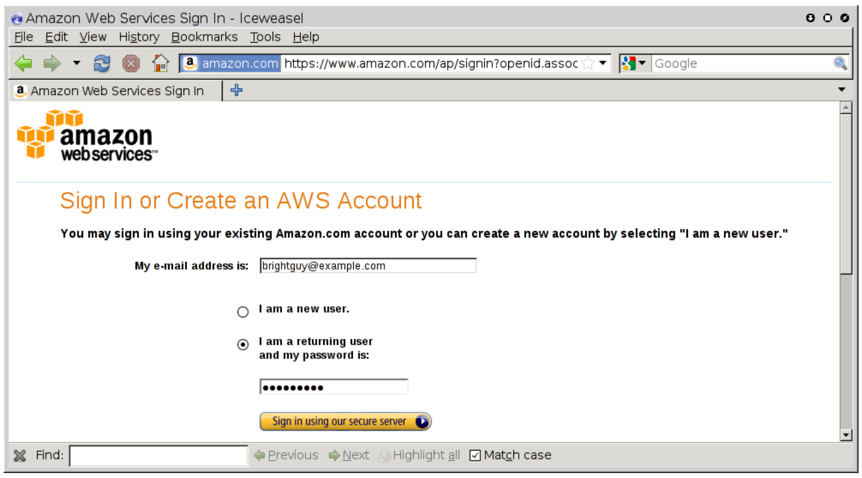
These steps are now covered in more detail.

7.1.1 Cluster-On-Demand: Launching The Head Node From The Cloud Provider

Launching a head node from within Amazon is covered in this section.

Getting To The "`Launch Instance`" Button

The Amazon management console can be logged into from `https://console.aws.amazon.com/console/` by using the e-mail address and password of the Amazon account (figure 7.1).


Amazon Web Services Sign In - Iceweasel
File Edit View History Bookmarks Tools Help
amazon.com https://www.amazon.com/ap/signin?openid.assoc
Google
Amazon Web Services Sign In
amazon web services
Find: Previous Next Highlight all Match case

Sign In or Create an AWS Account

You may sign in using your existing Amazon.com account or you can create a new account by selecting "I am a new user."

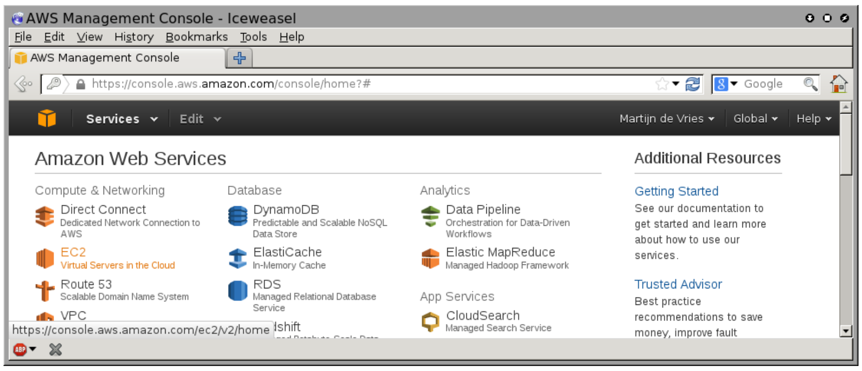
My e-mail address is: brightguy@example.com

I am a new user. [✓ I am a returning user and my password is:] ••••••••••

Sign in using our secure server

Figure 7.1: Logging Into The Amazon Management Console

By default, on login, the management console displays a list of accessible Amazon web services (figure 7.2).


AWS Management Console - Iceweasel
File Edit View History Bookmarks Tools Help
AWS Management Console
https://console.aws.amazon.com/console/home?#
Google
Services Edit
Martijn de Vries Global Help
https://console.aws.amazon.com/ec2/v2/home

Amazon Web Services

Compute & Networking
- Direct Connect — Dedicated Network Connection to AWS
- EC2 — Virtual Servers in the Cloud
- Route 53 — Scalable Domain Name System
- VPC

Database
- DynamoDB — Predictable and Scalable NoSQL Data Store
- ElastiCache — In-Memory Cache
- RDS — Managed Relational Database Service
- Redshift

Analytics
- Data Pipeline — Orchestration for Data-Driven Workflows
- Elastic MapReduce — Managed Hadoop Framework

App Services
- CloudSearch — Managed Search Service

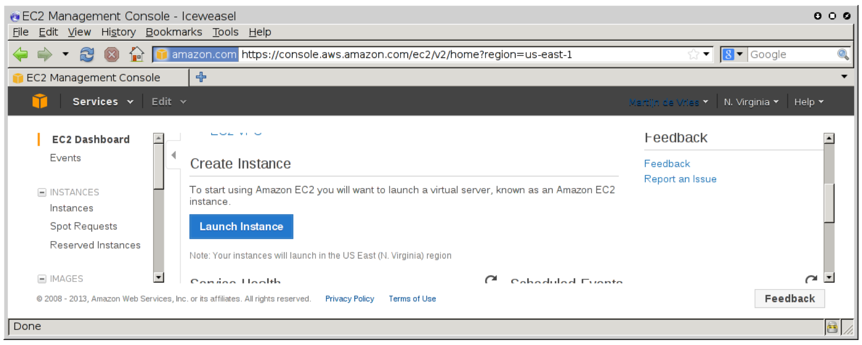
Additional Resources

Getting Started — See our documentation to get started and learn more about how to use our services.

Trusted Advisor — Best practice recommendations to save money, improve fault

Figure 7.2: Amazon Management Console: Accessible Services

To set up the Cluster-On-Demand cluster, the `EC2` service within the `Compute & Networking` grouping should be clicked. This brings up the EC2 Dashboard, which is also the top link of a resource tree that is displayed in a Navigation pane (figure 7.3).



To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US East (N. Virginia) region

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 7.3: The EC2 Dashboard With The "Launch Instance" Button

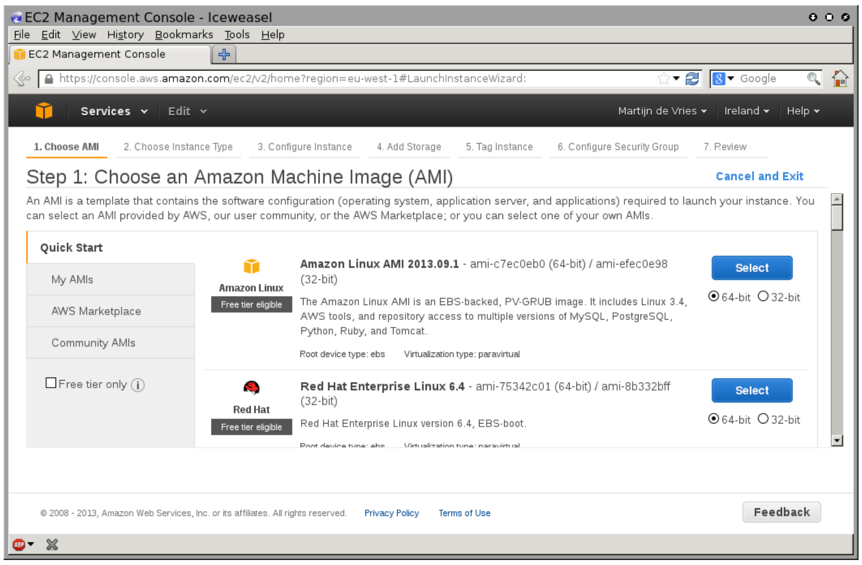
In the main pane of the dashboard is the Launch Instance button. Clicking it starts up Amazon's Launch Instance Wizard. Amazon documentation for the wizard is at http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/launching-instance.html.

Using the wizard to launch a head node instance is described next.

Launching The Head Node Instance

To start a Cluster-On-Demand cluster, a head node instance must first be launched. This can be done as follows:

- Step 1: Choose an Amazon Machine Image (AMI): The first step in the wizard offers a choice of Select buttons to launch an instance from an AMI image (figure 7.4).



An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Amazon Linux AMI 2013.09.1 - ami-c7ec0eb0 (64-bit) / ami-efec0e98 (32-bit)

The Amazon Linux AMI is an EBS-backed, PV-GRUB image. It includes Linux 3.4, AWS tools, and repository access to multiple versions of MySQL, PostgreSQL, Python, Ruby, and Tomcat.

Root device type: ebs Virtualization type: paravirtual

Red Hat Enterprise Linux 6.4 - ami-75342c01 (64-bit) / ami-8b332bff (32-bit)

Red Hat Enterprise Linux version 6.4, EBS-boot.

Root device type: ebs Virtualization type: paravirtual

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 7.4: EC2: Choosing An AMI, Step 1

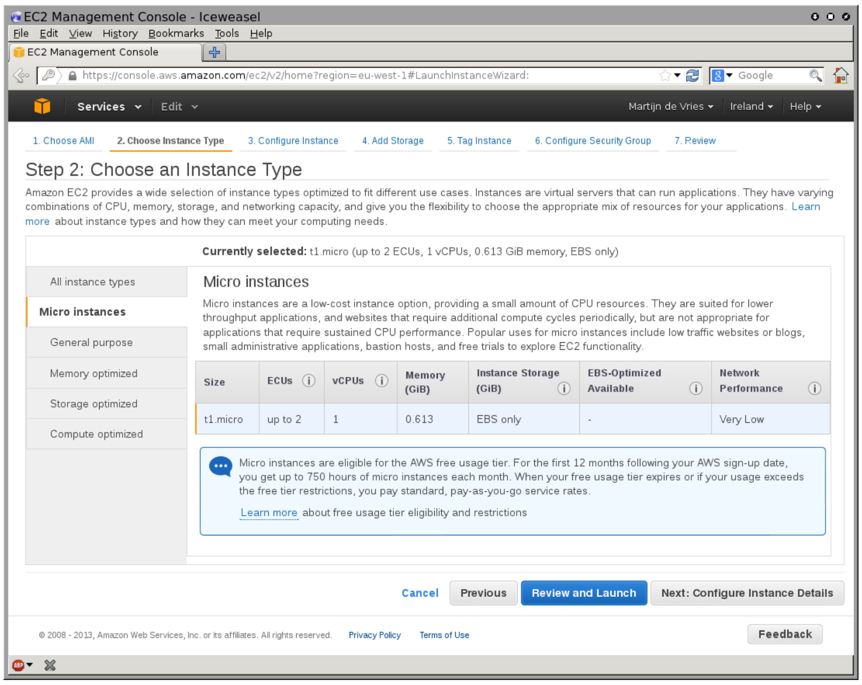
The default AMIs can be ignored. Clicking on the Community AMIs link in the left navigation pane brings up a new display of community AMIs. Entering a search text of "ddnheadnode" then shows only the AMIs appropriate for a DirectMon head node instance in a Cluster-On-Demand cluster. These are:

1. An AMI that uses standard XEN paravirtualization technology. This is available for all regions. If this image is used, hardware virtualization extensions acceleration is not implemented, even if available in the underlying cloud node hardware.
2. An AMI with `hvm` in the name. This is available for some regions. It is intended for use in regions that support HVM (Hardware Virtual Machines). HVM requires that the CPU used has the Intel VT or AMD-V virtualization extensions, to implement hardware acceleration for virtualized machines. At the time of checking (April 2013):
   - Regions supporting HVM are `eu-west-1`, `us-east-1`, and `us-west-2`.
   - Instance types supporting HVM are the `m3.xlarge` instance type, and higher. Instance types (`http://aws.amazon.com/ec2/instance-types/`) are a way of characterizing machine specifications, such as whether it has more RAM, more cores, or HVM.

   Updated details on the regions and instance types that Amazon EC2 supports can be found via the Amazon website, `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html`.

Clicking on the `Select` button for the appropriate XEN or HVM head node AMI then brings up the next step in the launch wizard:

- `Step 2: Choose an Instance Type`: This displays a *micro instance* by default (figure 7.5).


EC2 Management Console - Iceweasel
File Edit View History Bookmarks Tools Help
EC2 Management Console
https://console.aws.amazon.com/ec2/v2/home?region=eu-west-1#LaunchInstanceWizard:
Google
Services
Edit
Martijn de Vries
Ireland
Help
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Currently selected: t1.micro (up to 2 ECUs, 1 vCPUs, 0.613 GiB memory, EBS only)

Micro instances

Micro instances are a low-cost instance option, providing a small amount of CPU resources. They are suited for lower throughput applications, and websites that require additional compute cycles periodically, but are not appropriate for applications that require sustained CPU performance. Popular uses for micro instances include low traffic websites or blogs, small administrative applications, bastion hosts, and free trials to explore EC2 functionality.

Size	ECUs	vCPUs	Memory (GiB)	Instance Storage (GiB)	EBS-Optimized Available	Network Performance
t1.micro	up to 2	1	0.613	EBS only	-	Very Low

Micro instances are eligible for the AWS free usage tier. For the first 12 months following your AWS sign-up date, you get up to 750 hours of micro instances each month. When your free usage tier expires or if your usage exceeds the free tier restrictions, you pay standard, pay-as-you-go service rates.

Learn more about free usage tier eligibility and restrictions

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 7.5: EC2: Choosing An AMI, Step 2

The t1.micro is the smallest and least powerful type of instance that a head node can run as, but is only useful for quite minor testing. It is likely to be overwhelmed when doing any significant work. A more reasonable node type to use for testing is therefore the m1.small type, which is available under the General purpose navigation option of this window.

Steps 3 to 6 that follow are optional and can be skipped, by going ahead to Step 7: Review Instance Launch.

- Step 3: Configure Instance Details: Among other instance options, this optional step allows the following to be set:
  - Purchasing option, for spot instances (section 7.4.3)
  - Network This is a choice of EC2-Classic or EC2-VPC instances (section 7.5.1)
  - Availability Zone, for if there is a preference for the location of the instance. Nodes within the same availability zone can connect with low latency and high bandwidth to each other. They are also isolated from other availability zones in the same region, which reduces the risk of network outages from another zone affecting them. By default, no preference is specified for the head node, nor for the cloud nodes later. This is because spot pricing can increase as nodes from an availability zone become scarce, which may conflict with the intention of the administrator. The default availability setting for a cloud account can be set in cmsh from within cloud mode:

    Example

    ```
    [ddnmon61->cloud[Spare Capacity]]% set defaultavailabilityzone
    ```
  - Shutdown behavior, to decide if the instance should be stopped (kept around) or terminated (removed).
- Step 4: Add Storage: Among other storage options, this optional step allows the following options to be set:
  - Size (GB): The size of storage to be added to the instance
  - Type: Whether the storage is EBS or ephemeral
  - Device: A device name, chosen from /dev/sdb onwards, since /dev/sda is already taken
  - Delete on Termination: Whether the device is deleted when the instance terminates

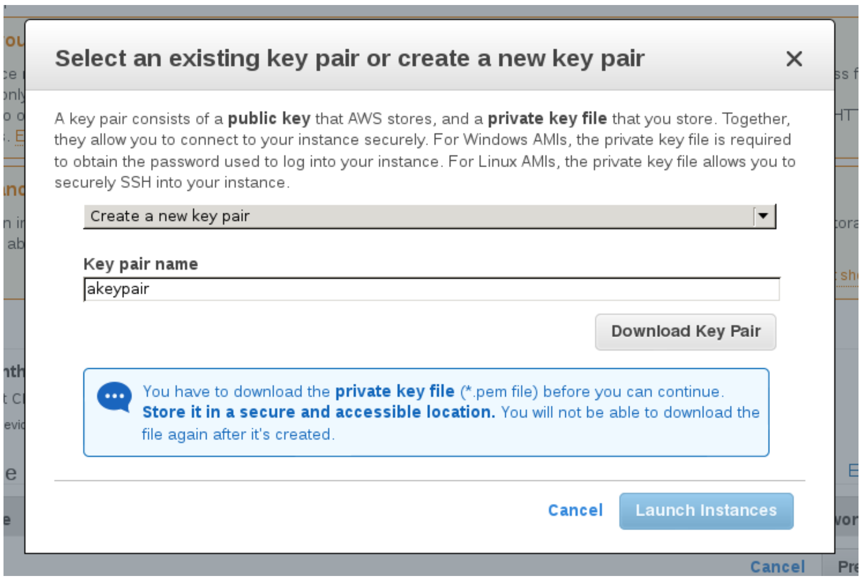
By default, the instance has a Type called Root, which is a special EBS volume. It has a default Size (GB) of 80, which can be edited.

For most instances other than micro, a certain amount of ephemeral storage is provided by Amazon for free, and can then be set for the root device in the Storage Device Configuration options of this screen. The EBS and ephemeral storage types are described in section 7.2.2.

- `Step 5: Tag instance`: This optional step allows the addition of metadata to an instance, via assignment of key-value pairs. A default key of `Name` is presented, and the administrator should put in a name for the instance as the associated value. The associated value can be arbitrary.
- `Step 6: Configure Security Group`: This optional step allows a *security group* to be defined. A security group is a configuration that defines how network access to the instance is allowed. By default all access to the instance is blocked, apart from SSH access.
  - Default: SSH inbound allowed. This means that `cmsh` can be used to control the Cluster-On-Demand cluster via SSH just like a regular cluster.

  Inbound connections can be defined, based on protocol, packet type, port, and source in CIDR specification. For example, allowing inbound connections via TCP port 8081 from anywhere (0.0.0.0/0) allows `cmgui` to communicate via its custom protocol with the default CMDaemon back end on the head node.

  The default security group setting should also be modified by the administrator at this point if a standalone `cmgui` is to be used to control the cluster (section 7.1.3). For regular use in a cluster-on-demand setup, lag is reduced if a standalone `cmgui` is used rather than running a `cmgui` originating from the head node via an `ssh -X` connection.
- `Step 7: Review Instance Launch`: The configuration so far can be reviewed. On clicking the `Launch` button, a pop-up dialog for "`Select an existing key pair or create a new key pair`" is displayed (figure 7.6).


Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Create a new key pair

Key pair name
akeypair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Figure 7.6: EC2: Choosing An AMI, Step 7 - Keypair generation/creation dialog

This dialog allows the creation and storage of a cryptographic key pair. It can alternatively allow an existing pair to be used from the "`Select a key pair`" selection. The private key of the key pair is used in order to allow SSH access to the head node instance when it is up and running.

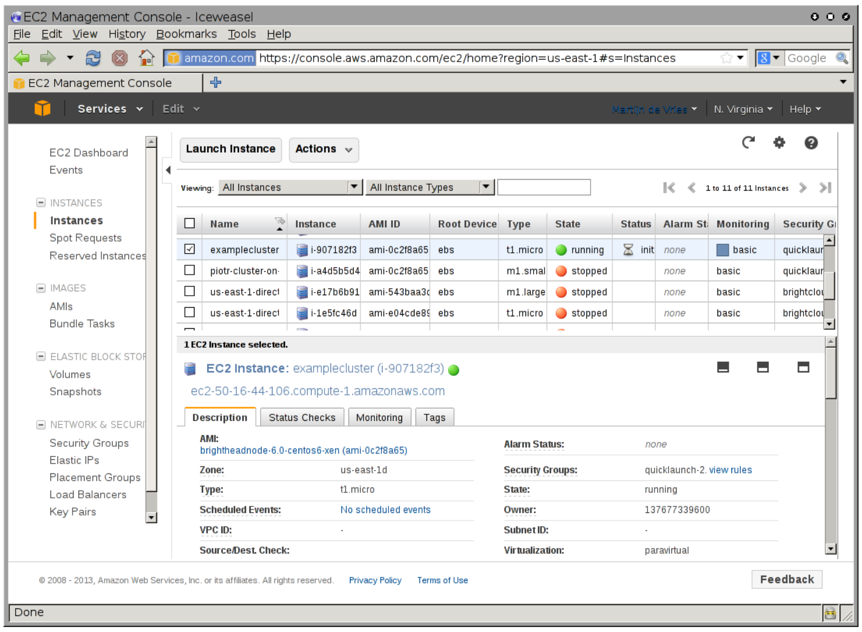
After the instance is launched, the web session displays a window informing that the instance is being started up.

Managing A Head Node Instance With The AWS EC2 Management Console

A *newly-launched* head node instance, after it is fully up, is a fully-booted and running Linux instance, but it is not yet a fully-configured head node. That is, it is capable of running DirectMon, but it is not yet running it, nor is it working with compute nodes at this point. The steps to make it a fully-configured head node are covered in section 7.1.2.

For now, the newly-launched head node instance can be watched and managed without DirectMon in the following ways.

Status checking via instance selection from instances list: Clicking the `Instances` menu resource item from the navigation pane opens up the "`Instances`" pane. This lists instances belonging to the account owner. An instance can be marked by ticking its checkbox. Information for the selected instance is then displayed in the lower main pane (figure 7.7).



	Name	Instance	AMI ID	Root Device	Type	State	Status	Alarm St	Monitoring	Security G
✓	examplecluster	i-907182f3	ami-0c2f8a65	ebs	t1.micro	● running	init	none	basic	quicklaur
	piotr-cluster-on-	i-a4d5b5d4	ami-0c2f8a65	ebs	m1.smal	● stopped		none	basic	quicklaur
	us-east-1-direct	i-e17b6b91	ami-543baa3c	ebs	m1.large	● stopped		none	basic	brightclou
	us-east-1-direct	i-1e5fc46d	ami-e04cde89	ebs	t1.micro	● stopped		none	basic	brightclou

1 EC2 Instance selected.

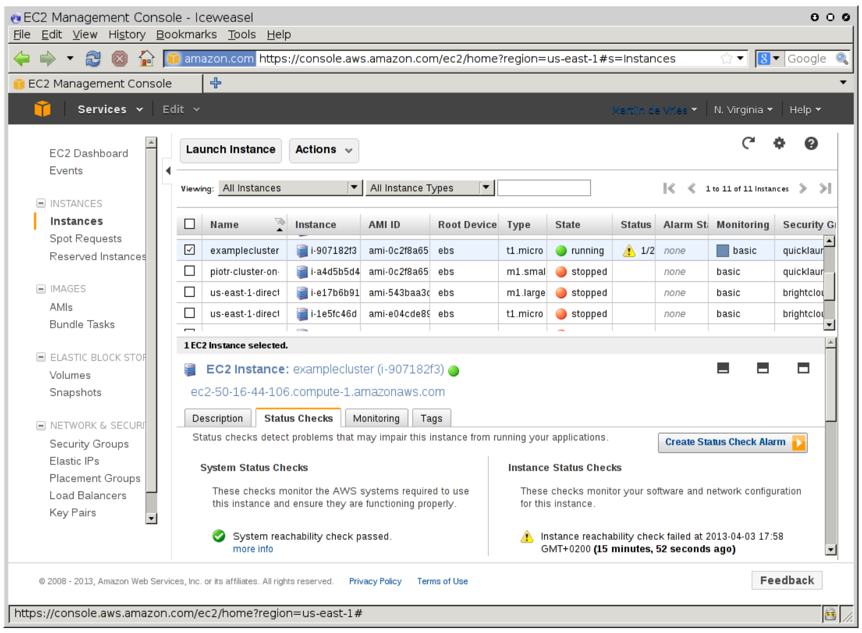
EC2 Instance: examplecluster (i-907182f3)
ec2-50-16-44-106.compute-1.amazonaws.com

AMI: brightheadnode-6.0-centos6-xen (ami-0c2f8a65)		Alarm Status:	none
Zone:	us-east-1d	Security Groups:	quicklaunch-2. view rules
Type:	t1.micro	State:	running
Scheduled Events:	No scheduled events	Owner:	137677339600
VPC ID:	-	Subnet ID:	-
Source/Dest. Check:		Virtualization:	paravirtual

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Figure 7.7: The EC2 Instances List

System (Amazon machine infrastructure) and instance (instance running under the infrastructure) reachabilities are similarly shown under the neighboring "`Status Checks`" tab (figure 7.8).



	Name	Instance	AMI ID	Root Device	Type	State	Status	Alarm St	Monitoring	Security G
✓	examplecluster	i-907182f3	ami-0c2f8a65	ebs	t1.micro	● running	⚠ 1/2	none	basic	quicklaur
	piotr-cluster-on-	i-a4d5b5d4	ami-0c2f8a65	ebs	m1.smal	● stopped		none	basic	quicklaur
	us-east-1-direct	i-e17b6b91	ami-543baa3c	ebs	m1.large	● stopped		none	basic	brightclou
	us-east-1-direct	i-1e5fc46d	ami-e04cde89	ebs	t1.micro	● stopped		none	basic	brightclou

1 EC2 Instance selected.

EC2 Instance: examplecluster (i-907182f3)
ec2-50-16-44-106.compute-1.amazonaws.com

Status checks detect problems that may impair this instance from running your applications.

System Status Checks

These checks monitor the AWS systems required to use this instance and ensure they are functioning properly.

System reachability check passed.

Instance Status Checks

These checks monitor your software and network configuration for this instance.

Instance reachability check failed at 2013-04-03 17:58 GMT+0200 **(15 minutes, 52 seconds ago)**

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 7.8: Reachability Status For An EC2 Instance

Acting on an instance from the AWS EC2 Management Console: An instance can be marked by clicking on it. Clicking the `Actions` button near the top of the main center pane, or equivalently from a right-mouse-button click in the pane, brings up a menu of possible actions. These actions can be executed on the marked instance, and include the options to `Start`, `Stop` or `Terminate` the instance.

Connecting to an instance from the AWS EC2 Management Console: A marked and running instance can have an SSH connection made to it. Clicking on the `Connect` button near the top of the main center pane displays a pop-up text that guides the user through the connection options for a running instance. These connection options are via:

- **a standalone SSH client**

  There is further documentation on this at:

  – `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html` for Linux clients

  – `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html` for PuTTY users

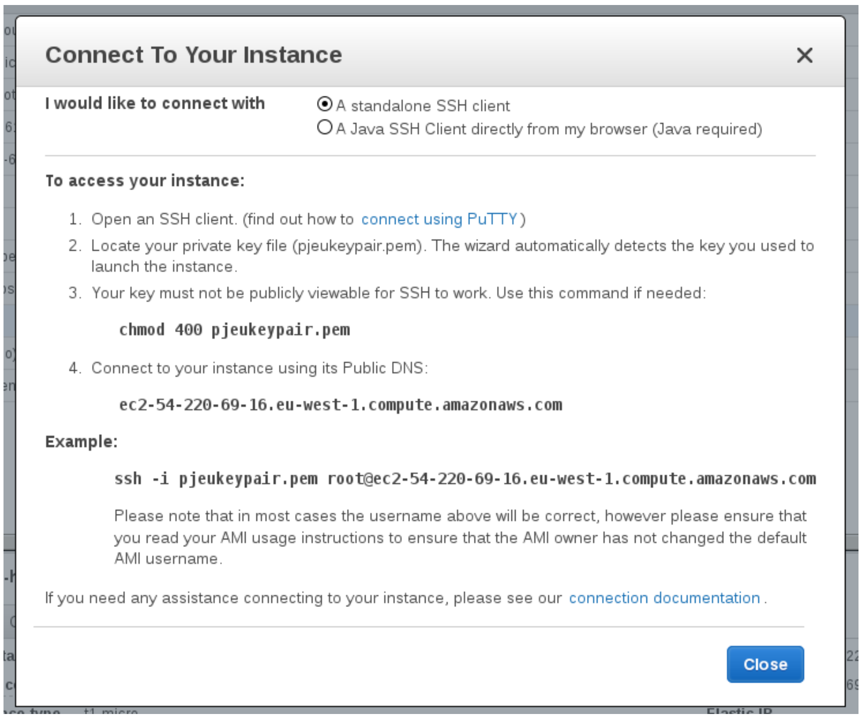
- **a browser-based Java SSH client, MindTerm**

  There is further documentation on this at:

  – `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/mindterm.html`

Most administrators should find the pop-up text enough, and the further documentation unnecessary.

The standalone SSH client help text displays instructions (figure 7.9) on how to run `ssh` from the command line to access the marked instance. If the launched head node is fully up then a login using those instructions succeeds.



Connect To Your Instance

I would like to connect with [✓ A standalone SSH client] A Java SSH Client directly from my browser (Java required)

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (pjeukeypair.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 pjeukeypair.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-54-220-69-16.eu-west-1.compute.amazonaws.com
```

Example:

```
ssh -i pjeukeypair.pem root@ec2-54-220-69-16.eu-west-1.compute.amazonaws.com
```

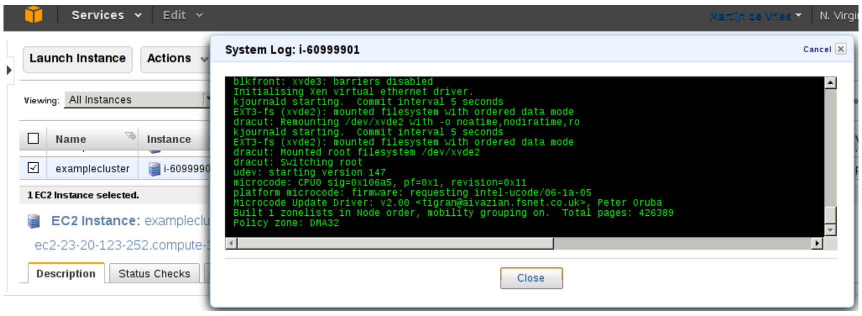
Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our connection documentation.

Close

Figure 7.9: SSH Instructions To Connect To The Marked Instance

Viewing the head node console: The head node takes about 2 minutes to start up. If, on following the instructions, an SSH connection cannot be made, then it can be worth checking the head node system log to check if the head node has started up correctly. The log is displayed on right-clicking on the "`Actions`" button and selecting the "`Get System Log`" menu item (figure 7.10). A successful start of the system generates a log with a tail similar to that of figure 7.10.



System Log: i-60999901

```
blkfront: xvde3: barriers disabled
Initialising Xen virtual ethernet driver.
kjournald starting.  Commit interval 5 seconds
EXT3-fs (xvde2): mounted filesystem with ordered data mode
dracut: Remounting /dev/xvde2 with -o noatime,nodiratime,ro
kjournald starting.  Commit interval 5 seconds
EXT3-fs (xvde2): mounted filesystem with ordered data mode
dracut: Mounted root filesystem /dev/xvde2
dracut: Switching root
udev: starting version 147
microcode: CPU0 sig=0x106a5, pf=0x1, revision=0x11
platform microcode: firmware: requesting intel-ucode/06-1a-05
Microcode Update Driver: v2.00 <tigran@aivazian.fsnet.co.uk>, Peter Oruba
Built 1 zonelists in Node order, mobility grouping on.  Total pages: 426389
Policy zone: DMA32
```

Figure 7.10: System Log Of The Checkboxed Instance

If the system and network are working as expected, then an SSH connection can be made to the head node to carry out the next step, which is the configuration of the head node and cluster.

7.1.2 Cluster-On-Demand: Head Node Login And Cluster Configuration

After the Amazon console manager has started up a head node instance, the head node instance and cluster must be configured. Logging into the head node via `ssh` allows this.

On logging in for the first time, the system suggests that the `ddn-setup` script be run:

Example

```
pj@office:~$ ssh -i pjkeypair.pem root@ec2-176-34-160-197.eu-west-1.com\
pute.amazonaws.com
The authenticity of host 'ec2-176-34-160-197.eu-west-1.compute.amazonaw\
s.com (176.34.160.197)' can't be established.
RSA key fingerprint is 66:1e:f3:77:83:f8:3f:42:c1:b7:d2:d5:56:d8:c3:58.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-176-34-160-197.eu-west-1.compute.amazon\
aws.com,176.34.160.197' (RSA) to the list of known hosts.
Welcome to DirectMon

                                             Based on Scientific Linux 5
                                             Cluster Manager ID: #999915

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

To set up your cluster, type

  ddn-setup

and follow the instructions

Creating DSA key for ssh
[root@headnode ~]#
```

Running the `ddn-setup` script goes through several screens, some of which prompt for input. At some prompts, it is hinted that typing "`I`" gives further explanation about the input.

The screens go through the following issues:

- The license agreement.
- Amazon Web Services account information. This asks for the AWS Username, AWS Account ID, Access Key ID, and Secret Access Key. These are needed by DirectMon to manage the cloud node instances.
- The installation of the DDN product key (formatted like 868868-797099-979199-091301-134414). This is the cloud version of the `request-license` command in section 4.1.3, and asks for:
  - The organization information for the license. This requires input for the fields: country, state, locality, organizational unit, unit and cluster name.
  - The values to be used for the head node machine name and its administrative password (used for the root and MySQL passwords).

- Optionally, setting up the secondary drives using Amazon's EBS service.
- Optionally, setting up extra storage for `/home` using Amazon's EBS service.
- Optionally, setting up extra storage for monitoring data (recommended for more than 500 nodes).
- Setting up cloud node instance types. Amazon instance types (`http://aws.amazon.com/ec2/instance-types/`) are choices presented from node specifications consisting of memory, storage, cores, GPUs, and others. The setting up of the instance type step is looped through if necessary to allow more than one instance type to be configured.
  - Setting up
    * the number <*N*> of cloud compute nodes and
    * their base name (the `cnode` part of the name if the nodes have the names `cnode001` to `cnode`<*N*>).
  - Setting up the amount of storage for the cloud compute node instance.

    The default disk partitioning layout for nodes can be modified as described in Appendix D. Using diskless nodes is also possible if the cloud compute node instance has enough RAM—about 2GB at the time of writing.

  Setting these values causes the cloud node objects to be created in the CMDaemon database. Cloud nodes are not however actually started up at this stage. Starting up must be done explicitly, and is covered in section 7.1.4.
- Setting up the workload manager, along with the number of slots, and if the head node is to be used for compute jobs too.

After accepting the input, the `ddn-setup` script runs through the configuration of the head node and the cloud nodes. Its progress is indicated by output similar to the following:

Example

```
[root@headnode ~]# ddn-setup
Retrieving Amazon instance information

-------------------------------------------------------------------------
|                             License agreements                        |
-------------------------------------------------------------------------
The end user license agreement will be shown in a new screen. To exit
this screen, type 'q'. Press any key to continue
Do you agree with the license terms and conditions? (yes, no, show): yes

-------------------------------------------------------------------------
|                         Amazon account information                    |
-------------------------------------------------------------------------
AWS username (I for information): exampleuser@ddn.com
AWS Account ID (I for information): 313234312222
Access Key ID (I for information): OUPOUASOUDSSSAOU
Secret Access Key (I for information): Aighei8EooLi1Dae8Nio5ohl4ieXiAiaiV

Verifying Amazon credentials

-------------------------------------------------------------------------
|                              DDN License             |
-------------------------------------------------------------------------
DDN Product Key (I for information):
                                       423112-231432-134234-132423-134221
   Country: US
   State: CA
   Locality: San Francisco
   Organization name: DDN
   Organizational Unit: Development
   Cluster Name: demo-cluster

-------------------------------------------------------------------------
|                                Head Node                              |
-------------------------------------------------------------------------
Hostname: ddnmon61
Administrative Password:
Verify password:
Do you want to create a second drive for more storage capacity?
(I for information) [YES|no] no
Extra storage for /home (I for information)? [NO|yes] n
Extra storage for monitoring data (I for information)? [NO|yes] n

-------------------------------------------------------------------------
|                              Compute Nodes                            |
-------------------------------------------------------------------------
Instance Type (I for information):
  m1.small
  m1.medium
  c1.medium
  m1.large
  t1.micro
  m2.xlarge
  m2.2xlarge
  m2.4xlarge
  c1.xlarge
  [t1.micro]
>
t1.micro
Node Count [2]:
2
Base name (I for information) [cnode]:
cnode
Instances of type t1.micro need to use EBS storage
Size of EBS (GB) [40]: 15
Do you want to configure more node types? [NO|yes]
no

-------------------------------------------------------------------------
```

```
|                        Workload Management system                       |
-------------------------------------------------------------------------
Which workload management system do you want to use? (I for information)?
  slurm
  sge
  torque
  [slurm]
>
slurm
Number of slots per node [8]:
8
Do you want to use the head node for compute jobs? [NO|yes]
no

The following information will be used to configure this head node:

  Amazon information:
    AWS username:             exampleuser@ddn.com
    AWS Account ID:           313234312222
    Access Key ID:            OUPOUASOUDSSSAOU
    Secret Access Key:        Aighei8EooLi1Dae8Nio5ohl4ieXiAin5eeRoaiV
  DDN Product Key:         423112-231432-134234-132423-134221
  License information
    Country:                  US
    State:                    CA
    Locality:                 San Francisco
    Organization name:        DDN
    Organizational Unit:      Development
    Cluster Name:             demo-cluster
  Hostname:                   ddnmon61
  Second drive size:          no
  Instance Type:              t1.micro
  Node Count:                 2
  Base name:                  cnode
  Storage type:               EBS
  Size of storage:            15 GB
  Workload management system: slurm
  Number of slots:            8
  Head node for compute jobs: no

The information to configure this head node has been collected, and shown
above. The next phase will be to process this information. A new DDN
license will be installed, the DirectMon software will be
initialized and the workload management software will be initialized
Do you want to continue? [YES|no]
yes

Starting to configure this head node
Successfully retrieved the license
Installed license
Initializing DirectMon
Installing admin certificates
Configuring default scheduler, slurm
Set up finished
```

It is recommended that the system administrator log out and login again after the script has been run, in order to enable the new environment for the shell that the administrator is in. If the hostname was changed in the `ddn-setup` script, for example, the name change shows up in the shell prompt only after the re-login.

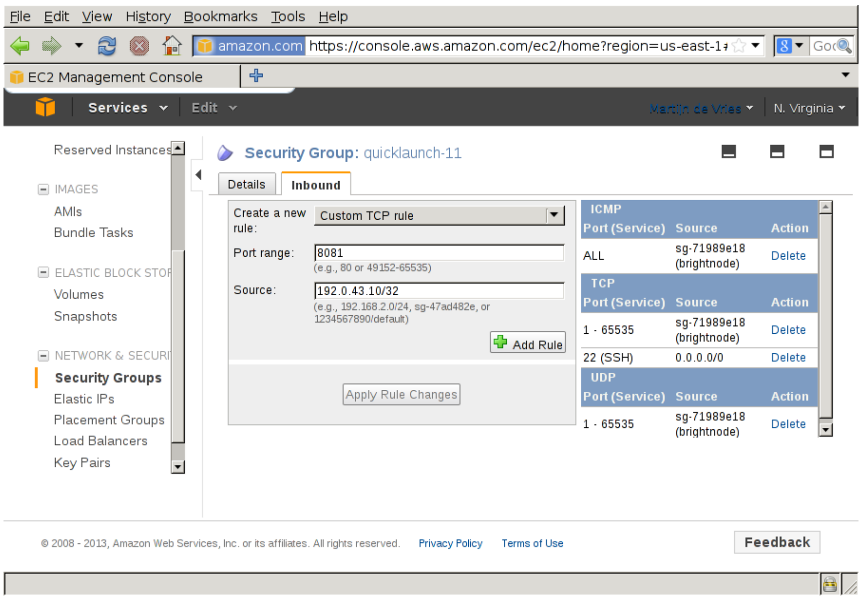
Once there is a head in the cloud, the other cloud nodes can be started up.

7.1.3 Cluster-On-Demand: Connecting To The headnode Via `cmsh` or `cmgui`

Amazon provides a security group to each instance. By default, this configures network access so that only inbound SSH connections are allowed from outside the cloud. A new security group can be configured, or an existing one modified, using the `Edit details` button in figure 7.11. Security groups can also be accessed from the navigation menu on the left side of the EC2 Management Console.

Cluster-On-Demand: Access With A Remote, Standalone `cmgui`

The security group defined by Amazon for the head node can be modified by the administrator to allow remote connections to CMDaemon running on the head node (figure 7.11).



Security Group: quicklaunch-11

Create a new rule: Custom TCP rule

Port range: 8081
(e.g., 80 or 49152-65535)

Source: 192.0.43.10/32
(e.g., 192.168.2.0/24, sg-47ad482e, or 1234567890/default)

Port (Service)	Source	Action
ICMP		
ALL	sg-71989e18 (brightnode)	Delete
TCP		
1 - 65535	sg-71989e18 (brightnode)	Delete
22 (SSH)	0.0.0.0/0	Delete
UDP		
1 - 65535	sg-71989e18 (brightnode)	Delete

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 7.11: Security Group Network And Port Access Restriction

- To allow only a specific network block to access the instance, the network from which remote connections are allowed can be specified in CIDR format.

- Explicitly allowing inbound connections to port 8081 on the head node allows the standalone `cmgui` (section 3.4) to connect to the head node. This is because the `cmgui` back end, which is CMDaemon, communicates via port 8081.

Cluster-On-Demand: Access With A Local `cmsh`

The security group created by Amazon by default already allows inbound SSH connections from outside the cloud to the instance running in the cloud, even if the incoming port 8081 is blocked. Launching a `cmsh` session within an SSH connection running to the head node is therefore possible, and works well.

Cluster-On-Demand: Access With A Local `cmgui`

It is possible to run an X-forwarded `cmgui` session from within an `ssh -X` connection that is already running to the head node. However, it suffers from significant X-protocol lag due to the various network encapsulation layers involved. The procedure described earlier for cluster-on-demand access with the remote, standalone `cmgui` from outside the cloud is therefore recommended instead for a more pleasant experience.

7.1.4 Cluster-On-Demand: Cloud Node Start-up

Cloud nodes must be explicitly started up. This is done by powering them up, assuming the associated cloud node objects exist. The cloud node objects are typically specified in the `ddn-setup` script—in the preceding example the cloud node objects are `cnode001` and `cnode002`.

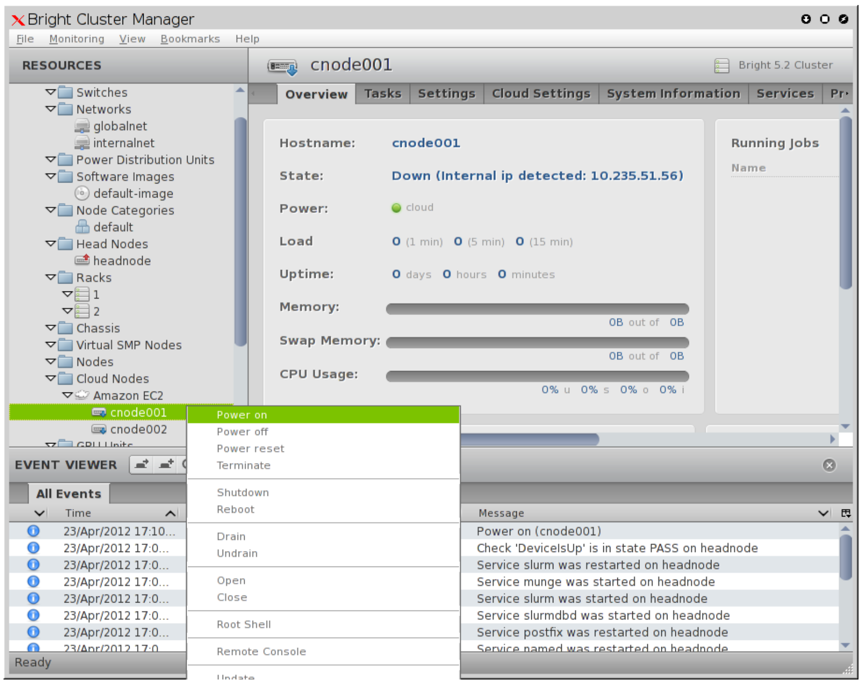
However, more cloud node objects can be created if needed after the `ddn-setup` script has run. The maximum number that may be created is set by the license purchased.

Large numbers of cloud node objects can be created with DirectMon as follows:

- In `cmgui` they are conveniently created with the Node Creation Wizard as described in section 7.2.3. Several of the steps described in that section are specific to Cluster Extension clusters. These steps are not needed for for Cluster-On-Demand clusters, and therefore do not come up when the wizard is used in this case.

- In `cmsh` a large number of cloud node objects can conveniently be created with the "`foreach --clone`" command instead, as described in section 7.3.3.

After creation, individual cloud nodes can be powered up from within `cmgui` by a right-click on the cloud node resource item (figure 7.12).



Hostname: cnode001
State: Down (Internal ip detected: 10.235.51.56)
Power: cloud
Load 0 (1 min) 0 (5 min) 0 (15 min)
Uptime: 0 days 0 hours 0 minutes
Memory: 0B out of 0B
Swap Memory: 0B out of 0B
CPU Usage: 0% u 0% s 0% o 0% i

Time	Message
23/Apr/2012 17:10...	Power on (cnode001)
23/Apr/2012 17:0...	Check 'DeviceIsUp' is in state PASS on headnode
23/Apr/2012 17:0...	Service slurm was restarted on headnode
23/Apr/2012 17:0...	Service munge was started on headnode
23/Apr/2012 17:0...	Service slurm was started on headnode
23/Apr/2012 17:0...	Service slurmdbd was started on headnode
23/Apr/2012 17:0...	Service postfix was restarted on headnode
23/Apr/2012 17:0	Service named was restarted on headnode

Figure 7.12: Powering on a cloud node with `cmgui`

As with regular non-cloud nodes, multiple cloud nodes can be powered up in `cmgui` by selecting them from the `Overview` tabbed pane. Switching to the `Tasks` tabbed pane and clicking on the power on button then powers them up.

As with regular non-cloud nodes, cloud nodes can also be powered up from within the `device` mode of `cmsh`. The initial power status (section 5.1) of cloud nodes is `FAILED`, because they cannot be communicated with. As they start up, their power status changes to `OFF`, and then to `ON`. Some time after that they are connected to the cluster and ready for use. The device status (as opposed to the power status) remains `DOWN` until it is ready for use, at which point it switches to `UP`:

Example

```
[head1->device]% power status
cloud ................. [  FAILED ] cnode001 (Cloud instance ID not set)
cloud ................. [  FAILED ] cnode002 (Cloud instance ID not set)
No power control ...... [ UNKNOWN ] head1
[head1->device]% power on -n cnode001
cloud ................. [   ON    ] cnode001
[head1->device]% power status
cloud ................. [   OFF   ] cnode001 (pending)
cloud ................. [  FAILED ] cnode002 (Cloud instance ID not set)
No power control ...... [ UNKNOWN ] head1
[head1->device]% power on -n cnode002
cloud ................. [   ON    ] cnode002
[head1->device]% power status
cloud ................. [   ON    ] cnode001 (running)
cloud ................. [   OFF   ] cnode002 (pending)
No power control ...... [ UNKNOWN ] head1
[head1->device]% !ping -c1 cnode001
```

```
ping: unknown host cnode001
[head1->device]% status
head1 .................... [   UP   ]
node001 .................. [   UP   ]
node002 .................. [  DOWN  ]
[head1->device]% !ping -c1 cnode001
PING cnode001.cm.cluster (10.234.226.155) 56(84) bytes of data.
64 bytes from cnode001.cm.cluster (10.234.226.155): icmp_seq=1 ttl=63 t\
ime=3.94 ms
```

Multiple cloud nodes can be powered up at a time in `cmsh` with the "`power on`" command using ranges and other options (section 5.2.2).

IP Addresses In The Cluster-On-Demand Cloud

- The IP addresses assigned to cloud nodes on powering them up are arbitrarily scattered over the 10.0.0.0/8 network and its subnets
  - No pattern should therefore be relied upon in the addressing scheme of cloud nodes
- Shutting down and starting up head and regular cloud nodes can cause their IP address to change.
  - However, DirectMon managing the nodes means that a regular cloud node re-establishes its connection to the cluster when it comes up, and will have the same node name as before.

7.2 Cluster Extension Cloudbursting

Cluster Extension cloudbursting ("hybrid" cloudbursting) in DirectMon is the case when a cloud service provider is used to provide nodes that are in the cloud as an extension to the number of regular nodes in a cluster. The head node in a Cluster Extension configuration is always outside the cloud, and there may be some regular nodes that are outside the cloud too.

Requirements Cluster Extension cloudbursting requires:

- **An activated cluster license.**

  Some administrators skip on ahead to try out cloudbursting right away in a Cluster Extension configuration, without having made the license active earlier on. That will not work.

  If activation is indeed needed, then it is most likely a case of simply running the `request-license` command with the product key. Further details on activating the license are in section 4.1.

- **Registration of the product key.**

  The product key must also be registered on the DDN Customer Portal website at `http://www.ddn.com/Customer-Login.php`. A Customer Portal account is needed to do this.

  The product key is submitted at the Customer Portal website specifically for a Cluster Extension setup, from the `Burst!` menu. The customer portal account is then automatically associated with the license installed (section 7.1.2) on the head node. The key is also needed to activate the cluster license, if that has not been done before.

- **An Amazon account**, if the cloud provider is Amazon.
- **An open UDP port.**

  By default, this is port 1194. It is used for the OpenVPN connection from the head node to the cloud and back. To use TCP, and/or ports other than 1194, the DDN knowledgebase at `http://kb.ddn.com` can be consulted using the keyword "openvpn".

  Outbound ssh access from the head node is also useful, but not strictly required.

  By default, Shorewall as provided by DirectMon on the head node is configured to allow all outbound connections, but other firewalls may need to be considered too.

Steps Cluster Extension cloudbursting uses a *cloud director*. A cloud director is a specially connected cloud node used to manage regular cloud nodes, and is described more thoroughly in section 7.2.2. Assuming the administrator has ownership of a cloud provider account, the following steps can be followed to launch Cluster Extension cloud nodes:

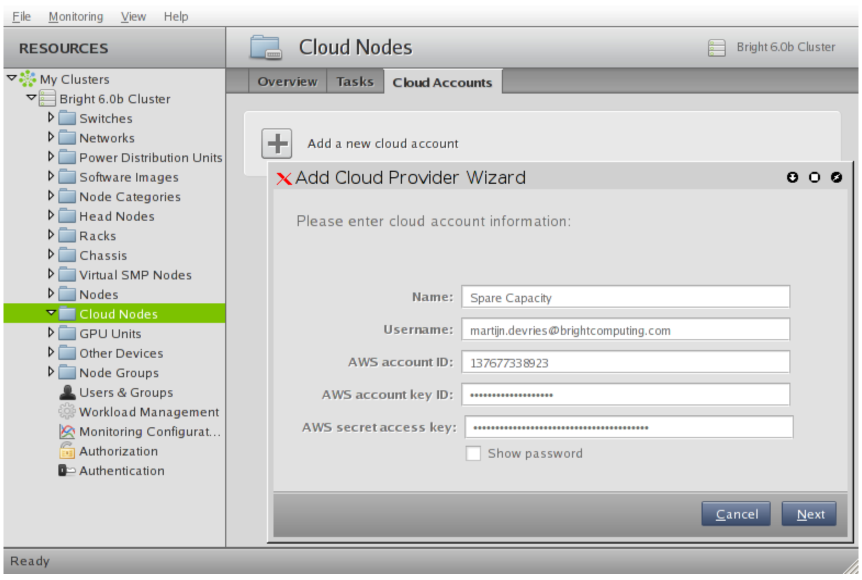
1. The cloud provider is logged into from `cmgui`, and a cloud director is configured (section 7.2.1).
2. The cloud director is started up (section 7.2.2).
3. The cloud nodes are provisioned from the cloud director (section 7.2.3).

The cloud nodes then become available for general use by the cluster.

7.2.1 Cluster Extension: Cloud Provider Login And Cloud Director Configuration

To access the Amazon cloud service from `cmgui`, the "`Cloud Nodes`" resource is selected, and the "`Cloud Accounts`" tabbed pane opened. This allows a cloud provider account to be edited or deleted from the available ones already there.

It also allows a new cloud account provider to be added and configured. This is done by clicking on the ⊞ button beside the text "`Add a new cloud account`", which opens up the "`Add Cloud Provider Wizard`" window (figure 7.13).



Please enter cloud account information:

Name: Spare Capacity
Username: martijn.devries@brightcomputing.com
AWS account ID: 137677338923
AWS account key ID: ••••••••••••••••••••
AWS secret access key: ••
Show password

Figure 7.13: Logging Into A Cloud Provider With cmgui

In the first screen, the cloud account subscription information is added. The subscription information could be from Amazon or from another supported provider.

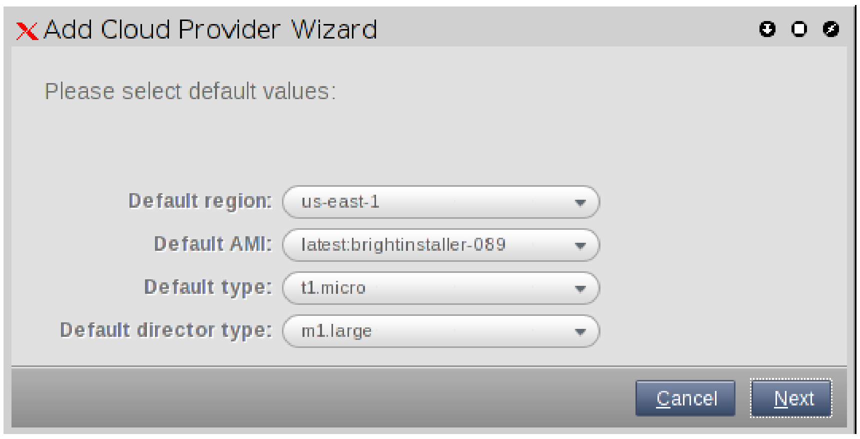
In the case of Amazon, the information is obtainable after signing up for Amazon Web Services (AWS) at http://aws.amazon.com. After sign-up, the Access Identifiers section of the subscription, at http://aws-portal.amazon.com/gp/aws/developer/account/index.html?action=access-key, provides the required information. If that URL does not work, then the Amazon documentation at http://docs.amazonwebservices.com/fws/latest/GettingStartedGuide/index.html?AWSCredentials.html can be followed instead.

For Amazon, the fields to enter in the wizard are:

- The Name: A convenient, arbitrary value.
- The Username: The e-mail address associated with the AWS account.
- The AWS account ID: The AWS Account ID.
- The AWS account key ID: The AWS Access Key ID.
- The AWS secret access key ID: The AWS Secret Access Key.

The "show password" checkbox toggles the visibility of the sensitive input. Clicking the Next button submits the details, and inputs for the next screen are retrieved from Amazon.

The next screen (figure 7.14) displays options for the Amazon cloud service.


Add Cloud Provider Wizard

Please select default values:

Default region: us-east-1

Default AMI: latest:brightinstaller-089

Default type: t1.micro

Default director type: m1.large

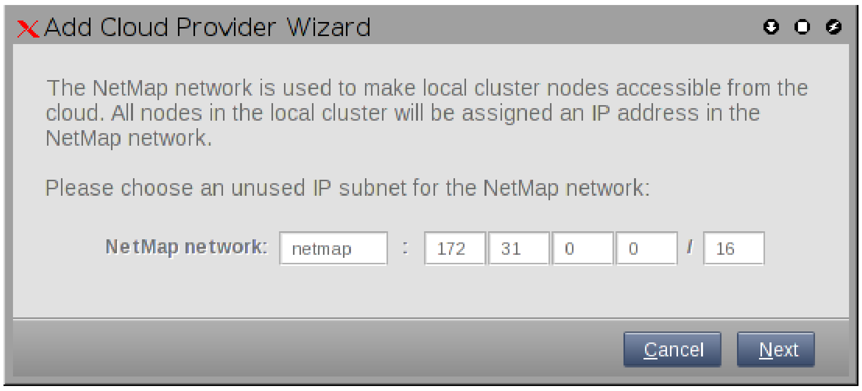
Figure 7.14: Selecting Options At The Cloud Provider With `cmgui`

In figure 7.14, the following options are shown:

- `Default region`: These are regions from which the service can be provided. Amazon, for example, offers a choice out of capacity on the East Coast of the USA, Western Europe, the Asia Pacific region and others.
- `Default AMI`: This is the Amazon Machine Instance image that DDN provides. The node-installer from this image installs the cloud director and cloud nodes.
- `Default type`: A choice out of a selection of possible virtual machine types (`http://aws.amazon.com/ec2/instance-types/`) made available by Amazon for the cloud node. The choices presented are from node specifications consisting of memory, storage, cores, GPUs, and others. In `cmsh`, running `cmsh -c "cloud types"` also shows the types available.
- `Default director type`: A choice for the cloud director node, made from a selection of possible virtual machine types made available by Amazon. This virtual machine type usually needs to be more powerful than a regular cloud node, and is by default set to `m1.large`.

The default settings are normally a good choice. On clicking the `Next` button, the choices are processed.

The next screen (figure 7.15) displays the NetMap network name and addressing scheme. This is a network mapping that assigns extra IP addresses to local nodes to make them accessible from the cloud. The addressing scheme can be changed if needed to another unused subnet. By default it uses 172.30.0.0/16.



The NetMap network is used to make local cluster nodes accessible from the cloud. All nodes in the local cluster will be assigned an IP address in the NetMap network.

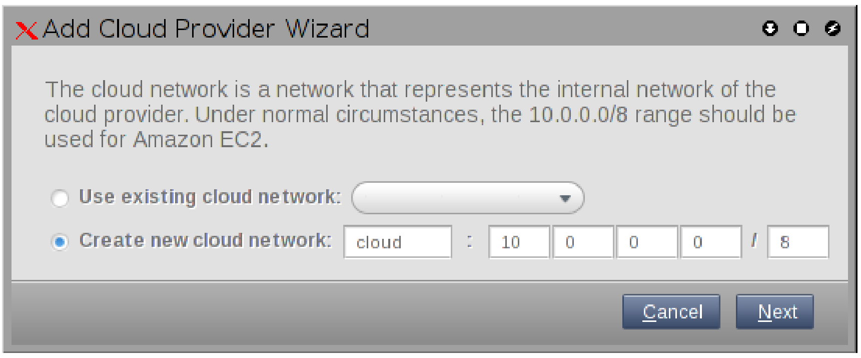
Please choose an unused IP subnet for the NetMap network:

NetMap network: netmap : 172 31 0 0 / 16

Figure 7.15: Setting The NetMap Network With `cmgui`

The default values are normally a good choice. On clicking the `Next` button, the values are processed.

The next screen (figure 7.16) displays the cloud network name and addressing scheme. This can be changed if needed, but for Amazon the 10.0.0.0/8 range is expected.



The cloud network is a network that represents the internal network of the cloud provider. Under normal circumstances, the 10.0.0.0/8 range should be used for Amazon EC2.

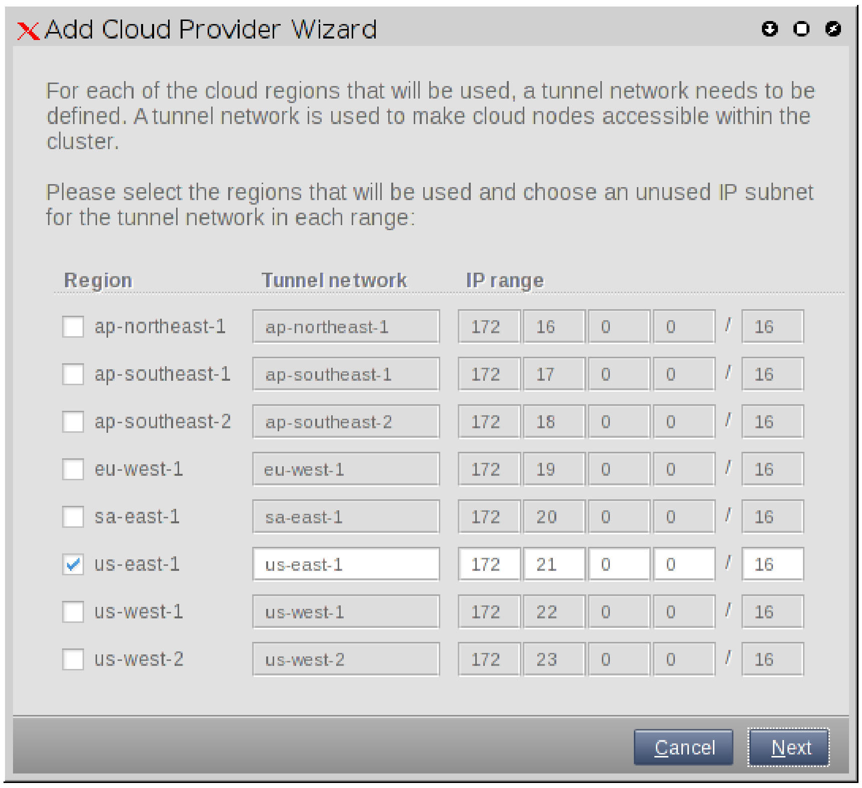
Use existing cloud network:

[✓ Create new cloud network:] cloud : 10 0 0 0 / 8

Figure 7.16: Setting The Cloud Network At The Cloud Provider With `cmgui`

On clicking the `Next` button, the configuration is processed.

The next screen (figure 7.17) displays a proposed DirectMon tunnel network naming and addressing scheme for each checkboxed cloud region. These can be changed if needed from the suggested defaults. For Amazon the `us-east-1` region shown in the figure has a default tunnel network value of 172.21.0.0/16.


Add Cloud Provider Wizard

For each of the cloud regions that will be used, a tunnel network needs to be defined. A tunnel network is used to make cloud nodes accessible within the cluster.

Please select the regions that will be used and choose an unused IP subnet for the tunnel network in each range:

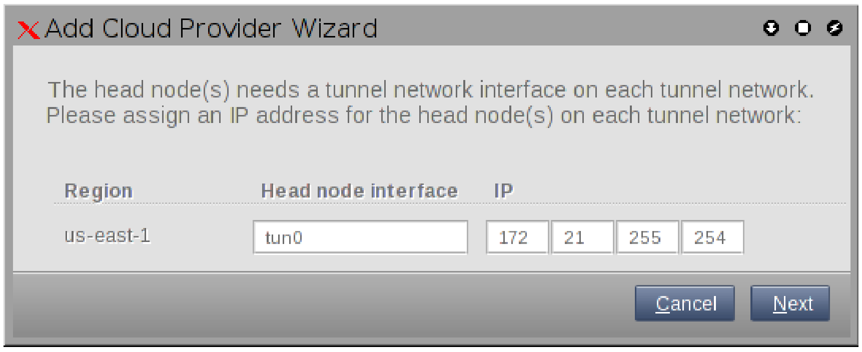
	Region	Tunnel network	IP range					
☐	ap-northeast-1	ap-northeast-1	172	16	0	0	/	16
☐	ap-southeast-1	ap-southeast-1	172	17	0	0	/	16
☐	ap-southeast-2	ap-southeast-2	172	18	0	0	/	16
☐	eu-west-1	eu-west-1	172	19	0	0	/	16
☐	sa-east-1	sa-east-1	172	20	0	0	/	16
✓	us-east-1	us-east-1	172	21	0	0	/	16
☐	us-west-1	us-west-1	172	22	0	0	/	16
☐	us-west-2	us-west-2	172	23	0	0	/	16

Figure 7.17: Setting The Tunnel Network For Regions With `cmgui`

On clicking the `Next` button, the configuration is processed.

The next screen (figure 7.18) displays a proposed DirectMon tunnel interface name and IP address for the head node(s). A tunnel interface is defined for the head node for each tunnel network. By default, the address ending in .255.254 is used, and appended to the first two parts of the dotted quad (for example, 172.21 for us-east-1), so that the suggested default IP address in this case becomes 172.21.255.254. The default suggested device name is tun0.

These can be changed if needed from the suggested defaults.


Add Cloud Provider Wizard

The head node(s) needs a tunnel network interface on each tunnel network. Please assign an IP address for the head node(s) on each tunnel network:

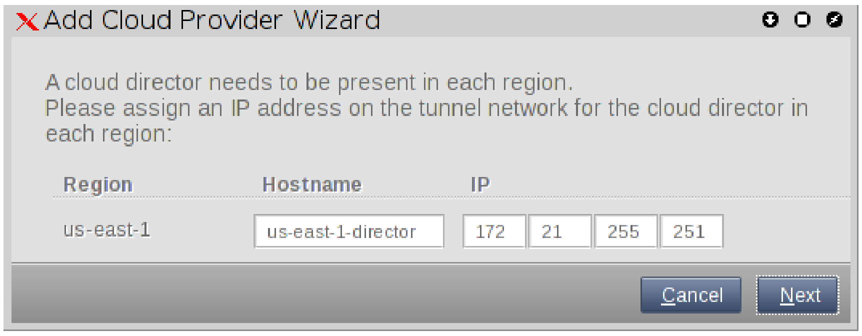
Region	Head node interface	IP			
us-east-1	tun0	172	21	255	254

Figure 7.18: Setting The Tunnel Network Interface For The Head Node(s) With `cmgui`

On clicking the `Next` button, the configuration is processed.

The next screen (figure 7.19) displays a proposed DirectMon hostname and tunnel IP address for the cloud director node(s). By default, the suggested hostname is the region name with `-director` as the suffix. For example, `us-east1-director` for the region `us-east1`. By default, an address ending in .255.251 is suggested for appending to the first two parts of the dotted quad (for example, the prefix 172.21 for us-east-1), so that the suggested default IP address in this case becomes 172.21.255.251. The addresses ending in 252 and 253 may be required by head nodes that implement failover (section 15).

These can be changed if needed from the suggested defaults, but should be consistent with the network address.


Add Cloud Provider Wizard

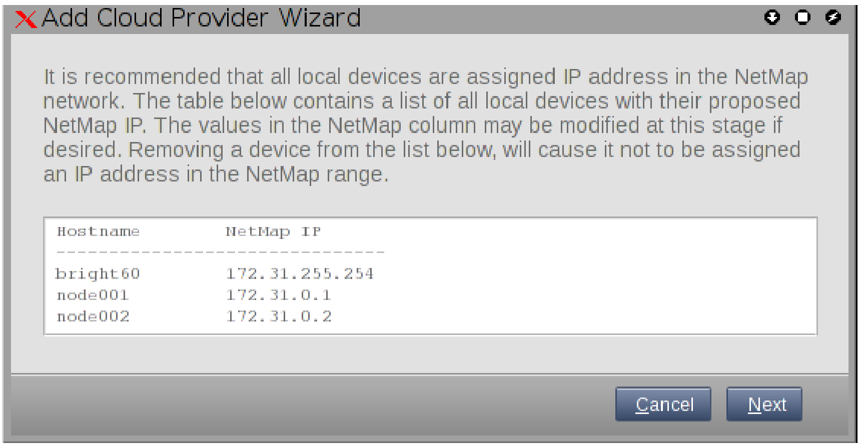
A cloud director needs to be present in each region.
Please assign an IP address on the tunnel network for the cloud director in each region:

Region	Hostname	IP
us-east-1	us-east-1-director	172 21 255 251

Figure 7.19: Setting The Tunnel Network Interface For The Cloud Director(s) With `cmgui`

On clicking the `Next` button, the configuration is processed.

The next screen (figure 7.20) displays the proposed assignment of IP addresses in the NetMap network. These can be changed from the suggested defaults, but should be consistent with the addressing schemes already defined.


Add Cloud Provider Wizard

It is recommended that all local devices are assigned IP address in the NetMap network. The table below contains a list of all local devices with their proposed NetMap IP. The values in the NetMap column may be modified at this stage if desired. Removing a device from the list below, will cause it not to be assigned an IP address in the NetMap range.

```
Hostname        NetMap IP
------------------------------
bright60        172.31.255.254
node001         172.31.0.1
node002         172.31.0.2
```

Figure 7.20: Setting The Tunnel Network Interface IP Addresses With `cmgui`

On clicking the `Next` button, the configuration is processed.

It should be noted that the default suggested NetMap, cloud network, and cloud region addresses configured by the wizard are all compliant with RFC1918 private network addressing, and are thus not public IP addresses.

If all is well, the configuration is successfully processed. A message is then displayed indicating that the cloud provider service has been added to the existing cluster and configured successfully, and that the wizard is finished with its job.

No nodes are activated yet within the cloud provider service. To start them up, the components of the cloud provider service must be started up by

- powering up the cloud directors (section 7.2.2)
- powering on the cloud nodes after the cloud directors are up. Often this involves creating new cloud nodes by using the "`Create Cloud Nodes`" wizard (section 7.2.3).

7.2.2 Cluster Extension: Cloud Director Start-up

The cloud director can take some time to start up the first time. The bottleneck is usually due to several provisioning stages, where the bandwidth between the head node and the cloud director means that the provisioning runs typically take tens of minutes to complete.

This bottleneck is one of the reasons why the cloud director is put in the cloud in the first place. The next time the cloud director powers up, and assuming persistent storage is used—as is the default—the cloud director runs through the provisioning stages much faster, and completes within a few minutes.

The cloud director acts as a helper instance in the cloud, providing some of the functions of the head node within the cloud in order to speed up communications and ensure greater resource efficiency. Amongst the functions it provides are:

- Cloud nodes provisioning
- Exporting a copy of the shared directory `/cm/shared` to the cloud nodes so that they can mount it
- Providing routing services using an OpenVPN server. While cloud nodes within a region communicate directly with each other, cloud nodes in one region use the OpenVPN server of their cloud director to communicate with the other cloud regions and to communicate with the head node of the cluster.

Cloud directors are not regular nodes, so they have their own category, `cloud-director`, into which they are placed by default.

The cloud-related properties of the cloud director are displayed in the "`Cloud Settings`" tab of the Cloud Nodes director item.

The cloud director can be started up in `cmgui` by right-clicking on the cloud director item from the `Cloud Nodes` resource, and selecting `Power on` from the menu. Any cloud settings options that have been set are frozen as the instance starts up, until the instance terminates.

Setting The Cloud Director Disk Storage Device Type

Amazon provides two kinds of storage types as part of EC2:

1. **Instance storage, using ephemeral devices:** Instance storage is not provided for the following instance types:
    - `t1.micro`

- m3.xlarge
- m3.2xlarge
- cr1.8xlarge

However, Amazon by default currently provides 150GB or more of instance storage for all other instance types.

Details on instance storage can be found at http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/InstanceStorage.html#StorageOnInstanceTypes.
Ephemeral means that the device is temporary, which means that whatever is placed on it is lost on reboot.

2. **Elastic Block Storage (EBS) volumes:** Normally, EBS is suggested for cloud director and cloud node use. The reasons for this include:
   - it can be provided to all nodes in the same availability zone
   - unlike instance storage, EBS remains available for use when an instance using it reboots
   - instance storage is not available for some instances types such as t1.micro.

Using the ephemeral device as the drive for the cloud director: Since the cloud provider instance type is essential, and contains so much data, it is rarely wise to use ephemeral devices as the drive for the cloud provider.

However, if for some reason the administrator would like to avoid using EBS, and use the instance storage, then this can be done by removing the default EBS volume suggestion for the cloud director provided by DirectMon. When doing this, the ephemeral device that is used as the replacement must be renamed. It must take over the name that the EBS volume device had before it was removed.

- In cmgui, this can be done in the "Cloud Settings" tab of the Cloud Nodes director item.
- In cmsh, this can be done in device mode, by going into the cloudsettings submode for the cloud director, and then going a level deeper into the storage submode. Within the storage submode, the list command shows the values of the storage devices associated with the cloud director. The values can be modified as required with the usual object commands. The set command can be used to modify the values.

  Example

```
[ddnmon61]% device use us-east-1-director
[ddnmon61->device[us-east-1-director]]% cloudsettings
[ddnmon61->device[us-east-1-director]->cloudsettings]% storage
[ddnmon61->...->cloudsettings->storage]% list
Type       Name (key)   Drive     Size    Volume ID
---------- ------------ --------- ------- ---------
ebs        ebs          sdb       40GB
ephemeral  ephemeral0   sdb       0B
```

```
[ddnmon61->...->cloudsettings->storage]% remove ebs
[ddnmon61->...->cloudsettings*->storage*]% set ephemeral0 drive sdb
[ddnmon61->...->cloudsettings*->storage*]% list
Type       Name (key)   Drive     Size    Volume ID
---------- ------------ --------- -------- ---------
ephemeral  ephemeral0   sdb       0B
[ddnmon61->...->cloudsettings*->storage*]% commit
```

Setting The Cloud Director Disk Size

The disk size for the cloud director can be set with `cmgui` in the `Cloud Settings` tab.

By default, an EBS volume size of 40GB is suggested. This is as for a standard node layout (section D.2), and no use is then made of the ephemeral device.

40GB on its own is unlikely to be enough for most purposes other than running basic `hello world` tests. In actual use, the most important considerations are likely to be that the cloud director should have enough space for:

- the user home directories (under `/home/`)
- the cluster manager shared directory contents, (under `/cm/shared/`)
- the software image directories (under `/cm/images/`)

The cluster administrator should therefore properly consider the allocation of space, and decide if the disk layout should be modified. An example of how to access the disk setup XML file to modify the disk layout is given in section 4.10.3.

For the cloud director, an additional sensible option may be to place `/tmp` and the swap space on an ephemeral device, by appropriately modifying the XML layout for the cloud director.

Tracking Cloud Director Start-up

Tracking cloud director start-up from the EC2 management console: the boot progress of the cloud director can be followed by watching the status of the instance in the Amazon EC2 management console, as illustrated in figure 7.8. The `Instance ID` that is used to identify the instance can be found

- with `cmgui`, within the `Cloud Settings` tab for the cloud director node
- with `cmsh`, by running something like:

  Example

  ```
  [ddnmon61]% device cloudsettings us-east-1-director
  [ddnmon61->device[us-east-1-director]]% get instanceid
  ```

Tracking cloud director start-up from `cmgui`: the boot progress of the cloud director can also be followed by

- watching the icon changes (as in section 6.5.1)
- watching the `State` in the `Overview` tabbed window
- watching the `Console log` from the `Tasks` tabbed window

Tracking cloud director start-up from the bash shell of the head node: there are some further possibilities to view the progress of the cloud director after it has reached at least the initrd stage. These possibilities include:

- an SSH connection to the cloud director can be made during the pre-init, initrd stage, after the cloud director system has been set up via an rsync. This allows a login to the node-installer shell.
- an SSH connection to the cloud director can be also be made after the initrd stage has ended, after the init process runs making an SSH daemon available again. This allows a login on the cloud director when it is fully up.

During the initrd stage, the cloud director is provisioned first. The cloud node image(s) and shared directory are then provisioned on the cloud director, still within the initrd stage. To see what rsync is supplying to the cloud director, the command "`ps uww -C rsync`" can be run on the head node. Its output can then be parsed to make obvious the source and target directories currently being transferred:

Example

```
[root@ddnmon61 ~]# ps uww -C rsync | cut -f11- -d" " #11th part onwards
/cm/shared/ syncer@172.22.255.251::target//cm/shared/
```

Tracking cloud director start-up from `cmsh`: the `provisioningstatus` command in `cmsh` can be used to view the provisioning status (some output elided):

Example

```
[root@ddnmon61 ~]# cmsh -c "softwareimage provisioningstatus"
...
+ us-east-1-director
...
  Up to date images:         none
  Out of date images:        default-image
```

In the preceding output, the absence of an entry for "`Up to date images`" shows that the cloud director does not yet have an image that it can provision to the cloud nodes. After some time, the last few lines of output should change to something like:

Example

```
+ us-east-1-director
...
  Up to date images:         default-image
```

This indicates the image for the cloud nodes is now ready.

With the `-a` option, the `provisioningstatus -a` command gives details that may be helpful. For example, while the cloud director is having the default software image placed on it for provisioning purposes, the source and destination paths are `/cm/images/default-image`:

Example

```
[root@ddnmon61 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):         4
Source node:           ddnmon61
Source path:           /cm/images/default-image
Destination node:      us-east-1-director
Destination path:      /cm/images/default-image
...
```

After some time, when the shared filesystem is being provisioned, the source and destination paths should change to the `/cm/shared` directory:

```
[root@ddnmon61 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):         5
Source node:           ddnmon61
Source path:           /cm/shared
Destination node:      us-east-1-director
Destination path:      /cm/shared
...
```

After the shared directory and the cloud node software images are provisioned, the cloud director is fully up. Cloud node instances can then be powered up and provisioned from the cloud director.

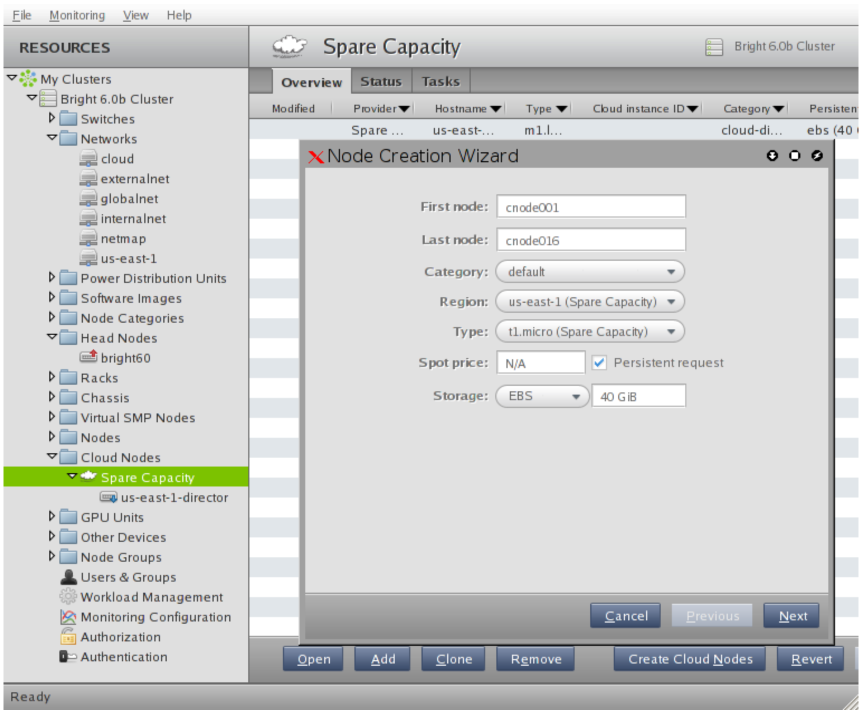
7.2.3 Cluster Extension: Cloud Node Start-up

The "`Create Cloud Nodes`" wizard button in `cmgui` conveniently creates cloud node objects. The wizard is accessed from within the "`Cloud Nodes`" resource, by selecting the provider item, and then choosing the `Overview` tab. Cloud node objects can also be created in `cmsh` as described in section 7.3.3.

A working cloud director is not needed to configure the regular cloud nodes. However the cloud director must be up, and the associated networks to the regular cloud nodes and to the head node must be configured correctly, in order for the regular cloud nodes to boot up properly. If needed, additional cloud provisioning nodes (section 6.2) can be configured by assigning the provisioning role to cloud nodes, along with appropriate nodegroups (page 189) values, in order to create a provisioning hierarchy.

By default, the first screen of the wizard (figure 7.21) allows the administrator to do the following:

- The first regular cloud node and last regular cloud node can be set. By default, 16 regular cloud nodes are suggested. The names of the nodes have a prefix of `cnode` by default, and end in three digit numbers, for example `cnode001`, `cnode002` and so on.
- The category can be set for these nodes. By default it is set to the `default` category
- The region for the regular cloud nodes can be set. By default it matches the cloud director region.
- The regular cloud node instance type can be set. By default, `t1.micro` is chosen.



First node: cnode001
Last node: cnode016
Category: default
Region: us-east-1 (Spare Capacity)
Type: t1.micro (Spare Capacity)
Spot price: N/A ✓ Persistent request
Storage: EBS 40 GiB

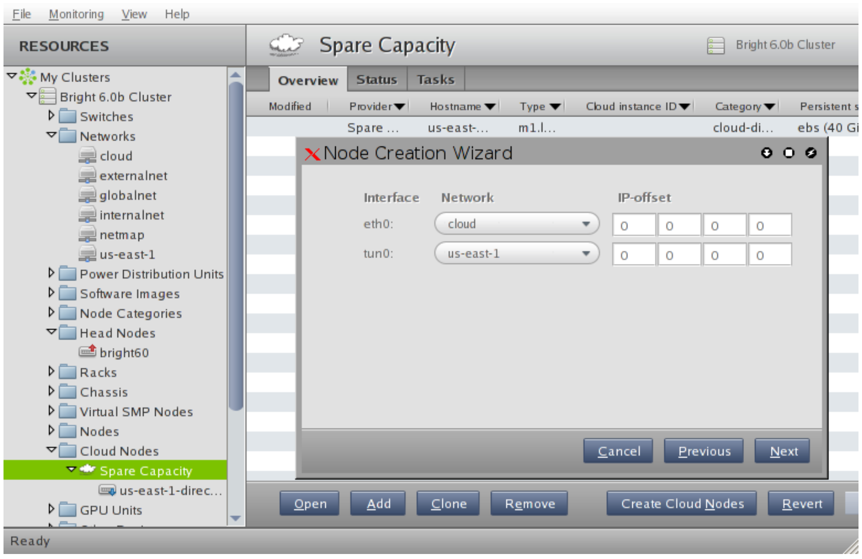
Figure 7.21: Main Cloud Node Creation Wizard Configuration Screen

- A spot price (section 7.4.3) can be set in this screen to take advantage of cheaper pricing to launch regular cloud nodes. By default, no spot price is set.
- The storage type and size used can be set. By default, it is EBS, and 40GB. If the `t1.micro` instance type has been chosen, then there is no ephemeral device storage available, in accordance with Amazon policies.

The next screen of the wizard (figure 7.22) applies to the region chosen in the previous screen (in figure 7.21 the region is `us-east-1`). Within the region, IP offsets (footnote on page 18) can be set:

- for nodes in the associated cloud network
- for nodes in the associated tunnel network

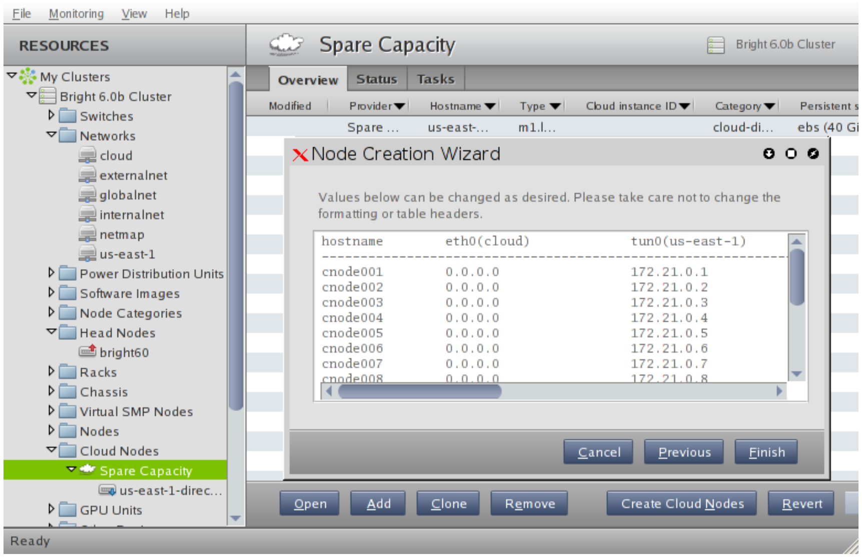
By default, both these IP offsets are `0.0.0.0`.



Interface	Network	IP-offset			
eth0:	cloud	0	0	0	0
tun0:	us-east-1	0	0	0	0

Figure 7.22: Cloud Node Wizard Network And IP Offset Configuration Screen

The last screen of the wizard (figure 7.23) shows a summary screen of the proposed IP address allocations. If the cloud IP addresses are to be assigned using DHCP, then their values are 0.0.0.0.



Values below can be changed as desired. Please take care not to change the formatting or table headers.

```
hostname        eth0(cloud)          tun0(us-east-1)
--------------------------------------------------------------
cnode001        0.0.0.0              172.21.0.1
cnode002        0.0.0.0              172.21.0.2
cnode003        0.0.0.0              172.21.0.3
cnode004        0.0.0.0              172.21.0.4
cnode005        0.0.0.0              172.21.0.5
cnode006        0.0.0.0              172.21.0.6
cnode007        0.0.0.0              172.21.0.7
cnode008        0.0.0.0              172.21.0.8
```

Figure 7.23: Cloud Node Wizard Network And IP Layout Screen

When the wizard is finished, the regular cloud nodes must be saved. This adds them to the default category by default.

If the cloud director is up, then the cloud nodes can be booted up by powering them up (section 5.2) by category, or individually.

7.3 Cloudbursting Using The Command Line And `cmsh`

The command line and `cmsh` can be used to set up Cluster-On-Demand and Cluster Extension clusters.

For Cluster-On-Demand setups, a GUI web browser is needed initially to launch the head node AMI from Amazon. However, once an `ssh` connection is made by the administrator to the head node, cloudbursting can be continued from command line. Thus, the `ddn-setup` script is run from the command line as in section 7.1.2, and the cloud nodes can be powered up from the command line as in section 7.1.4.

For Cluster Extension setups, cloudbursting can be carried out entirely from the command line. The `cloud-setup` script (section 7.3.1) sets up the cloud provider login and cloud director configuration GUI steps of section 7.2.1 in a guided manner on a command line dialog, after which `cmsh` power commands can launch the required cloud nodes (sections 7.3.2 and 7.3.3).

7.3.1 The `cloud-setup` Script

The `cloud-setup` script is run on the head node, and allows the administrator to specify settings to launch a cloud using the command line. The help text for this utility shows:

```
USAGE: /cm/local/apps/cluster-tools/bin/cloud-setup <OPTIONS>

OPTIONS:
---------
  -p | --provider <name>       Provider name (default: amazon)
  -n | --name <name>           Account name (default: Amazon EC2)
  -u | --username <username>   Username used in Amazon AWS console
  -w | --password <password>   Password used in Amazon AWS console (if
                               omitted you will be prompted, unless
                               account keys are provided)
  -a | --accountId             Amazon account ID
  -k | --awskey <key>          AWS access key ID
  -s | --secretkey <key>       AWS secret access ID

  -d | --delete                Delete Amazon account if it exists
  -y | --yes                   Use all
  -h | --help                  Print this help

NOTES:
------
  --password option does not work yet
```

It can be used as follows from the prompt:

Example

```
[root@ddnmon61 ~]# cloud-setup -u rotwang@example.com -a 123923792991 -k\
 OIQQOWU9LJJEI1232PJP -s ra9xaG7oUiy1uqu0ahW4aixuThee5ahmUGoh9cha
```

The administrator is then guided through the steps needed to launch the cloud. The session output may show up as something like (some text elided):

Example

```
Connecting to cluster
```

```
Waiting for data from cluster
Adding cloud provider Amazon EC2 ... ok
Waiting for cloud provider data ...
Got 7 regions, 29 images, 12 types
Default region (default: eu-west-1), options:
      ap-northeast-1,
      ...
      us-west-2
> eu-west-1
Default AMI (default: latest), options
      ...
      ddninstaller-074,
      ddninstaller-075
>
Default type (default: t1.micro), options
      c1.xlarge,
      ...
      t1.micro
>
Default cloud director type (default: m1.large), options
      c1.xlarge,
      ...
      t1.micro
>
Update cloud provider Amazon EC2... ok
Got 6 networks
Found tunnel network for eu-west-1: 172.16.0.0/16
Using NetMap network: 172.30.0.0/16
Using cloud network: 10.0.0.0/8
Use regions: (default eu-west-1, space separated / all), options:
      ap-northeast-1,
      ...
      us-west-2
>
Updating head node ddnmon61 ... ok
Updating tunnel network eu-west-1 ... ok
Cloud director ip on eu-west-1 (default 172.16.255.251)
>
Adding cloud director eu-west-1-director ... ok
Provisioning update started
[root@ddnmon61 ~]#
```

After `cloud-setup` has run, the cloud nodes (the cloud director and regular cloud nodes) can be launched.

7.3.2 Launching The Cloud Director

Launching the cloud requires that the cloud director and cloud nodes be powered up. This can be done using `cmgui` as described in sections 7.2.2 and 7.2.3. It can also be carried out in `cmsh`, for example, the cloud director `eu-west-1-director` can be powered up from device mode with:

Example

```
cmsh -c "device power on -n eu-west-1-director"
```

If the administrator is unsure of the exact cloud director name, one way it can easily be found is via tab-completion within the device mode of cmsh.

As explained in 7.2.2, the cloud director takes some time to power up.

7.3.3 Launching The Cloud Nodes

Once the cloud director is up, the cloud nodes can be powered up. This first requires that the cloud node objects exist and each have an IP address assigned to them that is consistent with that of the cloud director that manages them. With cmgui, this can be done with the help of a wizard to assign the IP addresses (section 7.2.3). With cmsh, assignment can be done for an individual cloud node, or for many cloud nodes, as follows:

Creating And Powering Up An Individual Node

In the example that follows, a single cloud node is assigned a management network, a tunnel IP address, and a tunnel network so that it can communicate with the cloud director.

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device
[ddnmon61->device]% add cloudnode cnode001
Warning: tunnel ip of cnode001 not set. This CloudNode will not start!
[ddnmon61->device*[cnode001*]]% set managementnetwork eu-west-1
[ddnmon61->device*[cnode001*]]% show
Parameter              Value
---------------------- --------------------------
...
Management network     eu-west-1
Network                eu-west-1
...
[ddnmon61->device*[cnode001*]]% interfaces
[ddnmon61->device*[cnode001*]->interfaces]% list
Type      Network device name   IP          Network
--------- --------------------- ----------- ----------------
physical  eth0 [prov,dhcp]      0.0.0.0     cloud-ec2classic
tunnel    tun0                  0.0.0.0
[ddnmon61->device*[cnode001*]->interfaces]% set tun0 ip 172.16.0.1
[ddnmon61->device*[cnode001*]->interfaces*]% set tun0 network eu-west-1
[ddnmon61->device*[cnode001*]->interfaces*]% list
Type      Network device name   IP          Network
--------- --------------------- ----------- ----------------
physical  eth0 [prov,dhcp]      0.0.0.0     cloud-ec2classic
tunnel    tun0                  172.16.0.1  eu-west-1
[ddnmon61->device*[cnode001*]->interfaces*]% commit
```

The preceding session completes the cloud node object configuration. The cloud node itself can now be launched with an explicit power command such as:

```
[ddnmon61->device[cnode001]->interfaces]% device power on -n cnode001
```

Creating And Powering Up Many Nodes

For a large number of cloud nodes, the creation and assignment of IP addresses can be done with the clone option of the foreach command, (section 3.5.5), together with a node range specification. This is the same syntax as used to create non-cloud regular nodes with `cmsh`. Continuing on with the preceding session, where a node `cnode001` was configured:

```
[ddnmon61->device]% foreach --clone cnode001 -n cnode002..cnode010 ()
The IP of network interface: eth0 was not updated
Warning: The Ethernet switch settings were not cloned, and have to be se\
t manually
...
[ddnmon61->device*]% commit
Mon Apr 23 04:19:41 2012 [alert] cnode002: Check 'DeviceIsUp' is in stat\
e FAIL on cnode002
[ddnmon61->device]%
Mon Apr 23 04:19:41 2012 [alert] cnode003: Check 'DeviceIsUp' is in stat\
e FAIL on cnode003
...
Successfully committed 9 Devices
[ddnmon61->device]%
```

The IP addresses are assigned via heuristics based on the value of `cnode001` and its cloud director.

As before, an alert warns each cloud node is down. The list of cloud nodes can be powered up using `cmsh` with the node range option:

Example

```
[ddnmon61->device]% foreach -n cnode002..cnode010 (power on)
```

7.3.4 Submitting Jobs With `cmsub`

The `cmsub` command is a user command wrapper to submit job scripts to a workload manager in a Cluster Extension cluster, so that jobs are considered for running in the cloud. Its use is covered in the chapter *Workload Management* in the *User Manual*.

The `cmsub` command is available from the DirectMon repository as part of the `cmsub` package. The package is installed by default on the head node.

The `cmsub` command requires that an environment module (section 3.2) called `cmsub` is loaded before it can be used. The `cmsub` environment module is not loaded by default on the head node.

7.3.5 Miscellaneous Cloud Commands

The `cloud-check` utility

The `cloud-check` utility checks the current cloud-bursting configuration for possible problems and misconfigurations. It reports any potential issues. The tool also tests communications with Amazon using some simple Amazon API calls.

Only a handful of tests are performed at present. More tests will be added over time.

The `cm-scale-cluster` utility

Introduction: The `cm-scale-cluster` command can help to reduce the energy and storage costs of compute nodes by changing their power state, or their existence state, according to workload demands. The utility collects information about jobs, and based on that it clones and starts compute nodes when the jobs are ready to start. The utility also stops or terminates compute nodes when no queued or running jobs remain on the managed nodes.

The utility is placed in a local directory when the administrator installs the `cm-scale-cluster` package. Therefore, if it is needed on a non-head node, the package should be installed in the corresponding software image. The administrator also typically configures the default user environment for users of the utility so that the `cm-scale-cluster` modules environment automatically loads up (section 3.2.3).

A compute node managed by `cm-scale-cluster` can be defined either via configuring its node group or via configuring its template node within the configuration file of the utility. The definitions in the configuration file are set with key=value options. By default, a file at `/cm/local/apps/cm-scale-cluster/etc/default.conf` is used, while the command line option `-c` allows any other file to be specified. The default configuration file has helpful commented-out configuration examples and explanations in it.

An example of a configuration file is:

Example

```
QUEUE=cloudq NODEGROUP=nodegroup1 TEMPLATE=cnode001 EXTRA_NODES=us-west\
-1-director
QUEUE=localq NODEGROUP=nodegroup2 JOBS_PER_NODE=4
TEMPLATE=cnode001 NODES=cnode[002..128] START=YES STOP=YES REMOVE=YES
EXTRA_NODE=eu-west-1-director IDLE_TIMEOUT=1800 START=YES STOP=YES
WORKLOAD_MANAGER = slurm
POLICY_MODULE = /cm/local/apps/cm-scale-cluster/lib/default-policy.py
ACTION_MODULE = /cm/local/apps/cm-scale-cluster/lib/default-action.py
PARALLEL_REQUESTS = 10
LOG_FILE  = /var/log/cm-scale-cluster.log
LOCK_FILE = /var/lock/subsys/cm-scale-cluster
SPOOL_DIR = /var/spool/cmd
DEBUG = YES
```

The command line options to `cm-scale-cluster` are described in the man page for the utility (`man(8) cm-scale-cluster`). The configuration file parameters and their key=value options are also described in the man page.

Configuration file parameters—what they mean: A further explanation of the configuration file parameters is now given. Refering back to the example configuration file should be helpful in understanding the parameters and their options when going through the explanations that follow:

- QUEUE=<*queue*> [<*key=value*>...]

  The QUEUE configuration file parameter sets, as an option, a queue that is to be watched. One queue is set per QUEUE parameter. Node groups, extra nodes, and additional options, are defined as key=value options for each queue.

- NODEGROUP: The value of the NODEGROUP option to QUEUE is a convenient way to specify compute nodes using node groups (section 3.1.4), so that they can be started, stopped, or terminated according to jobs in the associated queue. Only node groups of type `Normal` can currently be managed by `cm-scale-cluster`. These can therefore be regular local compute nodes and regular cloud compute nodes. Compute nodes are typically the most common cluster nodes. Significant resources can thus typically be saved by having the `cm-scale-cluster` utility managing these nodes, because:

  * regular cloud compute nodes can be cloned and terminated as needed. This saves on cloud storage costs associated with keeping virtual machine images.
  * regular local compute nodes can be started and stopped as needed. This reduces power consumption.

  The NODEGROUP option to QUEUE is mandatory if the TEMPLATE option to QUEUE is not set.

- TEMPLATE: The node specified for the TEMPLATE option of the QUEUE configuration file parameter must be one of the nodes specified for the TEMPLATE configuration file parameter.

  The TEMPLATE option defines a template node as a QUEUE configuration file parameter setting option, and then uses it to clone new nodes for that queue when the nodes specified by NODEGROUP are not enough. The following restrictions apply:

  * A workload manager client role must be assigned with a positive number of slots.
  * New node names should not conflict with the node names of nodes in a node group defined for the queue.
  * A specific template node is restricted to a specific queue.

  The TEMPLATE option to QUEUE is mandatory if the NODEGROUP option to QUEUE is not set.

- EXTRA_NODES: A node specified for the EXTRA_NODES option of the QUEUE configuration file parameter must be one of the extra nodes specified for the EXTRA_NODE configuration file parameter. A list of extra nodes can be assigned using commas to separate the extra nodes.

  The EXTRA_NODES option are extra nodes that are associated with the queue besides compute nodes. The `cm-scale-cluster` utility:

  * starts the extra nodes before the first job is started
  * stops the extra nodes after the last job from the managed queue is finished.

  The most common use case scenario for extra nodes in the case of cloud nodes is a cloud director node. The cloud director node provisions cloud compute nodes and performs other management operations in a cloud. In the case of non-cloud non-head nodes, extra nodes can be, for example, a license server, a provisioning node, or an additional storage node.

  - NEVER_TERMINATE: Number of cloud nodes which will never be terminated even if no jobs need them. If there are this number or fewer cloud nodes, then cm-scale-cluster no longer terminates them. Cloud nodes that cannot be terminated can, however, still be powered off, allowing them to remain configured in DirectMon.

    As an aside, local nodes that are under cm-scale-cluster control are powered off automatically when no jobs need them, regardless of the NEVER_TERMINATE value.

    Default: 0.
  - JOBS_PER_NODE: How many jobs can use a node at once. Default: 1.

- TEMPLATE=<*template node*> [<*key=value*>...]

  The TEMPLATE configuration file parameter sets a template node. One template node is set per TEMPLATE parameter. The template node set for this parameter is a node defined by DirectMon.

  Further key=value options specify the nodes that use the template node, along with other settings relevant to the template node.

  The cm-scale-cluster utility creates new nodes, including cloud nodes, by cloning the template if there are not enough nodes configured and running under CMDaemon control. The nodes that are cloned are listed in the NODES key to the TEMPLATE configuration file parameter.

  A template node only has to exist as an object in the DirectMon, with an associated node image. A template node does not need to be up and running physically in order for it to be used to create clones. Sometimes, however, an administrator may want it to run too, like the other nodes that are based upon it, in which case the START and STOP values apply.

  The TEMPLATE configuration file parameter takes the following key=value options:

  - START: Is the template node allowed to start automatically? Default: NO
  - STOP: Is the template node allowed to stop automatically? Default: NO
  - NODES: The range of the new nodes that can be cloned from the template. The ways in which the range can be specified are given in man(8) cm-scale-cluster. Default: <*blank*>
  - REMOVE: Should the new node be removed from DirectMon when the node terminates? If the node is not going to be terminated, but just stopped, then it will never be removed. Default: NO
  - INTERFACE: The network interface. Its IP address will be increased by 1 (IPv4) when the node is cloned. Default: tun0
  - LEAVE_FAILED_NODES: This setting decides if nodes discovered to be in a state of INSTALLER_FAILED or INSTALLER_UNREACHABLE (section 6.5.4) can be left alone,

so that the administrator can decide what do with them later on. Otherwise they are terminated automatically if they are cloud nodes, and powered off if they are local nodes. Default: `NO`

- EXTRA_NODE=<*extra node*> [<*key=value*>...]

  The EXTRA_NODE configuration file parameter sets, as an option, an extra node. One extra node is set per EXTRA_NODE parameter. This extra node is a node defined by DirectMon. It is a non-head node that is always needed for regular or cloud nodes to run. Typically, for cloud nodes it is a cloud director, while for regular nodes it is a provisioning node or a licensing server.

  Each extra node can take the following key=value options:

  – `IDLE_TIMEOUT`: The maximum time, in seconds, that extra nodes can remain unused. The `cm-scale-cluster` utility checks for the existence of queued and active jobs using the extra node when the time elapsed since the last check reaches `IDLE_TIMEOUT`. If there are jobs using the extra node, then the time elapsed is reset to zero and a time stamp is written into the file `cm-scale-cluster.state` under the directory set by the `SPOOL_DIR` parameter. The time stamp is used to decide when the next check is to take place. Setting `IDLE_TIMEOUT=0` means the extra node is stopped whenever it is found to be idle, and started again whenever jobs require it, which may result in a lot of stops and starts. Default: `3600`.

  – `START`: Automatically start extra node before the first compute node starts. Default: `YES`.

  – `STOP`: Automatically stop extra node after the last compute node stops. Default: `YES`.

- WORKLOAD_MANAGER=<*workload manager*>

  The WORKLOAD_MANAGER configuration file parameter sets the current workload manager as an option. Possible values are: `slurm`, `pbspro`, `torque`, `sge`, `lsf` , `openlava`. Default: `slurm`

- DEBUG=<`YES|NO`>

  The DEBUG parameter allows `cm-scale-cluster` to append debug information to the log file. Default: `NO`

- PARALLEL_REQUESTS=<*number*>

  The PARALLEL_REQUESTS parameter sets a limit to the number of simultaneous requests to CMDaemon from `cm-scale-cluster` when requesting information about the changes being applied. Default: `10`

- CMD_TIMEOUT=<*number*>

  The CMD_TIMEOUT parameter sets the maximum time, in seconds, that `cm-scale-cluster` waits for a response from CMDaemon. Default: 5

- LOG_FILE=<*file*>

  The LOG_FILE parameter sets the full path to the log file for `cm-scale-cluster`. Default: `/var/log/cm-scale-cluster.log`.

- LOCK_FILE=<*file*>

  The LOCK_FILE parameter prevents the simultaneous execution of several copies of `cm-scale-cluster` when the lock file is defined. Default: `/var/lock/subsys/cm-scale-cluster`

- SPOOL_DIR=<*directory*>

  The SPOOL_DIR parameter defines the directory where `cm-scale-cluster` creates temporary files. Default: `/var/spool/cmd`

- POLICY_MODULE=<*file*>

  The POLICY_MODULE parameter defines a path to a Python script containing policy rules. The policy rules define what kind of nodes should start, and how many, based on the current number of queued jobs requests. Default: `/cm/local/apps/cm-scale-cluster/lib/default-policy.py`

- ACTION_MODULE=<*file*>

  Set a Python script containing action rules. The action rules define how nodes start, stop, or terminate. Default: `/cm/local/apps/cm-scale-cluster/lib/default-action.py`

Configuration file parameters—in action: In the example configuration on page 279 two queues are defined: `cloudq` and `localq`.

- For `cloudq`: When all nodes from node group `nodegroup1` are busy and new jobs are queued in `cloudq`, then `cm-scale-cluster` will use the template of `cnode001` to clone the nodes `cnode[002..128]` to satisfy queued jobs needs. Before the first node from the `cloudq` queue is started, `cm-scale-cluster` makes sure that `us-west-1-director` extra node is running.

- For `localq`: In the case of the `localq` queue, no nodes will be created automatically, but they will be started (or stopped) nodes from the nodes of node group `nodegroup2`. Also, for jobs from `localq`, it will be taken into account that each node can run up to 4 jobs.

Running `cm-scale-cluster` with `cmsub`: When `cmsub` (section 7.3.4) is used with `cm-scale-cluster` to bring up cloud nodes, then it is recommended to specify a cloud director node as an EXTRA_NODE. This is because `cmsub` can only do its work if a cloud director is there to accept jobs and data.

7.4 Cloud Considerations And Issues With DirectMon

7.4.1 Differences Between Cluster-On-Demand And Cluster Extension

Some explicit differences between Cluster-On-Demand and Cluster Extension clusters are:

Cluster-On-Demand	Cluster Extension
cloud nodes only in 1 region	cloud nodes can use many regions
no cloud director	uses one or more cloud directors per region
no failover head node	failover head node possible
no VPN or NetMap	VPN and NetMap
no externalnet interface on head	can have an external interface
cluster has publicly accessible IP address	cloud directors have publicly accessible IP addresses

A note about the last entry: The access to the cloud director addresses can be restricted to an administrator-defined set of IP addresses, using the "`Externally visible IP`" entry in figure 4.2.

7.4.2 Hardware And Software Availability

DDN head node AMIs are available for the following distributions: RHEL5/RHEL6, SL5/SL6, CentOS5/CentOS6, and SLES 11 SP1/SP2.

AMIs with GPU computing instances are available with Amazon cloud computing services only in the US East (Virginia) region the last time this was checked (April 2012). These can be used with DDN AMIs with `hvm` in the name (not `xen` in the name).

To power the system off, a `shutdown -h now` can be used, or the power commands for `cmgui` or `cmsh` can be executed. These commands stop the instance, without terminating it. Any associated extra drives that were created need to be removed manually, via the `Volumes` screen in the `Elastic Block Store` resource item in the navigation menu of the AWS Management Console.

7.4.3 Reducing Running Costs

Spot Pricing

The spot price field is a mechanism to take advantage of cheaper pricing made available at irregular[1] times. The mechanism allows the user to decide a threshold spot price (a price quote) in US dollars per hour for instances. Instances that run while under the threshold are called *spot instances*. Spot instances are described further at `http://aws.amazon.com/ec2/spot-instances/`.

With the pricing threshold set:

[1]irregular turns out to be random within a tight range, bound to a reserve price. Or rather, that was the case during the period 20th January–13th July, 2010 that was analyzed by Ben-Yehuda et al, `http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-info.cgi/2011/CS/CS-2011-09`

- If the set spot price threshold is above the instantaneous spot price, then the spot instances run.
- If the set spot price threshold is below the instantaneous spot price, then the spot instances are killed.
- If the set spot price threshold is N/A, then no conditions apply, and the instances will run on demand regardless of the instantaneous spot price.

An *on demand instance* is one that runs regardless of the price, according to the pricing at http://aws.amazon.com/ec2/pricing/.

A *persistent request* is one that will retry running a spot instance if the conditions allow it.

Storage Space Reduction

Reducing the amount of EBS disk storage used per cloud node or per cloud director is often feasible. 15 GB is usually enough for a cloud director, and 5 GB is usually enough for a cloud node with common requirements. In cmsh these values can be set with:

Example

```
[ddnmon61]% device cloudsettings eu-west-1-director
[ddnmon61->device[eu-west-1-director]->cloudsettings]% storage
[ddnmon61->...->cloudsettings->storage]% set ebs size 15GB; commit
[ddnmon61->...->cloudsettings->storage]% device cloudsettings cnode001
[ddnmon61->device[cnode001]->cloudsettings]% storage
[ddnmon61->...->cloudsettings->storage]% set ebs size 5GB; commit
```

The value for the cloud node EBS storage can also be set in the cloud node wizard (fig. 7.21) for a Cluster Extension configuration.

7.4.4 Address Resolution In Cluster Extension Networks

Resolution And globalnet

The globalnet network is introduced in section 4.3.3 and allows an extra level of redirection during node resolution. The reason for the redirection is that it allows the resolution of node names across the entire cluster in a hybrid cluster, regardless of whether the node is a cloud node (cloud director node or regular cloud node) or a non-cloud node (head node, regular node or networked device). A special way of resolving nodes is needed because the Amazon IP addresses are in the 10.0.0.0/8 network space, which conflicts with some of the address spaces used by DirectMon.

There are no IP addresses defined by globalnet itself. Instead, a node, with its domain defined by the globalnet network parameters, has its name resolved by another network to an IP address. The resolution is done by the nameserver on the head node for all nodes.

Resolution In And Out Of The Cloud

The networks, their addresses, their types, and their domains can be listed from the network mode in cmsh:

```
[ddnmon61->network]% list
Name (key)    Type      Netmask bits  Base address  Domain name
```

```
------------ --------- ------------- ------------- -----------
bmcnet       Internal  16            10.148.0.0    bmc.cluster
cloud        Cloud     8             10.0.0.0      cloud.cluster
externalnet  External  16            10.2.0.0      ddn.com
globalnet    Global    0             0.0.0.0       cm.cluster
ibnet        Internal  16            10.149.0.0    ib.cluster
internalnet  Internal  16            10.141.0.0    eth.cluster
netmap       NetMap    16            172.30.0.0
us-east-1    Tunnel    16            172.21.0.0
```

In a Type 1 network (section 2.3.6), the head node is connected to internalnet. When a cloud service is configured, the head node is also "connected" to the CMDaemon-managed NetMap "network". It is useful to think of NetMap as a special network, although it is actually a network mapping from the cloud to internalnet. That is, it connects (maps) from the nodes in one or more cloud networks such as the us-east-1 network provided by Amazon, to IP addresses provided by netmap. The mapping is set up when a cloud extension is set up. With this mapping, packets using NetMap go from the cloud, via an OpenVPN connection to the NetMap IP address. Once the packets reach the OpenVPN interface for that address, which is actually on the head node, they are forwarded via Shorewall's IPtables rules to their destination nodes on internalnet.

With default settings, nodes on the network internalnet and nodes in a cloud network such as us-east-1 are both resolved with the help of the cm.cluster domain defined in globalnet. For a cluster with default settings and using the cloud network us-east-1, the resolution of the IP address of 1. a regular node and 2. a regular cloud node, takes place as follows:

1. node001, a regular node in the internalnet network, is resolved for node001.cm.cluster to
   (a) 10.141.0.1, when at the head node. The cluster manager assigns this address, which is on internalnet. It could also be an ibnet address instead, such as 10.149.0.1, if InfiniBand has been configured for the nodes instead of Ethernet.
   (b) 172.30.0.1 when at the cloud director or regular cloud node. The cluster manager assigns this address, which is a NetMap address. It helps route from the cloud to a regular node. It is not actually an IP address on the interface of the regular node, but it is convenient to think of it as being the IP address of the regular node.
2. cnode001, a regular cloud node in the us-east-1 network, is resolved for cnode001.cm.cluster to:
   (a) 172.21.0.1 when at the head node. The cluster manager assigns this address, which is an OpenVPN tunnel address on us-east-1.
   (b) an IP address within 10.0.0.0/8 (10.0.0.1–10.255.255.254) when at a regular cloud node or at a cloud director. The Amazon cloud network service assigns the addresses in this network to the cloud director and regular cloud nodes.

An explanation of the networks mentioned in the preceding list follows:

- The nodes within all available cloud networks (all networks such as for example, `us-east-1`, `us-west-1`, and so on) are given CMDaemon-assigned addresses in the cloud node space range 172.16.0.0–172.29.255.255. In CIDR notation that is: 172.16.0.0/12 (172.16.0.0–172.31.255.255), except for 172.31.0.0/15 (172.30.0.0–172.31.255.255).
- The network address space 172.30.0.0/16 (172.30.0.0–172.30.255.255) is taken by the CMDaemon-assigned NetMap network, explained shortly. The addressing scheme for each cloud network is assigned as suggested in figure 7.17.
- Each node in a cloud network is also assigned an address in the network addressing space provided by Amazon. The assignment of IP addresses to nodes within the 10.0.0.0/8 range is decided by Amazon via DHCP.
- The `netmap` "network" (figure 7.24) is a helper mapping reserved for use in routing from the cloud (that is, from a cloud director or a cloud node) to a regular node. The mapping uses the 172.30.0.0/16 addressing scheme. Its routing is asymmetrical, that is, a NetMap mapping from a regular node to the cloud does not exist. Packets from a regular node to the cloud do however resolve to the `cloud` network as indicated by 2(a) in the preceding.

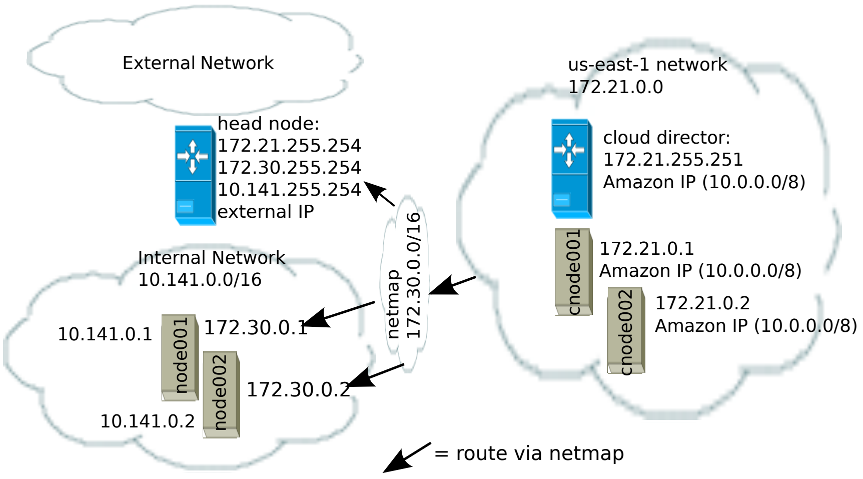


Figure 7.24: NetMap In Relation To The General Network Scheme

As pointed out in the introduction to this section (7.4.4), the main reason for the IP addressing network scheme used is to avoid IP address conflicts between nodes within the cloud and nodes outside the cloud.

The *difference* in resolution of the IP address for the nodes as listed in points 1 and 2 in the preceding text is primarily to get the lowest overhead route between the source and destination of the packet being routed. Thus, for example, a packet gets from the regular cloud node to the cloud director with less overhead if using the Amazon cloud IP addressing scheme (10.0.0.0/8) than if using the DDN OpenVPN addressing scheme (172.21.0.0/16). A secondary reason is convenience and reduction of networking complexity. For example, a node in the cloud may shut down and start up, and get an arbitrary Amazon IP address, but using an OpenVPN network such as `us-east-1` allows it to retain its OpenVPN address and thus stay identified instead of having the properties that have been assigned to it under DirectMon become useless.

7.5 Virtual Private Clouds

A *virtual private cloud* is an implementation of a cluster on a virtual network in a cloud service provider. The Amazon Virtual Private Cloud (Amazon VPC) is an implementation of such a virtual private cloud. The Amazon VPC is documented more fully at `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-vpc.html`.

Managing VPCs would normally require significant networking expertise. DirectMon makes it easier to do this, so that the administrator can focus on using them productively, instead of on working out VPC configurations.

The following VPC-related terms are explained and compared in this section:

- *EC2-Classic* (page 288)
- *EC2-VPC* (page 288)
- *classic cloud* (page 289)
- *defaultVPC* (page 289)
- *private cloud* (page 290)
- *custom VPC* (page 290)
- *elastic IP addresses* (page 293)

7.5.1 EC2-Classic And EC2-VPC

EC2-Classic Vs EC2-VPC Overview

So far, this manual has discussed configuring clusters within Amazon EC2. The EC2 designation actually covers two kinds of platforms:

- EC2-Classic: This platform provides an environment that corresponds to a physical network. Instances in the same region exist on the same physical network and rely on explicitly configured security groups to restrict unauthorized access from other instances on the same network. A cloud instance that is created in such a network can be called a classic cloud cluster, or simply a cloud cluster.

  Amazon is gradually phasing out the EC2-Classic platform.

- EC2-VPC: This platform is replacing EC2-Classic. It provides an environment corresponding to an isolated virtual network. A cloud cluster instance implemented on this virtual network is thus a virtual private cloud, or VPC, as described at the start of this section (section 7.5).

  The EC2-VPC platform offers some extra features that are not available, or not as easy to configure, on the EC2-Classic platform:

– Multiple VPCs can be configured per region
– The inherent isolation of Amazon VPCs makes them more secure by default
– their network properties can be customized

The isolated network design of a VPC means that instances started within a VPC cannot by default communicate with instances outside. *Elastic IP* addresses (page 293) are used to explicitly allow such communication.

EC2-Classic Vs EC2-VPC And AWS Account Creation Date

The type of platform that can be accessed by an AWS account varies as indicated by the following table:

Account Creation Date	Typical Platform Offered
Before start of 2013	EC2-Classic only
In first half of 2013	EC2-Classic or EC2-VPC*
After first half of 2013	EC2-VPC only, in most or all regions

*Typically depends on the region accessed.

Most new AWS accounts do not provide an EC2-Classic platform. However, to maintain backwards compatibility for users who are migrating to EC2-VPC, and who have applications that run on the EC2-Classic platform, Amazon provides the defaultVPC instance on the EC2-VPC platform.

The Classic Cloud And The DefaultVPC Instances

The *classic cloud* is a cloud instance that EC2-Classic supports.

The `defaultVPC` instance is a special VPC instance that emulates EC2-Classic behavior on the EC2-VPC platform. This allows legacy applications that do not support EC2-VPC to run on it. A legacy application that runs in a defaultVPC instance may be thought of as having its EC2-Classic API calls translated into EC2-VPC API calls. The defaultVPC instance is available in all regions that do not offer the EC2-Classic platform.

There is one major difference between the network environments of EC2-Classic and the defaultVPC instance: For EC2-Classic instances, the base address of network inside Amazon is 10.0.0.0/8. In contrast, for defaultVPC instances the base address is 172.30.0.0/16.

When creating a new cloud provider account, DirectMon automatically detects which regions offer the EC2-Classic platform, and which do not. The suggested base address of the cloud network that is to be created is then automatically matched according to the regions. The platform supported, EC2-Classic or EC2-VPC, is also displayed in `cmgui` when the cloud director is being created.

A few Amazon AWS accounts provide the EC2-Classic platform for only a certain subset of all available regions, and provide EC2-VPC in other regions. In such a case, when a new cloud provider account is created in DirectMon with a cloud director in both types of platforms, then two cloud networks can be created. If only a single cloud director is run, then only one network is created, and the network base address in that case depends on the platform, EC2-Classic or EC2-VPC, that it is run on. However, if two cloud directors are started up, with each cloud director on a different platform, then one cloud director runs on one platform and associated network, and the other cloud director on the other platform and associated network.

The Private Cloud And Custom VPC Instances

A *private cloud* is the term used in the manual, in DirectMon, by Amazon, and in general, for a general VPC instance.

A *custom VPC* is the term used in the manual to mean a general VPC instance, but one that is not a defaultVPC instance.

Thus, in terms of math sets:

`private clouds` = `custom VPCs` + `defaultVPCs`

In the context of Amazon VPCs, the term private cloud is often used by administrators, by convention and for convenience, to mean the more precise term of custom VPC as defined here, implicitly ignoring possible defaultVPC instances. The DirectMon software itself also follows this convention. In this section of the manual (7.5), however, using the term "private cloud" for this is avoided, and the terms are adhered to precisely as defined, in order to avoid confusion.

Attempting to change a defaultVPC instance to a custom VPC instance by editing defaultVPC properties directly with DirectMon is not possible, because these properties are hidden behind the EC2-Classic facade. This kind of change can be done via the Amazaon Webconsole instead. If DirectMon requires that the custom VPC functionality of a general VPC instance is needed in Amazon VPC, then a custom VPC has to be created within DirectMon. How to do this is described in section 7.5.3.

Cloud Cluster Terminology Summary

The cluster terminology used so far can be summarized as follows:

cluster term	platform	type and connectivity
classic cloud	EC2-Classic	classic cloud cluster that has direct connectivity to the outside
defaultVPC	EC2-VPC	a VPC that looks like it has direct connectivity to the outside because it emulates a classic cloud cluster
custom VPC	EC2-VPC	isolated VPC with no connectivity to the outside by default, and NAT gateway connectivity to the outside when made to connect
private cloud	EC2-VPC	both defaultVPC and custom VPC

7.5.2 Comparison Of EC2-Classic And EC2-VPC Platforms

There are several differences between EC2-Classic and EC2-VPC platforms. The most important ones are:

- Cloud nodes created inside the EC2-VPC platform do not have an external (public) IP address assigned to them by default. An exception to this is the case of nodes running in a defaultVPC instance, which emulates EC2-Classic network behaviour. Having no public IP address by default allows for a greater degree of out-of-the-box security.

- Custom VPCs are self-contained and securely isolated from the instance of other users.
- Custom VPCs are partitioned into multiple network segments, called *subnets* (section 7.5.3).
- It is possible to specify a custom base network address for the custom VPC. This is in contrast to the EC2-Classic platform, where a base network address always has the value of 10.0.0.0/8. For a defaultVPC instance the base network address takes the value of 172.30.0.0/16.

7.5.3 Setting Up And Creating A Custom VPC

By default, when DirectMon creates a new cloud provider account, the cloud nodes created are EC2-Classic instances or defaultVPC instances inside the EC2-VPC platform. That is, they are not nodes in a custom VPC instance. This default behavior is expect to change in a later version of DirectMon as Amazon and DirectMon both evolve.

DirectMon can be set to create custom VPC instances inside the EC2-VPC platform. The EC2-VPC platform is recommended for all new cloud-bursting setups.

Subnets In A Custom VPC

The components of a custom VPC include subnets, the nodes that run in them, and static IP addresses. The subnets are logical network segments within the network range of that custom VPC. Subnets can be thought of as interconnected with a central "black-box" router, with DirectMon managing the routing tables on that router. The routing ensures correct subnet communication. Inside DirectMon, subnets are represented as a type of network (section 4.3), with a value for type set to `CLOUD`.

Subnets for a custom VPC must have non-overlapping ranges. If there are multiple custom VPCs being managed by DirectMon, then a particular subnet may be assigned to one custom VPC at the most.

Two series of valid network ranges could be:

Example

1. 10.0.0.0-10.0.31.255 (10.0.0.0/19),
   10.0.32.0-10.0.63.255 (10.0.32.0/19),
   10.0.64.0-10.0.95.255 (10.0.64.0/19).
2. 192.168.0.0-192.168.0.255 (192.168.0.0/24),
   192.168.1.0-192.168.1.255 (192.168.1.0/24).

The `sipcalc` command (page 97) is a useful tool for calculating appropriate subnet ranges. At least one subnet must be assigned to a custom VPC before an instance can be created in that cloud. Typically two or more subnets are assigned, as shown in the custom VPC creation example in the following section.

Creating The Custom VPC

After subnets have been configured, a custom VPC can be created by specifying:

- the name
- the default region
- base address
- number of netmask bits

The network of the custom VPC must obviously be a superset of its subnets. Any subnets of the custom VPC must also be specified. Subnets can be added to or removed from an already-created custom VPC, but only if any cloud node instances within them are terminated first.

There are several ways to set up and create the subnets and custom VPC instance in DirectMon:

1. by using the command line `cloud-setup-private-cloud` utility,
2. by using the `cmgui` private cloud creation dialog box,
3. by manually creating and configuring the `private cloud` object using `cmsh`.

These are described next:

1. Subnet Setup And Custom VPC Instance Creation Using `cloud-setup-private-cloud`

Once the cloud provider account has been configured, using the `cloud-setup` utility (section 7.3.1), or by using the `cmgui` wizard (section 7.2.1), the `cloud-setup-private-cloud` utility can then be run to set up a custom VPC.

The utility prompts the user to choose a cloud provider account, a region to create the VPC in, and the base address of the VPC. It goes on to create the custom VPC, and finishes by prompting whether to move any eligible cloud nodes to the custom VPC.

2. Subnet Setup And Custom VPC Creation Using `cmgui`

For the cloud provider resource item, inside the `Private Clouds` tab, clicking the `Add` button launches a dialog box to create a custom VPC.

3. Subnet Setup And Custom VPC Creation Using `cmsh`

Similarly with `cmsh`, the subnets to be used for the custom VPC are created first, before creating the private cloud, as shown in the following examples.

- **Subnet creation and cloning:** In the following example session, an arbitrary naming scheme is used for subnets, with a pattern of: `<`*`name of custom VPC`*`>-sn-<`*`number`*`>`. Here, `sn` is an arbitrary abbreviation for “subnet”:

  Example

```
[ddnmon61->network]% add vpc-0-sn-0
[ddnmon61->network*[vpc-0-sn-0*]]% set type cloud
[ddnmon61->network*[vpc-0-sn-0*]]% set baseaddress 10.0.0.0
[ddnmon61->network*[vpc-0-sn-0*]]% set netmaskbits 24
[ddnmon61->network*[vpc-0-sn-0*]]% set ec2availabilityzone eu-west-1a
[ddnmon61->network*[vpc-0-sn-0*]]% commit
```

Setting the `ec2availabilityzone` property is optional. It causes the subnet to be created in a specific availability zone. Leaving its value empty creates the subnet inside a randomly chosen availability zone. Having all subnets of the custom VPC inside the same availability zone is advised for better network performance. The availability zone set for the network must be one of the availability zones available for the region inside which the private cloud will be created.

Once the first subnet has been created, it can be cloned:

Example

```
[ddnmon61->network]% clone vpc-0-sn-0 vpc-0-sn-1
[ddnmon61->network*[vpc-0-sn-1*]]% set baseaddress 10.0.1.0
[ddnmon61->network*[vpc-0-sn-1*]]% commit
```

- **Custom VPC creation:** The following example session in the `privateclouds` submode of the `cloud` mode, creates a private cloud called `vpc-0`. The private cloud is actually a custom VPC according to the strict definition of a private cloud instance in the section on page 290. It is of type `ec2` and within a network that contains the two subnets specified earlier.

Example

```
[ddnmon61->cloud[Amazon EC2]->privateclouds]%
[ddnmon61->...->privateclouds]% add ec2privatecloud vpc-0
[ddnmon61->...->privateclouds*[vpc-0*]]% set region eu-west-1
[ddnmon61->...*[vpc-0*]]% set baseaddress 10.10.0.0
[ddnmon61->...*[vpc-0*]]% set netmaskbits 16
[ddnmon61->...*[vpc-0*]]% set subnets vpc-0-sn-0 vpc-0-sn-1
[ddnmon61->...*[vpc-0*]]% commit
```

Elastic IP Addresses And Their Use In Configuring Static IP Addresses

Unlike defaultVPC and EC2-Classic instances, a custom VPC instance does not have an externally visible (public) IP address assigned to it by Amazon by default. Without an externally visible IP address, the custom PVC cannot communicate with the internet, and it cannot even be an endpoint to an outside connection. To solve this issue, Amazon *elastic IP addresses* (EIPs) can be used to assign a public IP address to a custom VPC cloud.

EIP addresses are the public IP addresses that Amazon provides for the AWS account. These addresses are associated with defaultVPC and EC2-Classic cloud instances by Amazon by default. These addresses can also be associated with custom VPC instances. The public addresses in the set of addresses can then be used to expose the custom VPC instance. In this manual and in DirectMon, EIPs are referred to as “static IPs” in the cloud context. When allocating a static IP addres, the exact IP address that is allocated is a random IP address from the set of all public IP addresses made available in the specified region by the configured cloud provider.

Automatic allocation of static IP addresses: When a cloud director instance is started inside a custom VPC, CMDaemon automatically allocates and assigns a static IP address to it. By default, the static IP address is automatically released when the cloud director instance is terminated. This behavior can be changed in the CMDaemon cloud settings for the cloud director.

Manual allocation of static IP addresses: It is also possible to manually allocate a static IP address to a cloud director using `cmgui` or `cmsh`.

Allocating a static IP address in `cmsh` is done using the `staticip allocate` command, followed by the string indicating the region in which the static IP address is to be allocated. In `cmsh`, the command is issued inside a cloud provider object. A new static IP address is then made available and can be assigned to instances running within custom VPCs.

After allocation, the static IP address can be assigned and reassigned to any instance inside any custom VPC created within the region in which the IP address was allocated.

Example

```
[ddnmon61] cloud use amazonec2
[ddnmon61->cloud[Amazon EC2]]% staticip allocate us-west-1
Allocating Static IP. Please wait...
Successfully allocated the following static IP: 54.215.158.42
[ddnmon61->cloud[Amazon EC2]]% staticip list
Cloud Provider   Cloud Region  Static IP      Assigned to
---------------- ------------- -------------- ---------------
Amazon EC2       us-west-1     54.215.158.42  <not assigned>
[ddnmon61->cloud[Amazon EC2]]%
```

An allocated static IP can be released with the `staticip release` command in `cmsh`:

Example

```
[ddnmon61->cloud[Amazon EC2]]% staticip release 54.215.158.42
Releasing static IP 54.215.158.42. Please wait...
Successfully released the static ip.
[ddnmon61->cloud[Amazon EC2]]%
```

Once the IP address has been released, it may no longer be used for instances defined in the custom VPC.

The `staticips` command lists all allocated static IPs for all configured cloud providers.

The `staticip list` command lists static IP addresses for the currently active cloud provider object.

In `cmgui` the static IPs can be managed via the “Static IPs” tab of a cloud provider object.

Subnets With Static IP Addresses In A Custom VPC Recommendation

Subnets can be set up in many ways inside a custom VPC. The following is recommended:

- There must be exactly one network containing all the instances which have static IP addresses. This network should contain the cloud director. The network with the cloud director is arbitrarily referred to as the "public" network.
- There must be zero or more networks containing instances with no static IP addresses assigned to them. Such networks are arbitrarily referred to as the "private" subnets.

Instances in the private subnets have no static IP addresses assigned to them, so by default they do not communicate with outside networks. To allow them to connect to the outside, the cloud director instance is automatically configured by CMDaemon as a NAT gateway for outside-bound traffic, for the instances existing inside the private subnets.

Assignment Of Nodes To Subnets And Cloud Platforms

A cloud node instance is connected to a network by its `eth0` interface. The network is one of those covered in following table, that is: classic physical, classic emulated, or subnet of a custom VPC.

what cloud is the eth0 interface connected to?	**cloud instance type and network that the node joins**
classic cloud	classic cloud cluster instance, in classic physical network (10.0.0.0/8)
defaultVPC	defaultVPC instance, in classic emulated network (172.30.0.0/16)
custom VPC	inside VPC instance, in the connected subnet (if any) of that network

Therefore, when the cloud node is being created inside EC2, the CMDaemon must tell the EC2 environment which of these networks is going to be attached to the eth0 interface of the newly created cloud node.

This information is deduced by CMDaemon by looking at the interface configuration of the cloud node DirectMon. More specifically, it is deduced from the value set for `network` in the cloud node's eth0 interface settings.

If that network is part of a custom VPC, that is, if it is a subnet, then the node starts inside the custom VPC instance. Otherwise, it starts inside the EC2-Classic or defaultVPC instance in that region.

For example, the cloud director node is started inside the `vpc-0-sn-0` network in the following session. It is considered a custom VPC node, and starts up inside the EC2-VPC platform:

Example

```
[ddnmon61->device[us-west-1-director]->interfaces]% list
Type         Network device name  IP               Network
```

```
------------ -------------------- ---------------- ----------
physical     eth0 [dhcp]          0.0.0.0          vpc-0-sn-0
tunnel       tun0 [prov]          172.18.255.251   us-west-1
```

In contrast, if the cloud network assigned to the `eth0` interface is the cloud network representing the network environment of an EC2-Classic or defaultVPC cloud, then the node is considered to be an EC2-Classic node. It then starts up inside the EC2-Classic platform by default:

Example

```
[ddnmon61->device[us-west-1-director]->interfaces]% list
Type         Network device name  IP               Network
------------ -------------------- ---------------- ----------
physical     eth0 [dhcp]          0.0.0.0          cloud
tunnel       tun0 [prov]          172.18.255.251   us-west-1
```

Once a cloud node instance has been instantiated inside a specified subnet it cannot be reassigned to a different subnet, nor can it be reassigned to a different custom VPC. The cloud instance must first be terminated and reconfigured. Only then can it be powered on inside a different subnet.

Creating A Cloud Director In A Custom VPC

To create a cloud director in a custom VPC using `cmgui`, the cloud director must first be created inside the EC2-Classic platform region in which the custom VPC is created. This can be done via the `Add Cloud Director` button inside the `Overview` tab of a cloud provider account. After this has been done, the cloud director must be moved from the EC2-Classic platform to the custom VPC, as explained in section 7.5.3.

Creating Cloud Compute nodes In A Custom VPC

Creating cloud compute nodes inside a custom VPC can be done in a similar way to creating cloud compute nodes for the EC2-Classic platform. That is, by clicking the `Create Cloud Nodes` button in the overview tab of the cloud provider in `cmgui`. However, to create a cloud node inside the custom VPC, a subnet of the custom VPC must be specified when selecting the network of the eth0 interface of the node. To avoid confusion, it is sensible to make this a different subnet from the one in which the cloud director node for that particular custome VPC is assigned.

An alternative solution to creating cloud compute nodes in a custom VPC is to instruct the cluster manager to automatically move the existing ones while also moving the cloud director to the custom VPC, as explained in the following section.

Moving Existing Nodes To A Custom VPC

After a custom VPC has been configured, it is possible to automatically reconfigure the existing cloud nodes to make them start inside that custom VPC. This is an alternative to creating new nodes inside the custom VPC from scratch. An existing cloud director can be moved to a custom VPC cloud using the `Move Cloud Director` button in `cmgui`. This button can be clicked in the `Overview` tab of a cloud provider account, and it opens up a dialog box. After completion of the dialog, the cloud director is moved to a custom VPC in the same region. It can also move any other cloud compute nodes managed by the selected cloud director.

Moving a node to a custom VPC effectively terminates the current EC2 instance, and creates a new one inside the target custom VPC.

8 User Management

Users and groups for the cluster are presented to the administrator in a single system paradigm. That is, if the administrator manages them with the DirectMon, then the changes are automatically shared across the cluster (the single system).

DirectMon runs its own LDAP service to manage users, rather than using Unix user and group files. In other words, users and groups are managed via the centralizing LDAP database server running on the head node, and not via entries in `/etc/passwd` or `/etc/group` files.

Sections 8.1 and 8.2 cover the most basic aspects of how to add, remove and edit users and groups using DirectMon.

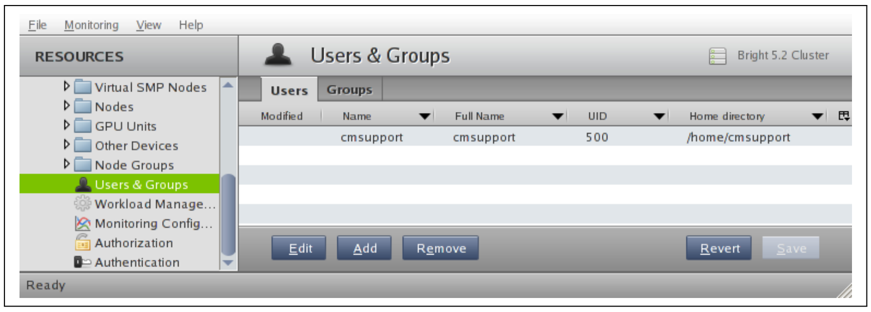
Section 8.3 describes how an external LDAP server can be used for authentication services instead of the one provided by DirectMon.

Section 8.4 discusses how users can be assigned only selected capabilities when using `cmgui` or `cmsh`, using profiles with sets of tokens.

8.1 Managing Users And Groups With `cmgui`

Selecting "`Users & Groups`" from the `Resources` tree within `cmgui` (figure 8.1) by default lists the LDAP object entries for regular users. These entries are clickable and can be managed further.

There is already one user on a newly installed DirectMon: `cmsupport`. This user has no password set by default, which means (section 8.2.2) no logins to this account are allowed by default. DirectMon uses the user `cmsupport` to run various diagnostics utilities, so it should not be removed, and the default contents of its home directory should not be removed.

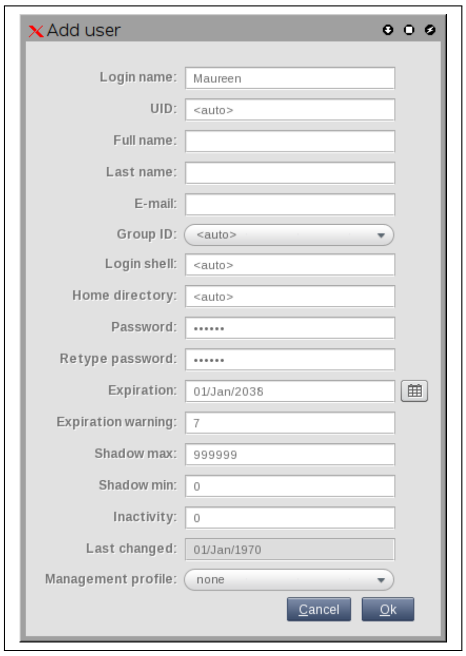


Modified	Name	Full Name	UID	Home directory
	cmsupport	cmsupport	500	/home/cmsupport

Figure 8.1: `cmgui` User Management

There are five buttons, `Add`, `Save`, `Edit`, `Revert`, and `Remove`, available in the `Users & Groups` resource tabbed pane:

1. `Add`: allows users to be added via a dialog (figure 8.2). These additions can be committed via the `Save` button.



Login name:	Maureen
UID:	<auto>
Full name:	
Last name:	
E-mail:	
Group ID:	<auto>
Login shell:	<auto>
Home directory:	<auto>
Password:	••••••
Retype password:	••••••
Expiration:	01/Jan/2038
Expiration warning:	7
Shadow max:	999999
Shadow min:	0
Inactivity:	0
Last changed:	01/Jan/1970
Management profile:	none

Figure 8.2: `cmgui` User Management: Add Dialog

An explanation of the less obvious items in the dialog follows:

- `Expiration warning`: The number of days, before the password expires, that the user is warned of the expiry
- `Shadow max`: The maximum number of days the password is valid
- `Shadow min`: The minimum number of days required between password changes. A value of zero means the user may change their password at any time
- `Inactivity`: The number of days of inactivity allowed for the user before the account is blocked. A value of zero means the user is never blocked
- `Management profile`: The preconfigured capability that the user is assigned. Available settings are:
  - `admin`: Allows the user to run `cmgui` with the same privileges as user `admin`
  - `readonly`: Allows the user to run `cmgui` without the ability to modify settings.
  - `none`: (default). Prevents the user from using `cmgui`

  A profile setting only takes effect if the certificate for the user is used. User certificates are only persistent for a cluster with a permanent license (page 89), so the administrator should check the license is not a temporary license before attempting to use this feature. Section 8.4 explains the concepts of capabilities, profiles, certificates, and tokens.

2. Save: saves the as-yet-uncommitted Add or Edit operations. When saving an addition:
   - User and group ID numbers are automatically assigned from UID and GID 1000 onwards. Normally Red Hat and similar distributions assign from 500 onwards, while SUSE assigns from 1000 onwards.
   - A home directory is created and a login shell is set. Users with unset passwords cannot log in.
3. Edit: allows user attributes to be modified via a dialog similar to the Add dialog of figure 8.2.
4. Revert: discards unsaved edits that have been made via the Edit button. The reversion goes back to the last save.
5. Remove: removes selected rows of users. By default, along with their home directories.

Group management in cmgui is started by selecting the Groups tab in the Users & Groups pane. Clickable LDAP object entries for regular groups then show up, similar to the user entries already covered. Management of these entries is done with the same button functions as for user management.

8.2 Managing Users And Groups With cmsh

User management tasks as carried out by cmgui in section 8.1, can be carried with the same end results in cmsh too.

A cmsh session is run here in order to cover the functions corresponding to the user management functions of cmgui of section 8.1. These functions are run from within the user mode of cmsh:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% user
[ddnmon61->user]%
```

8.2.1 Adding A User

This part of the session corresponds to the functionality of the Add button operation in section 8.1. In user mode, the process of adding a user maureen to the LDAP directory is started with the add command:

Example

```
[ddnmon61->user]% add user maureen
[ddnmon61->user*[maureen*]]%
```

The cmsh utility helpfully drops into the context of the user just added, and the prompt shows the user name to reflect this. Going into user context would otherwise be done manually by typing use user maureen at the user mode level.

Asterisks in the prompt are a helpful reminder of a modified state, with each asterisk indicating that there is an unsaved, modified property at that asterisk's level.

The `modified` command displays a list of modified objects. This corresponds roughly to the functionality of the `List of Changes` menu option under the `View` menu of the main menu bar of `cmgui`.

Running `show` at this point reveals a user name entry, but empty fields for the other properties of user `maureen`. So the account in preparation, while it is modified, is clearly not yet ready for use:

Example

```
[ddnmon61->user*[maureen*]]% show
Parameter                      Value
------------------------------ --------------------------
Common name
Expiration date                2038/1/1
Group ID
Home directory
Inactive                       0
Last change                    1970/1/1
Login shell
Password                       < not set >
Profile
Revision
Shadow max                     999999
Shadow min                     0
Shadow warning                 7
User ID
User name                      maureen
```

8.2.2 Saving The Modified State

This part of the session corresponds to the functionality of the `Save` button operation in section 8.1.

In section 8.2.1 above, user `maureen` was added. `maureen` now exists as a proposed modification, but has not yet been committed to the LDAP database.

Running the `commit` command now at the `maureen` prompt stores the modified state at the user `maureen` level:

Example

```
[ddnmon61->user*[maureen*]]% commit
[ddnmon61->user[maureen]]% show
Parameter                      Value
------------------------------ ---------------------------
Common name                    maureen
Expiration date                2038/1/1
Group ID                       1002
Home directory                 /home/maureen
Inactive                       0
Last change                    2011/5/30
Login shell                    /bin/bash
Password                       *********
Profile
```

```
Revision
Shadow max                     999999
Shadow min                     0
Shadow warning                 7
User ID                        1002
User name                      maureen
```

If, however, `commit` were to be run at the user mode level without dropping down to the username level, then instead of just that modified user, all modified users and groups would be committed.

When the commit is done, all the empty fields for the user are automatically filled in with defaults based the underlying Linux distribution used. Also, as a security precaution, if an empty field (that is, a "not set") password entry is committed, then a login into the account is not allowed. So, in the example, the account for user `maureen` exists at this stage, but still cannot be logged into until the password is set. Editing passwords and other properties is covered in section 8.2.3.

The default file and directory permissions for the home directory of the user are defined by the umask settings in `/etc/login.defs`, as would be expected if the administrator were to use the standard `useradd` command.

8.2.3 Editing Properties Of Users And Groups

This corresponds roughly to the functionality of the `Edit` button operation in section 8.1.

In the preceding section 8.2.2, a user account `maureen` was made, with an unset password as one of its properties. Logins to accounts with an unset password are refused. The password therefore needs to be set if the account is to function.

Editing Users With `set` And `clear`

The tool used to set user and group properties is the `set` command. Typing `set` and then either using tab to see the possible completions, or following it up with the `enter` key, suggests several parameters that can be set, one of which is `password`:

Example

```
[ddnmon61->user[maureen]]% set
Name:
          set - Set specific user or group property

Usage:
          set <parameter>
          set user <name> <parameter>
          set group <name> <parameter>

Arguments:
          name
               Name of the user or group

Parameters:
          Revision ........... Object revision
          commonname ......... Full user name
          email ............... email
          expirationdate ...... Indicates the date on which the user logi\
                                n will be disabled
          groupid ............. Base group of this user
          homedirectory ....... Home directory
          inactive ............ Indicates the number of days of inactivit\
                                y allowed for the user
          loginshell .......... Login shell
          password ............ Password
          profile ............. Profile for Authorization
          shadowmax ........... Indicates the maximum number of days for \
                                which the user password remains valid.
          shadowmin ........... Indicates the minimum number of days requ\
                                ired between password changes
          shadowwarning ....... The number of days of advance warning giv\
                                en to the user before the user password e\
                                xpires
          surname ............. Surname
          userid .............. User id number
          username ............ User name

[ddnmon61->user[maureen]]%
```

Continuing the session from the end of section 8.2.2, the password can be set at the user context prompt like this:

Example

```
[ddnmon61->user[maureen]]% set password seteca5tr0n0my
[ddnmon61->user*[maureen*]]% commit
[ddnmon61->user[maureen]]%
```

At this point, the account `maureen` is finally ready for use.

The converse of the `set` command is the `clear` command, which clears properties:

Example

```
[ddnmon61->user[maureen]]% clear password; commit
```

Editing Groups With `append` And `removefrom`

While the above commands `set` and `clear` also work with groups, there are two other commands available which suit the special nature of groups. These supplementary commands are `append` and `removefrom`. They are used to add extra users to, and remove extra users from a group.

For example, it may be useful to have a `printer` group so that several users can share access to a printer. For the sake of this example (continuing the session from where it was left off in the preceding), `tim` and `fred` are now added to the LDAP directory, along with a group `printer`:

Example

```
[ddnmon61->user[maureen]]% add user tim; add user fred
[ddnmon61->user*[fred*]]% add group printer
[ddnmon61->user*[printer*]]% commit
[ddnmon61->user*[printer]]%
```

The context switch that takes place in the preceding session should be noted: The context of user maureen was eventually replaced by the context of group printer. As a result, the group printer is committed, but the users tim and fred are not yet committed, which is indicated by the asterisk at the user mode level.

Continuing onwards, to add users to a group the append command is used. A list of users maureen, tim and fred can be added to the group printer like this:

Example

```
[ddnmon61->user[printer]]% append groupmembers maureen tim fred; commit
[ddnmon61->user*[printer]]% show
Parameter                      Value
------------------------------ --------------------------
Group ID                       1003
Group members                  maureen tim fred
Group name                     printer
```

To remove users from a group, the removefrom command is used. A list of specific users, for example, tim and fred, can be removed from a group like this:

```
[ddnmon61->user*[printer]]% removefrom groupmembers tim fred; commit
[ddnmon61->user*[printer]]% show
Parameter                      Value
------------------------------ --------------------------
Group ID                       1003
Group members                  maureen
Group name                     printer
```

The clear command can also be used to clear members—but it also clears all of the extras from the group:

Example

```
[ddnmon61->user[printer]]% clear groupmembers
[ddnmon61->user*[printer*]]% show
Parameter                      Value
------------------------------ --------------------------
Group ID                       1003
Group members
Group name                     printer
```

The commit command is intentionally left out at this point in the session in order to illustrate how reversion is used in the next section.

8.2.4 Reverting To The Unmodified State

This corresponds roughly to the functionality of the Revert button operation in section 8.1.

This section (8.2.4) continues on from the state of the session at the end of section 8.2.3. There, the state of group printers was changed so that the extra added members were removed. This state (the state with no group members showing) was however not yet committed.

The refresh command reverts an uncommitted object back to the last committed state.

This happens at the level of the object it is using. For example, the object that is being handled here is the properties of the group printer. Running revert at a higher level prompt—say, in the user mode level—would revert everything at that level and below. So, in order to affect only the properties of the group printer, the refresh command is used at the group printer level prompt. It then reverts the properties of group printer back to their last committed state, and does not affect other objects:

Example

```
[ddnmon61->user*[printer*]]% refresh
[ddnmon61->user*[printer]]% show
Parameter                      Value
------------------------------ --------------------------
Group ID                       1003
Group members                  maureen
Group name                     printer
```

Here, the user maureen reappears because she was stored in the last save. Also, because only the group printer object has been committed, the asterisk indicates the existence of other uncommitted, modified objects.

8.2.5 Removing A User

Removing a user using cmsh corresponds roughly to the functionality of the Remove button operation in section 8.1.

The remove command removes a user or group. The useful "-r" flag added to the end of the username removes the user's home directory too. For example, within user mode, the command "remove user maureen -r; commit" removes user maureen, along with her home directory. Continuing the session at the end of section 8.2.4 from where it was left off, as follows, shows this result:

Example

```
[ddnmon61->user*[printer]]% use user maureen
[ddnmon61->user*[maureen]]% remove -r;  commit
[ddnmon61->user*]% !ls -d /home/*| grep maureen   #no maureen left behind
[ddnmon61->user*]%
```

8.3 Using An External LDAP Server

Sometimes, an external LDAP server is used to serve the user database. If, instead of just using the database for authentication, the user database is also to be managed, then its LDAP schema must match the DirectMon LDAP schema.

By default, DirectMon runs an LDAP health check using the cmsupport user on the LDAP server. The LDAP health check may need to be modified or disabled by the administrator to prevent spurious health warnings with an external LDAP server:

Modifying Or Disabling The LDAP Healthcheck

Modifying the LDAP healthcheck: To keep a functional LDAP health check with an external LDAP server, a permanent external LDAP user name, for example ldapcheck, can be added. This user can then be set as the parameter for DirectMon's ldap health check object that is used to monitor the LDAP service. An example of a health check object is shown on page 450.

- If user management is not configured to work on CMDaemon for the external LDAP server, then the user management tool that is used with the external LDAP server should be used by the administrator to create the ldapcheck user instead.
- If user management is still being done via CMDaemon, then an example session for configuring the ldap script object to work with the new external LDAP user is (some prompt text elided):

  Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% user
[ddnmon61]% add user ldapcheck
[ddnmon61]% monitoring setup healthconf headnode
[ddnmon61->monitoring->setup[HeadNode]->healthconf]% use ldap
...[HeadNode]->healthconf[ldap]]% set healthcheckparam ldapcheck
...*[HeadNode*]->healthconf*[ldap:ldapcheck*]]% commit
```

Disabling the LDAP healthcheck: Instead of modifying the LDAP healthcheck to work when using an external LDAP server, the DirectMon LDAP health check can be disabled entirely via cmgui or cmsh.

- cmgui: the ldap health check is disabled as follows: Within the Monitoring Configuration resource the Health Check Configuration tabbed pane is selected. All Head Nodes is then selected from the dropdown menu, and the ldap health check is selected from the list of health checks that are displayed. Clicking the edit button then opens Edit Health Check Configuration" dialog (figure 11.29), which has a Disabled checkbox. Ticking this checkbox disables the health check.
- cmsh: the disabled parameter of the ldap health check object is set to yes. The disabled parameter for the ldap health check can be set as follows:

```
[root@ddnmon61 ~]# cmsh -c "monitoring setup healthconf headnode;\
set ldap disabled yes; commit"
```

Configuring The Cluster To Authenticate Against An External LDAP Server

The cluster can be configured in different ways to authenticate against an external LDAP server.

For smaller clusters, a configuration where LDAP clients on all nodes point directly to the external server is recommended. An easy way to set this up is as follows:

- On the head node:
  - In distributions that are:

    * derived from prior to RHEL 6: the URIs in /etc/ldap.conf, and in the image file /cm/images/default-image/etc/ldap.conf are set to point to the external LDAP server.
    * derived from RHEL 6 or higher: the file /etc/ldap.conf does not exist. The files in which the changes then need to be made are /etc/nslcd.conf and /etc/pam_ldap.conf. To implement the changes, the nslcd daemon must then be restarted from its init script.
  - the updateprovisioners command (section 6.2.4) is run to update any other provisioners.

- Then, to update configurations on the regular nodes so that they are able to do LDAP lookups:
  - They can simply be rebooted to pick up the updated configuration, along with the new software image.
  - Alternatively, to avoid a reboot, the imageupdate command (section 6.6.2) can be run to pick up the new software image from a provisioner.

- The CMDaemon configuration file cmd.conf (Appendix C) has LDAP user management directives. These may need to be adjusted:
  - If another LDAP tool is to be used for external LDAP user management instead of cmgui or cmsh, then altering cmd.conf is not required, and DirectMon's user management capabilities do nothing in any case.
  - If, however, system users and groups are to be managed via cmgui or cmsh, then CMDaemon, too, must refer to the external LDAP server instead of the default LDAP server. This configuration change is actually rare, because the external LDAP database schema is usually an existing schema generated outside of DirectMon, and so it is very unlikely to match the DirectMon LDAP database schema. To implement the changes:
    * On the node that is to manage the database, which is normally the head node, the LDAPHost, LDAPUser, LDAPPass, and LDAPSearchDN directives in cmd.conf are changed so that they refer to the external LDAP server.
    * CMDaemon is restarted to enable the new configurations.

For larger clusters the preceding solution can cause issues due to traffic, latency, security and connectivity fault tolerance. If such occur, a better solution is to replicate the external LDAP server onto the head node, hence keeping all cluster authentication local, and making the presence of the external LDAP server unnecessary except for updates. This optimization is described in the next section.

8.3.1 External LDAP Server Replication

This section explains how to set up replication for an external LDAP server to an LDAP server that is local to the cluster, if improved LDAP services are needed. Section 8.3.2 then explains how this can then be made to work with a high availability setup.

Typically, the DirectMon LDAP server is configured as a replica (consumer) to the external LDAP server (provider), with the consumer refreshing its local database at set timed intervals. How the configuration is done varies according to the LDAP server used. The description in this section assumes the provider and consumer both use OpenLDAP.

External LDAP Server Replication: Configuring The Provider

It is advisable to back up any configuration files before editing them.

The provider is assumed to be an external LDAP server, and not necessarily part of the DirectMon cluster. The LDAP TCP ports `389` and `689` may therefore need to be made accessible between the consumer and the provider by changing firewall settings.

If a provider LDAP server is already configured then the following synchronization directives must be in the `slapd.conf` file to allow replication:

```
index entryCSN eq
index entryUUID eq
overlay syncprov
syncprov-checkpoint <ops> <minutes>
syncprov-sessionlog <size>
```

The openldap documentation (`http://www.openldap.org/doc/`) has more on the meanings of these directives. If the values for <*ops*>, <*minutes*>, and <*size*> are not already set, typical values are:

```
syncprov-checkpoint 1000 60
```

and:

```
syncprov-sessionlog 100
```

To allow the consumer to read the provider database, the consumer's access rights need to be configured. In particular, the `userPassword` attribute must be accessible. LDAP servers are often configured to prevent unauthorized users reading the `userPassword` attribute.

Read access to all attributes is available to users with replication privileges. So one way to allow the consumer to read the provider database is to bind it to replication requests.

Sometimes a user for replication requests already exists on the provider, or the root account is used for consumer access. If not, a user for replication access must be configured.

A replication user, `syncuser` with password `secret` can be added to the provider LDAP with adequate rights using the following `syncuser.ldif` file:

```
dn: cn=syncuser,<suffix>
objectClass: person
cn: syncuser
sn: syncuser
userPassword: secret
```

Here, <*suffix*> is the suffix set in `slapd.conf`, which is originally something like `dc=example,dc=com`. The `syncuser` is added using:

```
ldapadd -x -D "cn=root,<suffix>" -W -f syncuser.ldif
```

This prompts for the root password configured in `slapd.conf`.

To verify `syncuser` is in the LDAP database the output of `ldapsearch` can be checked:

```
ldapsearch -x "(sn=syncuser)"
```

To allow access to the `userPassword` attribute for `syncuser` the following lines in `slapd.conf` are changed, from:

```
 access to attrs=userPassword
   by self write
   by anonymous auth
   by * none
```

to:

```
 access to attrs=userPassword
   by self write
   by dn="cn=syncuser,<suffix>" read
   by anonymous auth
   by * none
```

Provider configuration is now complete and the server can be restarted using `/etc/init.d/ldap restart`.

External LDAP Server Replication: Configuring The Consumer(s)
The consumer is an LDAP server on a DirectMon head node. It is configured to replicate with the provider by adding the following lines to `/cm/local/apps/openldap/etc/slapd.conf`:

```
syncrepl rid=2
   provider=ldap://external.ldap.server
   type=refreshOnly
   interval=01:00:00:00
   searchbase=<suffix>
   scope=sub
   schemachecking=off
   binddn="cn=syncuser,<suffix>"
   bindmethod=simple
   credentials=secret
```

Here:

- The `rid=2` value is chosen to avoid conflict with the `rid=1` setting used during high availability configuration (section 8.3.2).
- The provider argument points to the external LDAP server.
- The interval argument (format DD:HH:MM:SS) specifies the time interval before the consumer refreshes the database from the external LDAP. Here, the database is updated once a day.
- The credentials argument specifies the password chosen for the `syncuser` on the external LDAP server.

More on the syncrepl directive can be found in the openldap documentation (http://www.openldap.org/doc/).

The configuration files must also be edited so that:

- The <*suffix*> and rootdn settings in slapd.conf both use the correct <*suffix*> value, as used by the provider.
- The base value in /etc/ldap.conf uses the correct <*suffix*> value as used by the provider. This is set on all DirectMon nodes including the head node(s). If the ldap.conf file does not exist, then the note on page 307 about RHEL versions applies.

Finally, before replication takes place, the consumer database is cleared. This can be done by removing all files, except for the DB_CONFIG file, from under the configured database directory, which by default is at /var/lib/ldap/.

The consumer is restarted using service ldap restart. This replicates the provider's LDAP database, and continues to do so at the specified intervals.

8.3.2 High Availability

No External LDAP Server Case

If the LDAP server is not external—that is, if the DirectMon is set to its high availability configuration, with its LDAP servers running internally, on its own head nodes—then by default LDAP services are provided from both the active and the passive node. The high-availability setting ensures that CMDaemon takes care of any changes needed in the slapd.conf file when a head node changes state from passive to active or vice versa, and also ensures that the active head node propagates its LDAP database changes to the passive node via a syncprov/syncrepl configuration in slapd.conf.

External LDAP Server With No Replication Locally Case

In the case of an external LDAP server being used, but with no local replication involved, no special high-availability configuration is required. The LDAP client configuration in /etc/ldap.conf simply remains the same for both active and passive head nodes, pointing to the external LDAP server. The file /cm/images/default-image/etc/ldap.conf, in each software image also point to the same external LDAP server. If the ldap.conf files referred to here in the head and software images do not exist, then the note on page 307 about RHEL versions applies.

External LDAP Server With Replication Locally Case

In the case of an external LDAP server being used, with the external LDAP provider being replicated to the high-availability cluster, it is generally more efficient for the passive node to have its LDAP database propagated and updated only from the active node to the passive node, and not updated from the external LDAP server.

The configuration should therefore be:

- an active head node that updates its consumer LDAP database from the external provider LDAP server

- a passive head node that updates its LDAP database from the active head node's LDAP database

Although the final configuration is the same, the sequence in which LDAP replication configuration and high availability configuration are done has implications on what configuration files need to be adjusted.

1. For LDAP replication configuration done after high availability configuration, adjusting the new suffix in `/cm/local/apps/openldap/etc/slapd.conf` and in `/etc/ldap.conf` on the passive node to the local cluster suffix suffices as a configuration. If the `ldap.conf` file does not exist, then the note on page 307 about RHEL versions applies.

2. For high availability configuration done after LDAP replication configuration, the initial LDAP configurations and database are propagated to the passive node. To set replication to the passive node from the active node, and not to the passive node from an external server, the provider option in the `syncrepl` directive on the passive node must be changed to point to the active node, and the suffix in `/cm/local/apps/openldap/etc/slapd.conf` on the passive node must be set identical to the head node.

The high availability replication event occurs once only for configuration and database files in DirectMon's high availability system. Configuration changes made on the passive node after the event are therefore persistent.

8.4 Tokens And Profiles

Tokens are used to assign capabilities to users, who are grouped according to their assigned capabilities. A *profile* is the name given to each such group. A profile thus consists of a set of tokens. The profile is stored as part of the authentication certificate which is generated for running authentication operations to the cluster manager for the certificate owner. Authentication is introduced earlier in section 3.3.

Every cluster management operation requires the user's profile to have the relevant tokens for the operation.

Profiles are handled with the `profiles` mode of `cmsh`, or from the `Authorization` resource of `cmgui`. The following preconfigured profiles are available:

Profile name	**Default Tasks Allowed**
`Admin`	all tasks
`CMHealth`	health-related prejob tasks
`Node`	node-related
`Power`	device power
`Readonly`	view-only

Custom profiles can be created to include a custom collection of capabilities in `cmsh` and `cmgui`. Cloning of profiles is also possible from `cmsh`.

The time of expiry for a token cannot be extended after creation. An entirely new token is required after expiry of the old one.

8.4.1 Modifying Profiles

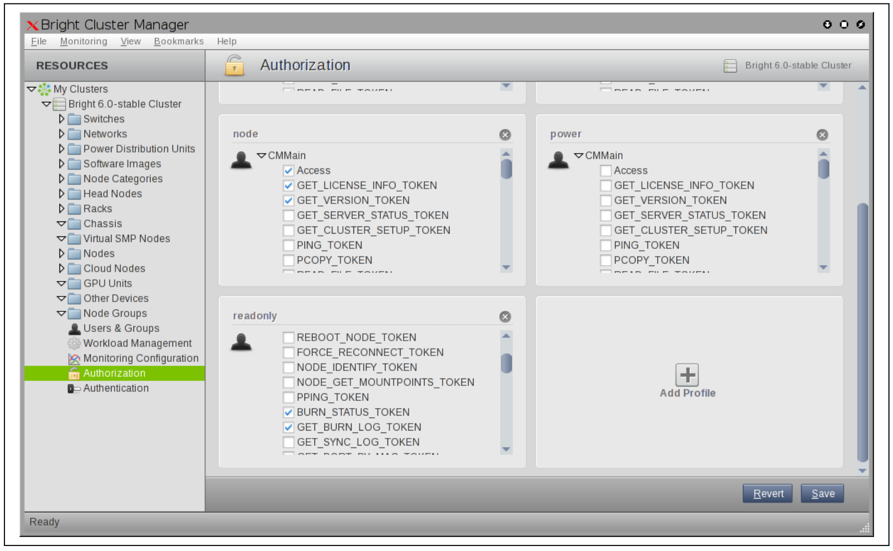
A profile can be modified by adding or removing appropriate tokens to it. For example, the Readonly group by default has access to the burn status and burn log results. Removing the appropriate tokens stops users in that group from seeing these results.

In cmsh the removal can be done from within profile mode as follows:

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% profile use readonly
[...[readonly]]% removefrom tokens burn_status_token get_burn_log_token
[ddnmon61]%->profile*[readonly*]]% commit
```

Tab-completion after typing in removefrom tokens helps in filling in the tokens that can be removed.

In cmgui (figure 8.3), the removal can be done by selecting the Authorization item from the resources tree. In the display pane, the BURN_STATUS_TOKEN and GET_BURN_LOG_TOKEN can have their ticks removed from their checkboxes in the readonly profile group (in the CMDevice subgroup). The changed settings can then be saved.



node
CMMain
- ✓ Access
- ✓ GET_LICENSE_INFO_TOKEN
- ✓ GET_VERSION_TOKEN
- ☐ GET_SERVER_STATUS_TOKEN
- ☐ GET_CLUSTER_SETUP_TOKEN
- ☐ PING_TOKEN
- ☐ PCOPY_TOKEN

power
CMMain
- ☐ Access
- ☐ GET_LICENSE_INFO_TOKEN
- ☐ GET_VERSION_TOKEN
- ☐ GET_SERVER_STATUS_TOKEN
- ☐ GET_CLUSTER_SETUP_TOKEN
- ☐ PING_TOKEN
- ☐ PCOPY_TOKEN

readonly
- ☐ REBOOT_NODE_TOKEN
- ☐ FORCE_RECONNECT_TOKEN
- ☐ NODE_IDENTIFY_TOKEN
- ☐ NODE_GET_MOUNTPOINTS_TOKEN
- ☐ PPING_TOKEN
- ✓ BURN_STATUS_TOKEN
- ✓ GET_BURN_LOG_TOKEN
- ☐ GET_SYNC_LOG_TOKEN

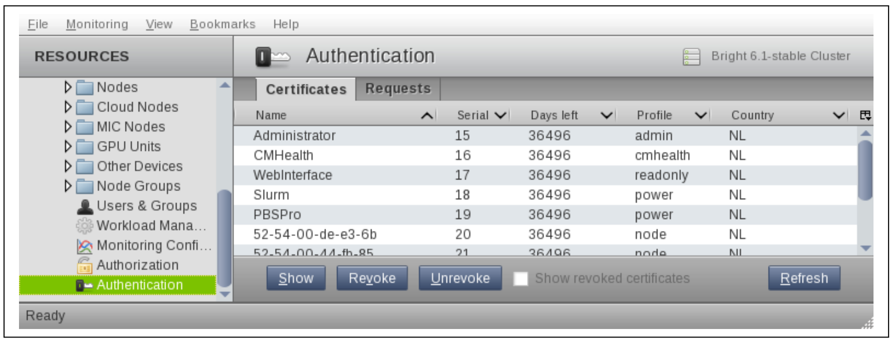
Figure 8.3: cmgui Profile Token Management

8.4.2 Creation Of Custom Certificates With Profiles, For Users Managed By DirectMon's Internal LDAP

CMDaemon tracks the issuance of the certificates generated by the cluster. Certificates that have been issued by the cluster can be listed using cmsh:

```
[root@ddnmon61 ~]# cmsh -c "cert; listcertificates"
Serial num Days left  Profile  Country  Name              Revoked
---------- ---------- -------- -------- ----------------- --------
1          36451      admin    US       Administrator     Yes
10         9          readonly ef       democert          Yes
11         36496      node     NL       52-54-00-de-e3-6b No
12         36496      node     NL       52-54-00-44-fb-85 No
13         36496      admin    UK       otheradmin        No
...
```

They can also be listed using `cmgui`, via the `Authentication` resource, in the `Certificates` tabbed pane (figure 8.4):



Name	Serial	Days left	Profile	Country
Administrator	15	36496	admin	NL
CMHealth	16	36496	cmhealth	NL
WebInterface	17	36496	readonly	NL
Slurm	18	36496	power	NL
PBSPro	19	36496	power	NL
52-54-00-de-e3-6b	20	36496	node	NL
52-54-00-44-fb-85	21	36496	node	NL

Figure 8.4: `cmgui` Certificates Tab

Thus, node certificates that are issued by the node-installer (section 6.4.1) for each node for CMDaemon use are listed.

Custom certificates are also listed. Custom certificates can be generated by a user with the appropriate tokens in their profile, such as `root` with the admin profile. Such a user can create a certificate containing a specified profile, as discussed in the next section, by using:

- `cmsh`: with the `createcertificate` operation from within `cert` mode
- `cmgui`: within the `Users & Groups` resource, within the `Users` tabbed pane, using the `Add` dialog to set the `Management profile`.

Creating A New Certificate For `cmsh` Users

Creating a new certificate in `cmsh` is done from `cert` mode using the `createcertificate` command, which has the following help text:

```
[ddnmon61->cert]% help createcertificate
Name:
          createcertificate - Create a new certificate

Usage:
          createcertificate <key-length> <common-name> <organization> <o\
rganizational-unit> <locality> <state> <country> <profile> <sys-login> <\
days> <key-file> <cert-file>

Arguments:
          key-file
                Path to key file that will be generated

          cert-file
                Path to pem file that will be generated
```

Accordingly, as an example, a certificate file with a read-only profile set to expire in 30 days, to be run with the privileges of user `peter`, can be created with:

Example

```
createcertificate 2048 democert a b c d ef readonly peter 30 /home/peter\
/peterfile.key /home/peter/peterfile.pem

Thu Apr 14 15:10:53 2011 [notice] ddnmon61: New certificate request with\
 ID: 1
[ddnmon61->cert]% createcertificate 2048 democert a b c d ef readonly pe\
ter 30 /home/peter/peterfile.key /home/peter/peterfile.pem
Certificate key written to file: /home/peter/peterfile.key
Certificate pem written to file: /home/peter/peterfile.pem
```

The certificates are owned by the owner generating them, so they are root-owned if `root` was running `cmsh`. This means that user `peter` cannot use them until their ownership is changed to user `peter`:

Example

```
[root@ddnmon61 ~]# cd /home/peter
[root@ddnmon61 peter]# ls -l peterfile.*
-rw------- 1 root root 1704 Jul 19 10:02 peterfile.key
-rw------- 1 root root 1107 Jul 19 10:02 peterfile.pem
[root@ddnmon61 peter]# chown peter:peter peterfile.*
```

Other users must have the certificate ownership changed to their own user names. Users associated with such a certificate can then carry out `cmdaemon` tasks that have a read-only profile, and CMDaemon sees such users as being user `peter`. Two ways of being associated with the certificate are:

1. The paths to the `pem` and `key` files can be set with the `-i` and `-k` options respectively of `cmsh`. For example, in the home directory of `peter`, for the files generated in the preceding session, `cmsh` can be launched with these keys with:

   ```
   [peter@ddnmon61 ~] cmsh -i peterfile.pem -k peterfile.key
   [ddnmon61]% quit
   ```

2. If the `-i` and `-k` options are not used, then `cmsh` searches for default keys. The default keys for `cmsh` are under these paths under `$HOME`, in the following order of priority:

   (a) `.cm/admin.{pem,key}`

   (b) `.cm/cert.{pem,key}`

   (c) `.cm/cmsh/admin.{pem,key}`

   (d) `.cm/cmsh/cert.{pem,key}`

   For example, putting the key pair in the path in (a) overrides any key pairs placed in (b), (c), or (d). The order is based on the path with the lowest directory depth having a higher priority, then the path that is earlier in alphabetical order having a higher priority.

   For example, continuing the preceding session, in the home directory of the user `peter`:

```
[peter@ddnmon61 ~]$ mkdir -p /home/peter/.cm/cmsh
[peter@ddnmon61 ~]$ mv peterfile.* /home/peter/.cm/cmsh/
[peter@ddnmon61 cmsh]$ cd /home/peter/.cm/cmsh
[peter@ddnmon61 cmsh]$ mv peterfile.key admin.key
[peter@ddnmon61 cmsh]$ mv peterfile.pem admin.pem
[peter@ddnmon61 cmsh]$ cd /home/peter
[peter@ddnmon61 ~]$ cmsh
[ddnmon61]%
```

Creating A New Certificate For `cmgui` Users

Like in the case of `cmsh`, a `cmgui` user having a sufficiently privileged tokens profile, such as the `admin` profile, can create a certificate and key file for themselves or another user. This is done by associating a value for the `Management profile` from the `Add` or `Edit` dialog for the user (figure 8.2).

The certificate files, `cert.pem` and `cert.key`, are then placed under these paths under `$HOME`:

```
.cm/cert.{pem,key}
```

Users that authenticate with their user name and password when running `cmgui` use this certificate for their `cmgui` clients, and are then restricted to the set of tasks allowed by their associated profile. However, if other keys do exist, then they are given priority according to the logic in item 2 on page 315.

8.4.3 Creation Of Custom Certificates With Profiles, For Users Managed By An External LDAP

The use of an external LDAP server instead of DirectMon's for user management is described in section 8.3. Generating a certificate for an external LDAP user must be done explicitly in DirectMon. The `external-user-cert.py` script does this, embedding the user and profile in the certificate during the process. It has the following usage:

```
Usage:

For a single profile:
external-user-cert.py <profile> <user> [<user> ... ] --home=<home-prefix>
   [-g <group>] [-p <port>]

For several profiles:
external-user-cert.py --home=<home-prefix> --file=<inputfile>
   [-g <group>] [-p <port>]
                          where lines of <inputfile> have the syntax
                          <profile> <user> [<user> ... ]

The -p <port> option must be set if CMDaemon is using a non-standard SSL
   port
```

Here,

- `<profile>` should be a valid profile
- `<user>` should be an existing user
- `<home-prefix>` is usually `/home`

- `<group>` is a group, such as `wheel`
- `<port>` is a port number, such as the CMDaemon SSL port `8081`

One or more external LDAP user certificates can be created by the script. The certificate files generated are `cert.pem` and `cert.key`. They are stored in the home directory of the user.

For example, a user `spongebob` that is managed on the external server, can have a read-only certificate generated with:

```
# external-user-cert.py readonly spongebob --home=/home
```

If the home directory of `spongebob` is `/home/spongebob`, then the key files that are generated are `/home/spongebob/.cm/cert.key` and `/home/spongebob/.cm/cert.pem`.

Assuming no other keys are used by `cmsh`, a `cmsh` session that runs as user `spongebob` with `readonly` privileges can now be launched:

```
$ module load cmsh
$ cmsh
```

If other keys do exist, then they may be used according to the logic explained in item 2 on page 315.

8.4.4 Logging The Actions Of CMDaemon Users

The following directives allow control over the logging of CMDaemon user actions.

- `CMDaemonAudit`: Enables logging
- `CMDaemonAuditorFile`: Sets log location
- `DisableAuditorForProfiles`: Disables logging for particular profiles

Details on these directives are given in Appendix C.

9 Workload Management

For clusters that have many users and a significant load, a workload management system allows a more efficient use of resources to be enforced for all users than if there were no such system in place. This is because without resource management, there is a tendency for each individual user to over-exploit common resources.

When a workload manager is used, the user submits a batch (i.e. non-interactive) job to it. The workload manager assigns resources to the job, and checks the current availability as well as checking its estimates of the future availability of the cluster resources that the job is asking for. The workload manager then schedules and executes the job based on the assignment criteria that the administrator has set for the workload management system. After the job has finished executing, the job output is delivered back to the user.

Among the hardware resources that can be used for a job are GPUs. Installing CUDA software to enable the use of GPUs is described in section 13.5.

The details of job submission from a user's perspective are covered in the *User Manual*.

Sections 9.1–9.5 cover the installation procedure to get a workload manager up and running.

Sections 9.6–9.7 describe how `cmgui` and `cmsh` are used to view and handle jobs, queues and node drainage.

Section 9.8 shows examples of workload manager assignments handled by DirectMon.

Section 9.9 ends the chapter by describing the power saving features of workload managers.

9.1 Workload Managers Choices

Some workload manager packages are installed by default, others require registration from the distributor before installation.

During cluster installation, a workload manager can be chosen (figure 2.21) for setting up. The choices are:

- None
- Slurm v2.5.7 (default)

- Grid Engine 2011.11p1. An open source development of Sun Grid Engine
- Torque v4.2.2 and its built-in scheduler
- Torque v4.2.2 and the Maui scheduler
- Torque v4.2.2 and the Moab scheduler
- PBS Pro v12.1

These workload managers can also be chosen and set up later using the `wlm-setup` tool (section 9.3).

Besides the preceding workload managers, the installation of the following workload managers is also possible, and described in their own sections:

- openlava (section 9.5.5)
- Load Sharing Facility v7 and v8 (LSF) (section 9.5.6)

9.2 Forcing Jobs To Run In A Workload Management System

Another preliminary step is to consider forcing users to run jobs only within the workload management system. Having jobs run via a workload manager is normally a best practice.

For convenience, the DirectMon defaults to allowing users to login via `ssh` to a node, using the authorized keys files stored in each users directory in `/home` (section 3.3.2). This allows users to run their processes outside the workload management system without restriction. For clusters with a significant load this policy results in a sub-optimal use of resources, since such unplanned-for jobs disturb any already-running jobs.

Disallowing user logins to nodes, so that users have to run their jobs through the workload management system, means that jobs are then distributed to the nodes only according to the planning of the workload manager. If planning is based on sensible assignment criteria, then resources use is optimized—which is the entire aim of a workload management system in the first place.

9.2.1 Disallowing User Logins To Regular Nodes Via `cmsh`

The `usernodelogin` setting of `cmsh` restricts direct user logins from outside the workload manager, and is thus one way of preventing the user from using node resources in an unaccountable manner. The `usernodelogin` setting is applicable to node categories only, rather than to individual nodes.

In `cmsh` the attribute of `usernodelogin` is set from within `category` mode:

Example

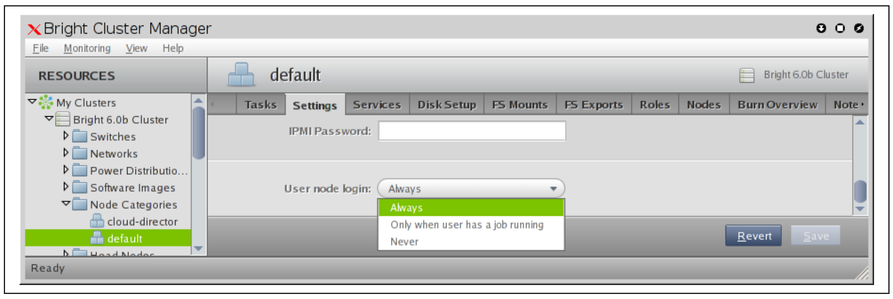
```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% category use default
[ddnmon61->category[default]]% set usernodelogin onlywhenjob
[ddnmon61->category*[default*]]% commit
```

The attributes for usernodelogin are:

- always (the default): This allows all users to ssh directly into a node at any time.
- never: This allows no user other than root to directly ssh into the node.
- onlywhenjob: This allows the user to ssh directly into the node when a job is running on it. It typically also prevents other users from doing a direct ssh into the same node during the job run, since typically the workload manager is set up so that only one job runs per node.

9.2.2 Disallowing User Logins To Regular Nodes Via cmgui

In cmgui, user node login access is set from the Settings tab for a category selected in the Node Categories resource, in the section labeled "User node login" (figure 9.1).



IPMI Password:

User node login: Always

Always
Only when user has a job running
Never

Figure 9.1: Disallowing User Logins To Nodes Via cmgui

9.2.3 Disallowing Other User Processes Outside Of Workload Manager User Processes

Besides disabling user logins, administrators may choose to disable interactive jobs in the workload management system as an additional measure to prevent users from starting jobs on other nodes.

Administrators may also choose to set up scripts that run after job execution. Such scripts can terminate user processes outside the workload manager, as part of a policy, or for general administrative hygiene. These are Epilog scripts and are part of the workload manager.

The workload management system documentation has more on configuring these options.

9.3 Installation Of Workload Managers

Normally the administrator selects a workload manager to be used during DirectMon installation (figure 2.21). A workload manager may however also be added and configured after DirectMon has been installed, using wlm-setup.

9.3.1 Setting Up, Enabling, And Disabling The Workload Manager With wlm-setup

The following options from wlm-setup must be chosen to set up, enable or disable a workload manager for DirectMon:

- `-w` or `--wlmanager`: the **workload manager** option, together with <*name*>, the workload manager name. The value of <*name*> can be one of
  - · `sge`
  - · `torque` (Torque with its built-in scheduler)
  - · `torquemaui` (Torque with the Maui scheduler)
  - · `torquemoab` (Torque with the Moab scheduler)
  - · `slurm`
  - · `pbspro`
  - · `openlava` (This has a prerequisite that the packages must be installed from the openlava site first, as explained in section 9.5.5)

and for the named workload manager, either:

- `-s` or `--setup`: the **setup** option. This does a basic setup of the workload manager, initializing the database for the workload manager, setting up default queues, and enabling the workload manager clients. It can optionally take the further options of section 9.3.2

  or

- `-e` or `--enable`: the **enable** option. Enabling assigns workload manager client roles and default queues on the regular node categories. This option need not be run if the `setup` option has just run. CMDaemon attempts to run workload managers that have been enabled.

  or

- `-d` or `--disable`: the **disable** option, which disables the workload manager

  or

- `-i` or `--image`: the **image** option, which places the job execution daemons in a specific software image, for example when a new node category is added to the cluster. The default image directory is `/cm/images/default-image`.

The `setup` option of `wlm-setup` installs a workload manager package along with its scheduler. Roles, queues, and databases used by the workload manager are initialized on the head. Software images stored on the head node are also set up. However, the nodes themselves are only updated after a reboot, or manually running `imageupdate` (section 6.6.2)—the idea being to avoid an automatic update so that an administrator is encouraged to check via a dry-run if unintended consequences would happen.

The `enable` and `disable` options of `wlm-setup` are the same as those that take place during role assignment (sections 9.4.1 and 9.4.2). These options therefore make CMDaemon enable or disable the workload manager as a service.

For example, setting up SGE can be done as follows:

Example

```
[root@ddnmon61 ~]# wlm-setup -w sge -s
                   Disabling sge services  .....   [  OK  ]
                   Initializing sge setup  .....   [  OK  ]
                   Installing sge qmaster  .....   [  OK  ]
                  Updating image services  .....   [  OK  ]
                  Updating qmaster config  .....   [  OK  ]
               Creating default sge setup  .....   [  OK  ]
                      Setting permissions  .....   [  OK  ]
                    Enabling sge services  .....   [  OK  ]
```

For example, Slurm job daemons can be installed in the node image `new-image` as follows:

Example

```
wlm-setup -w slurm -i /cm/images/new-image
```

If there are provisioning nodes, the `updateprovisioners` command (section 6.2.4) should be run after the software image is changed. The nodes can then simply be rebooted to pick up the new image, or alternatively, to avoid rebooting, the `imageupdate` command (section 6.6.2) places a new image on the node from the provisioner.

9.3.2 Other Options With `wlm-setup`

Other options exist for `wlm-setup`, which are run as further options to the `-s|-setup` option. These are:

- **offload** The offload option (`-o` or `--offload`) deals with setting up, enabling or disabling a workload manager server running on another specified node, other than the default head node:

  Example

  ```
  wlm-setup -w slurm -s -o node003
  ```

- **head as a compute node** This option (`-m` or `--mcompute`) sets the head node to join in as a compute node for job execution tasks in a workload management system. This can be significantly worthwhile on smaller clusters:

  Example

  ```
  wlm-setup -w slurm -s -m
  ```

- **slots** The slots option (`-n` or `--slots`) sets the value of slots (SGE terminology), or `np` (terminology used by other workload managers), and is typically set to the number of cores per node:

  Example

  ```
  wlm-setup -w torque -s -n 4
  ```

9.3.3 Prolog And Epilog Scripts

What Prolog And Epilog Scripts Do

The workload manager runs prolog scripts before job execution, and epilog scripts after job execution. The purpose of these scripts can include:

- checking if a node is ready before submitting a job execution that may use it
- preparing a node in some way to handle the job execution
- cleaning up resources after job execution has ended.

The administrator can run custom prolog or epilog scripts for the queues from CMDaemon for SGE, LSF, or openlava, by setting such scripts in the `cmgui` or `cmsh` front ends. Default epilogs and prologs are created automatically when a new queue is created.

Example

```
[ddnmon61->jobqueue]% add lsf newq
[ddnmon61->jobqueue*(lsf)->newq*] show | grep . | grep -i epilog
Epilog                  /cm/local/apps/cmd/scripts/epilog
Prolog/Epilog user      root
```

For Torque/PBS Pro and Slurm, there are global prolog and epilog scripts, but editing them is not recommended. Indeed, in order to discourage editing them, the scripts cannot be set via the cluster manager front ends, but must be edited directly.

Detailed Workings Of Prolog And Epilog Scripts

Even though it is not recommended, some administrators may nonetheless wish to edit the scripts directly for their own needs, outside of the `cmgui` or `cmsh` front ends. A more detailed explanation of how the prolog scripts work therefore follows:

The prolog scripts have names and function according to their locations. The scripts are placed in two directories:

1. In the **main scripts directory**, at:

   `/cm/local/apps/cmd/scripts/`

   In this directory, a main prolog script, `prolog`, can call a sequence of rc.d-style prolog scripts for a particular workload manager in an rc.d-style directory.

2. **rc.d-style prolog directory**, at:

   `/cm/local/apps/`*<workload manager>*`/var/prologs/`

   In this directory, prolog scripts, if they exist, are stored in a directory path associated with the workload manager name. The names of the scripts have suffixes and prefixes associated with them that make them run in special ways, as follows:

   - **suffixes used in the rc.d-style directory:**

  - `-healthchecker` script runs prior to all jobs
  - `.clouds`: script runs prior to job run in a cloud
  - `-mic`: script runs prior to job run in a MIC
  - `-michost`: script runs prior to job started on a MIC host

- **prefixes used in the rc.d-style directory:**
  - `00-` to
  - `99-`

  Number prefixes determine the order of script execution. This is like for SysV-style `rc.d` names, where scripts with the lower number are run earlier. Hence the terminology "rc.d-style" associated with these prolog scripts.

The script names can therefore look like:

Example

- `00-prolog-healthchecker`
- `10-prolog.clouds`
- `15-prolog-mic`

Return values for the rc.d-style scripts have these meanings:

- `0`: the next script in the directory is run.
- *A non-zero return value*: no further scripts are executed from the rc.d-style directory.

Often, the script in an rc.d-style prolog directory is not a real script but a symlink, with the symlink going to a general script located in the main scripts directory. In that case, this general script is then able to take care of what is expected of the symlink. The name of the symlink, and destination file, usually hints at what the script is expected to do.

For example, the Torque workload manager uses the symlink `10-prolog-healthchecker` within the rc.d-style directory `/cm/local/apps/torque/var/prologs/`. The symlink links to the script `prolog-healthchecker` within the main scripts directory `/cm/local/apps/cmd/scripts/`. In this case, the script is expected to do a health check.

Epilog scripts for Torque and PBS Pro can be set via `/cm/local/apps/{torque|pbspro}/var/spool/mom_priv/` on the head and compute nodes. Also, a Slurm epilog script can be run by setting its path as the value to `Epilog` in `/etc/slurm/slurm.conf`.

The Torque/PBS Pro and Slurm prolog and epilog scripts are global in effect.

Workload Manager Package Configuration For Prolog And Epilog Scripts
Each workload manager package configures prolog- and epilog-related scripts or links during installation, as follows:

- **Slurm**
  - `prolog-healthchecker`: in the main scripts directory, is assigned to `PrologSlurmctld` in `/etc/slurm/slurm.conf`. It runs by default during job execution, and is executed with slurm user permissions.
  - `prolog`: in the main scripts directory, is assigned to the variable `Prolog` in `/etc/slurm/slurm.conf`. The script executes the <*number*>`-prolog{.clouds|-mic|-michost|-healthchecker}` scripts located in the rc.d-style directory, if they exist. By default, none exist. A global epilog script can be run by setting its path as the value to `Epilog` in `/etc/slurm/slurm.conf`
  - `slurm.epilog.clean`: in the `/etc/slurm/` directory, is an example epilog script.
- **SGE, LSF/openlava**
  - `prolog`: in the main scripts directory, executes any <*number*>`-prolog{.clouds|-mic|-michost|-healthchecker}` scripts located in the rc.d-style directory, if they exist.

    It is set by default to execute `10-prolog-healthchecker` in the rc.d-style directory. This in turn is configured as a symlink to `prolog-healthchecker` in the main scripts directory, which is able to handle the execution of scripts in a general manner.
- **PBS Pro/Torque**
  - `mom_priv/prologue`: (with the -ue ending) in the rc.d-style directory, is a symlink to `prolog` in the main scripts directory. This in turn executes any <*number*>`-prolog{.clouds|-mic|-michost|-healthchecker}` scripts located in the rc.d-style directory, if they exist. Similar to this are the `epilogue` scripts in the rc.d-style directory.

9.4 Enabling, Disabling, And Monitoring Workload Managers

After a workload manager package is installed and initialized with `wlm-setup` (section 9.3), it can also be enabled or (if already enabled) disabled, with `wlm-setup`. Enabling and disabling means the workload management services start or stop.

Alternatively, a workload manager can be enabled or disabled by the administrator with `cmgui` or `cmsh`. This is described further on in this section.

In DirectMon™, workload managers can even run concurrently. For example, Slurm, SGE, and Torque can run at the same time in the cluster, for example with one workload manager assigned to one category. Running only one workload manager is however generally recommended for production environments.

From the `cmgui` or `cmsh` point of view a workload manager consists of

- a workload manager server, usually on the head node
- workload manager clients, usually on the compute nodes

For the administrator, enabling or disabling the servers or clients is then simply a matter of assigning or unassigning a particular workload manager server or client role on the head or compute nodes, as deemed appropriate.

The administrator typically also sets up an appropriate workload manager environment module (`slurm`, `sge`, `torque`, `pbspro`, or `openlava`) so that it is loaded up for the end user (section 3.2.3).

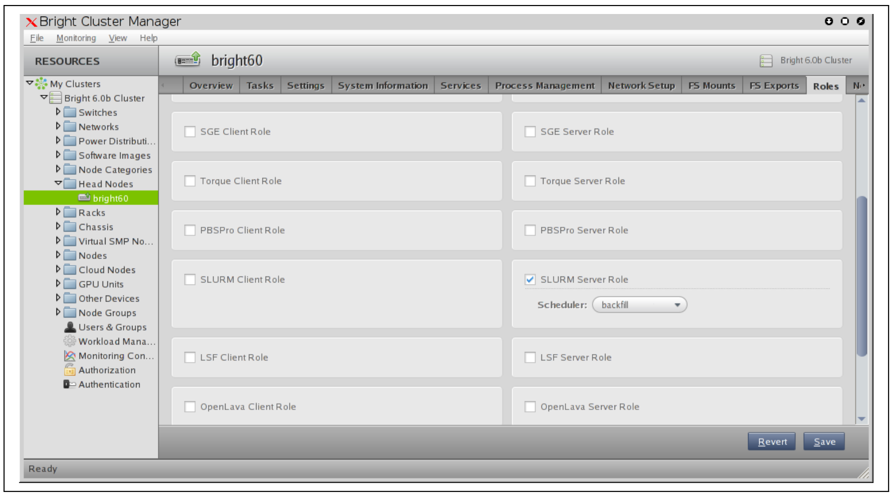
9.4.1 Enabling And Disabling A Workload Manager With `cmgui`

A particular workload manager package may be set up, but the workload manager may not be enabled. This can happen, for example, if using `wlm-setup` (section 9.3) to install the package without enabling it, or for example, if disabling a workload manager that was previously enabled.

The workload manager client and server can be enabled from `cmgui` using the `Roles` tab. Within the `Roles` tab, the properties of the workload manager may be further configured.

Workload Manager Role Assignment To An Individual Node With `cmgui`

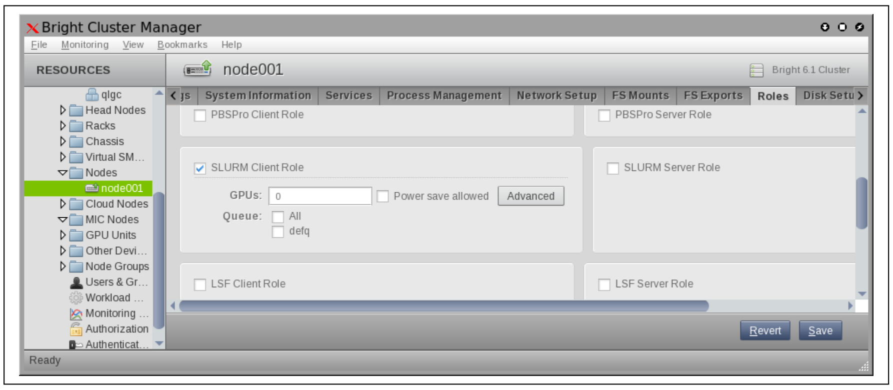
Workload Manager Server Enabling the server on a node can be done by clicking on the "`Head Nodes`" or `Nodes` folder, selecting the node item, and selecting the `Roles` tab to display the possible roles. A workload manager server role is then chosen and its options set. For example, for Torque, the Maui or Moab schedulers must be set as options instead of Torque's built-in scheduler, when enabling Torque with Maui or Moab. The workload manager server role is then saved with the selected options (figure 9.2). For starting it up on non-head nodes (but not for a head node), the `imageupdate` command (section 6.6.2) is then run. The workload manager server process and any associated schedulers then automatically start up.


Bright Cluster Manager
File Monitoring View Help
RESOURCES
bright60
Bright 6.0b Cluster
Overview Tasks Settings System Information Services Process Management Network Setup FS Mounts FS Exports Roles
Revert Save
Ready

SGE Client Role
SGE Server Role
Torque Client Role
Torque Server Role
PBSPro Client Role
PBSPro Server Role
SLURM Client Role
[✓] SLURM Server Role
Scheduler: backfill
LSF Client Role
LSF Server Role
OpenLava Client Role
OpenLava Server Role

Figure 9.2: Workload Management Role Assignment On A Head Node

Workload Manager Client Similarly, the workload manager client process can be enabled on a node or head node by having the workload manager client role assigned in the `Roles` tab. Some basic options can be set for the client role right away in the tab (figure 9.3), and some advanced options can be set by clicking on the `Advanced` button for that role (section 9.5.1, figure 9.6).


Bright Cluster Manager
File Monitoring View Bookmarks Help
RESOURCES
node001
Bright 6.1 Cluster
System Information Services Process Management Network Setup FS Mounts FS Exports Roles Disk Setup
Advanced
Revert Save
Ready

PBSPro Client Role
PBSPro Server Role
[✓] SLURM Client Role
GPUs: 0
Power save allowed
Queue: All
defq
SLURM Server Role
LSF Client Role
LSF Server Role

Figure 9.3: Workload Manager Role Assignment By Node For A Compute Node

Saving the `Roles` pane, and then running `imageupdate` (section 6.6.2), automatically starts up the client process with the options chosen, and managed by CMDaemon.

Workload Manager Role Assignment To A Category With `cmgui`

While workload manager role assignment can be done as described in the preceding text for individual non-head nodes, it is usually more efficient to assign roles using categories due to the large number of compute nodes in typical clusters.

All non-head nodes are by default placed in the `default` category. This means that by default roles in the category are automatically assigned to all non-head nodes, unless by way of exception an individual node configuration overrides the category setting and uses its own role setting instead.

Viewing the possible workload manager roles for the category `default` is done by clicking on the "`Node Categories`" folder, selecting the `default` category, and selecting the `Roles` tab. The appropriate workload manager role is then configured.

For compute nodes, the role assigned is usually a workload manager client. The assigned role for a compute node allows queues and GPUs to be specified, and other parameters depending on the workload manager used.

The workload manager server role can also be assigned to a non-head node. For example, a Torque server role can be carried out by a non-head node. This is the equivalent to the `offload` option of `wlm-setup`. As in the individual node assignment case, for Torque with Maui or Torque with Moab, the Maui scheduler or the Moab scheduler options must be set if these schedulers are to be used.

Saving the roles with their options and then running `imageupdate` (section 6.6.2) automatically starts up the newly-configured workload manager.

Workload Manager Client Basic Role Options With `cmgui`

Each compute node role (workload manager client role) has options that can be set for `GPUs`, `Queues`, and `Slots`.

- `Slots`, from SGE terminology, corresponds in DirectMon to:
  - the `CPUs` setting (a `NodeName` parameter) in Slurm's `slurm.conf` terminology
  - the `np` setting in Torque and PBS Pro terminology,

  and is normally set to the number of cores per node.

  In LSF and openlava, setting the number of slots for the client role to `0` means that the client node does not run jobs on itself, but becomes a submit host, which means it is able to forward jobs to other client nodes.

- `Queues` with a specified name are available in their associated role after they are created. The creation of queues is described in sections 9.6.2 (using `cmgui`) and 9.7.2 (using `cmsh`).

Workload Manager Client Advanced Role Options With `cmgui`

The roles tabbed pane displays advanced role options for Workload manager clients when the `Advanced` button for the client role (figure 9.3) is clicked. These advanced options, as well as the basic options described previously, are managed by CMDaemon and should not be configured separately outside of CMDaemon control. For other options, changes beyond the scope of DirectMon are required. Such changes are then typically managed by setting the options in a workload manager configuration file, and freezing the configuration. Freezing a configuration is done using a CMDaemon directive of the form `FreezeChangesTo`<*workloadmanager*>`Config` (page 589).

Workload Manager Server Basic Role Options With cmgui

Various schedulers can be selected from a menu when configuring the Slurm and Torque workload managers in the basic server role options.

When the checkbox for the role is ticked, an `Advanced` button appears in the basic workload manager server role options. More role options are shown when the `Advanced` button is clicked.

Workload Manager Server Advanced Role Options With cmgui

- Most of the options displayed on clicking the `Advanced` button are installation path or spool path settings.

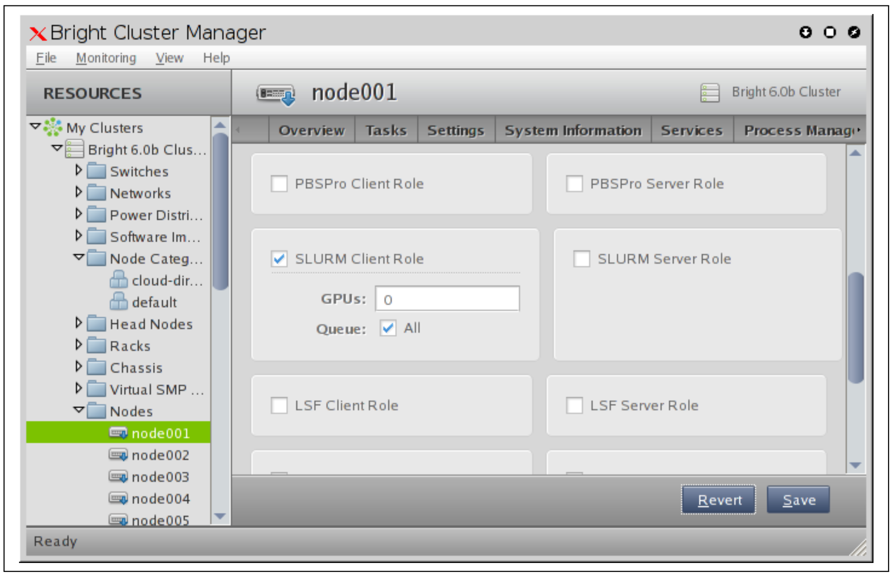
- All server roles also provide the option to toggle the `External Server` checkbox. A ticked checkbox means that the server is no longer managed by DirectMon, but provided by an external device.

  An external Torque server role on a ticked node also means that additional scheduler services—Maui or Moab—are not started, nor are they monitored, on that node. For the ticked node, using the head node is practical and convenient.

  The `cmsh` equivalent of this is described on page 333.

Overriding Category Settings Per Node With cmgui

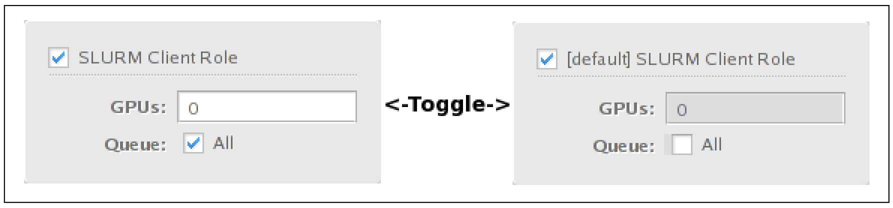
If the role for the individual non-head node is set and saved then it overrides its corresponding category role. In `cmgui` this is done by selecting the particular node device from the `Nodes` folder, then selecting the `Roles` tab. The appropriate workload manager client role can then be configured (figure 9.4).



☐ PBSPro Client Role
☐ PBSPro Server Role
☑ SLURM Client Role
GPUs: 0
Queue: ☑ All
☐ SLURM Server Role
☐ LSF Client Role
☐ LSF Server Role

Figure 9.4: Workload Management Role Assignment For An Individual Node

A useful feature of `cmgui` is that the role displayed for the individual node can be toggled between the category setting and the individual setting by clicking on the role checkbox (figure 9.5). Clicking on the `Save` button of the tabbed pane saves the displayed setting.



GPUs: 0
Queue: [✓ All]

GPUs: 0
Queue: All

Figure 9.5: Workload Management Role Assignment Toggle States For An Individual Node

9.4.2 Enabling And Disabling A Workload Manager With cmsh

A particular workload manager package may be set up, but not enabled. This can happen, for example, if using wlm-setup (section 9.3) on the package without enabling it. The workload manager client and server can be enabled from cmsh by assigning it from within the roles submode. Within the assigned role, the properties of the workload manager may be further configured.

Workload Manager Role Assignment To A Category With cmsh

Workload manager role assignment of a node category is done using category mode, using the category name, and assigning a role from the roles submode:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% category
[ddnmon61->category]% use default
[ddnmon61->category[default]]% roles
[ddnmon61->category[default]->roles]% assign torqueclient
[ddnmon61->category[default]->roles*[torqueclient*]]% commit
[ddnmon61->category[default]->roles[torqueclient]]%
```

After workload manager roles are assigned or unassigned, and after running imageupdate (section 6.6.2) for non-head nodes, the associated workload manager services automatically start up or stop as appropriate.

Workload Manager Role Assignment To An Individual Node With cmsh

In cmsh, assigning a workload manager role to a head node is done in device mode, using master as the device, and assigning the workload manager role from the roles submode:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device
[ddnmon61->device]% use ddnmon61
[ddnmon61->device[ddnmon61]]% roles
[ddnmon61->device[ddnmon61]->roles]% assign torqueserver
[ddnmon61->device*[ddnmon61*]->roles*[torqueserver*]]% commit
[ddnmon61->device[ddnmon61]->roles[torqueserver]]%
```

For regular nodes, role assignment is done via device mode, using the node name, and assigning a role from the roles submode:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device
[ddnmon61->device]% use node001
[ddnmon61->device[node001]]% roles
[ddnmon61->device[node001]->roles]% assign torqueclient
[ddnmon61->device[node001]->roles*[torqueclient*]]% commit
[ddnmon61->device[node001]->roles[torqueclient]]%
```

Role assignment values set in `device` mode have precedence over any role assignment values set in `category` mode for that node. This means, for example, that if a node is originally in a node category with a `torqueclient` role and queues set, then when the node is assigned a `torqueclient` role from `device` mode, its queue properties are empty by default.

Setting Options For Workload Manager Settings With `cmsh`

In the preceding text, it is explained how the workload manager client or server is assigned a role (such as `torqueclient` or `torqueserver`) within the `roles` submode. It is done from within a main mode of `category` or `devices`.

Options for workload managers in general: Whatever the main mode used, the workload manager settings can then be handled with the usual object commands introduced in section 3.5.3. For example, the number of slots can be set for Torque clients as follows:

Example

```
[ddnmon61->category[default]->roles[torqueclient]]% show
Parameter                      Value
------------------------------ ---------------------------------
All Queues                     yes
GPUs                           0
Name                           torqueclient
Queues                         shortq longq
Revision
Slots                          4
Type                           TorqueClientRole
[ddnmon61->category[default]->roles[torqueclient]]% set slots 5
[ddnmon61->category*[default*]->roles*[torqueclient*]]% commit
[ddnmon61->category[default]->roles[torqueclient]]%
```

Options for the Torque variants: In particular, the `Scheduler` parameter can be set from the `torqueserver` role:

Example

```
[ddnmon61->device[ddnmon61]->roles]% set torqueserver scheduler
backfill  builtin   maui      moab      torque
[ddnmon61->device[ddnmon61]->roles]% use torqueserver
[ddnmon61->device[ddnmon61]->roles[torqueserver]]% show
Parameter                      Value
------------------------------ ---------------------------------------
External Server                no
Name                           torqueserver
```

```
Revision
Scheduler                      torque
Type                           TorqueServerRole
[ddnmon61->device[ddnmon61]->roles[torqueserver]]% set scheduler moab
[ddnmon61->device[ddnmon61]->roles[torqueserver*]]% commit
```

In the preceding example, the available schedulers list can be displayed using tab-completion, and the Moab scheduler is then enabled. Running`imageupdate` (section 6.6.2) for non-head nodes ensures that the built-in Torque scheduler stops and the Moab scheduler starts. It is possible to switch to the Maui scheduler with the same approach. The Moab or Maui schedulers do however need to be installed before starting them in this manner.

In `cmgui`, when the `Torque Server Role` is selected in the `Roles` tab of the node, a scheduler service can then be selected from a drop-down menu. Selecting a new scheduler and saving it enables the new scheduler. Running `imageupdate` (section 6.6.2) for non-head nodes stops the previous scheduler and starts the newly selected and enabled scheduler.

Option to set an external workload manager: A workload manager can be set to run as an external server from within a device mode role:

Example

```
[ddnmon61->device[ddnmon61]->roles[torqueserver]]% set externalserver on
[ddnmon61->device[ddnmon61]->roles[torqueserver*]]% commit
```

For convenience, setting it on the head node is recommended.

If an external server is set for the `torqueserver` role, and uses the schedulers Maui or Moab, then these scheduler services are not started or monitored on the device. The device is the head node, if the earlier recommendation for convenience is followed. The default setting of `torque` remains the scheduler.

The `cmgui` equivalent of configuring `externalserver` is described on page 330.

9.4.3 Monitoring The Workload Manager Services

By default, the workload manager services are monitored. The DirectMon attempts to restart the services using the service tools (section 4.12), unless the role for that workload manager service is disabled, or the service has been stopped (using `cmsh` or `cmgui`). Workload manager roles and corresponding services can be disabled using `wlm-setup` (section 9.3.1), `cmgui` (section 9.4.1), or `cmsh` (section 9.4.2).

The daemon service states can be viewed for each node via the shell, `cmsh`, or `cmgui` (section 4.12).

Queue submission and scheduling daemons normally run on the head node. From `cmgui` their states are viewable by clicking on the node folder in the resources tree, then on the node name item, and selecting the `Services` tab (figures 4.17 and 11.5).

The job execution daemons run on compute nodes. Their states are viewable by clicking on the `Nodes` folder, then on the node name item, and selecting the `Services` tab.

From `cmsh` the services states are viewable from within `device` mode, using the `services` command. One-liners from the shell to illustrate this are (output elided):

Example

```
[root@ddnmon61 ~]# cmsh -c "device services node001; status"
        sgeexecd[   UP   ]
[root@ddnmon61 ~]# cmsh -c "device services ddnmon61; status"
                   ...
             sge[   UP   ]
```

9.5 Configuring And Running Individual Workload Managers

DirectMon deals with the various choices of workload managers in as generic a way as possible. This means that not all features of a particular workload manager can be controlled, so that fine-tuning must be done through the workload manager configuration files. Workload manager configuration files that are controlled by DirectMon should normally not be changed directly because DirectMon overwrites them. However, overwriting by CMDaemon is prevented on setting the directive:

FreezeChangesTo<*workload manager*>Config = <true|false>

in `cmd.conf` (Appendix C), where <*workload manager*> takes the value of `Slurm`, `SGE`, `Torque`, `PBSPro`, or `openlava`, as appropriate. The value of the directive defaults to `false`.

A list of configuration files that are changed by CMDaemon, the items changed, and the events causing such a change are listed in appendix L.

A very short guide to some specific workload manager commands that can be used outside of the DirectMon™ system is given in Appendix G.

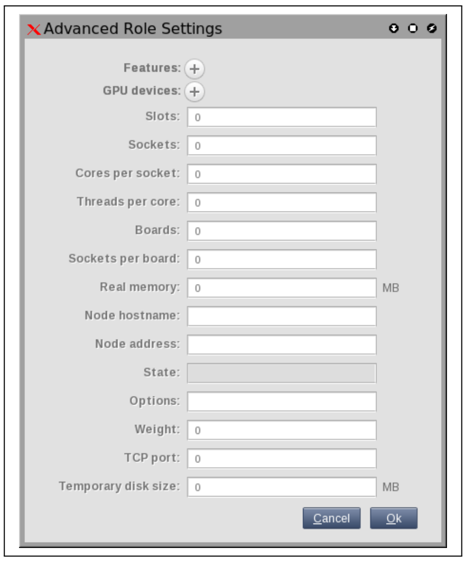
9.5.1 Configuring And Running Slurm

Configuring Slurm

After package setup is done with `wlm-setup` (section 9.3), Slurm software components are installed in `/cm/shared/apps/slurm/current`.

Slurm clients and servers can be configured to some extent via role assignment (sections 9.4.1 and 9.4.2).

For example, the advanced options can be set in `cmgui` (figure 9.6).



Features:	+
GPU devices:	+
Slots:	0
Sockets:	0
Cores per socket:	0
Threads per core:	0
Boards:	0
Sockets per board:	0
Real memory:	0 MB
Node hostname:	
Node address:	
State:	
Options:	
Weight:	0
TCP port:	0
Temporary disk size:	0 MB

Figure 9.6: Workload Manager Role Assignment By Node For A Compute Node—Dialog Example For Advanced Options, For Slurm

Using `cmsh` the advanced option parameters can be set under the `slurmclient` role:

Example

```
[ddnmon61->category[default]->roles[slurmclient]]% set corespersocket 2
```

The option parameter values are used unless they have the value: `0`.

A parameter value of `0` means that the default values of Slurm are used. These usually have the value: `1`.

The advanced options that CMDaemon manages for Slurm are:

`cmgui` **Option**	**Slurm Option**	**Description**

`Features`	`Feature=`<*string*> entry in the file `slurm.conf`	Arbitrary strings can be entered to indicate some characteristics of a node, one string per entry. For example:
		`text1`
		`text2`
		and so on. These become part of the:
		`Feature=text1,text2...`
		attribute to the
		`NodeName=<`*`node name`*`>`
		entry line in `slurm.conf`, as indicated in `man(5) slurm.conf`. The strings also become added attributes to the `GresTypes` entry of that file.
		Default: blank.

`GPU devices`	`File=`<*device*> entries in the file `gres.conf`	This is only implemented if the number of GPUs has been set by the administrator using DirectMon, by setting the basic Slurm client option `GPUs`. The `GPUs` option is seen, for example, in `cmgui`, in figures 9.4, and 9.5.
		Each device detected on the host node can be added as an entry. For example:
		`/dev/nvidia0`
		`/dev/nvidia1`
		and so on. These become part of the:
		`Name=gpu File=`<*device*>
		lines in `gres.conf`, as indicated in `man(5) gres.conf`.
		Default: blank

...continues

...continued

Option	Slurm Option	Description
`Slots`	`CPU`	Number of logical processors on the node. Default: `1`
`Sockets`	`Sockets`	Processor chips on node. Default: `1`
`Cores per socket`	`CoresPerSocket`	Number of cores per socket. Default: `1`
`Threads per core`	`ThreadsPerCore`	Number of logical threads for a single core. Default: `1`
`Boards`	`Boards`	Number of baseboards in a node. Default: `1`
`Sockets per board`	`SocketsPerBoard`	Number of processor chips on baseboard. Default: `1`
`Real memory`	`RealMemory`	Size of real memory on the node, MB. Default: `1`
`Node hostname`	`NodeHostname`	Default: as defined by Slurm's `NodeName` parameter.
`Node address`	`NodeAddr`	Default: as set by Slurm's `NodeHostname` parameter.
`State`	`State`	State of the node with user jobs. Possible Slurm values are: `DOWN`, `DRAIN`, `FAIL`, `FAILING`, and `UNKNOWN`. Default: `UNKNOWN`
`Weight`	`Weight`	The priority of the node for scheduling. Default: `1`
`TCP port`	`Port`	Port that slurmd listens to on the compute node. Default: as defined by `SlurmdPort` parameter. If `SlurmdPort` is not specified during build: Default: `6818`
`Temporary disk size`	`TmpDisk`	Total size of Slurm's temporary filesystem, `TmpFS`, typically `/tmp`, in MB. `TmpFS` is the storage location available to user jobs for temporary storage. Default: `0`

...continues

...continued

Option	Slurm Option	Description
`Options`	`extra options`	Extra options that are added to `slurm.conf`

Further Slurm documentation is available:

- via `man` pages under `/cm/shared/apps/slurm/current/man`.
- as HTML documentation in the directory `/cm/shared/apps/slurm/current/share/doc/slurm-2.5.7/html`
- at the Slurm website at `https://computing.llnl.gov/linux/slurm/slurm.html`

Slurm is set up with reasonable defaults, but administrators familiar with Slurm can reconfigure the configuration file `/cm/shared/apps/slurm/var/etc/slurm.conf` using the JavaScript-based configuration generator in `/cm/shared/apps/slurm/current/share/doc/slurm-2.5.7/html/configurator.html` in a web browser. If the file becomes mangled beyond repair, the original default can be regenerated once again by re-installing the Slurm package, then running the script `/cm/shared/apps/slurm/var/cm/cm-restore-db-password`, and then running `wlm-setup`.

Running Slurm

Slurm can be disabled and enabled with the `wlm-setup tool` (section 9.3) during package installation itself. Alternatively, role assignment and role removal also enables and disables Slurm from `cmgui` (sections 9.4.1 or `cmsh` (section 9.4.2).

The Slurm workload manager runs these daemons:

1. as servers:
   (a) `slurmdbd`: The database that tracks job accounting. It is part of the `slurmdbd` service.
   (b) `slurmctld`: The controller daemon. Monitors Slurm processes, accepts jobs, and assigns resources. It is part of the `slurm` service.
   (c) `munged`: The authentication (client-and-server) daemon. It is part of the `munge` service.
2. as clients:
   (a) `slurmd`: The compute node daemon that monitors and handles tasks allocated by `slurmctld` to the node. It is part of the `slurm` service.
   (b) `slurmstepd`: A temporary process spawned by the `slurmd` compute node daemon to handle Slurm job steps. It is not initiated directly by users or administrators.
   (c) `munged`: The authentication (client-and-server) daemon. It is part of the `munge` service.

Logs for the daemons are saved on the node that they run on. Accordingly, the locations are:

- `/var/log/slurmdbd`
- `/var/log/slurmd`
- `/var/log/slurmctld`
- `/var/log/munge/munged.log`

9.5.2 Configuring And Running SGE

Configuring SGE

After installation and initialization, SGE has reasonable defaults, with `$SGE_ROOT` set to `/cm/shared/apps/sge/current`.

Administrators familiar with SGE can reconfigure it using the template files in `/cm/shared/apps/sge/var/cm/templates/`, which define the queues, host-groups and parallel environments that the `wlm-setup` utility uses. To configure the head node for use with SGE, the `install_qmaster` wrapper script under `$SGE_ROOT` is run. To configure a software image for use with SGE the `install_execd` wrapper script under `$SGE_ROOT` is run.

By default, the SGE application is installed in `/cm/shared/apps/sge/current`, the SGE documentation in `/cm/shared/docs/sge/current`, and job examples in `/cm/shared/examples/workload/sge/jobscripts/`.

Running SGE

SGE can be disabled and enabled with the `wlm-setup tool` (section 9.3) during package installation itself. Alternatively, role assignment and role removal also enables and disables SGE from `cmgui` (sections 9.4.1 or `cmsh` (section 9.4.2).

The SGE workload manager runs the following two daemons:

1. an `sge_qmaster` daemon running on the head node. This handles queue submissions and schedules them according to criteria set by the administrator.

2. an `sge_execd` execution daemon running on each compute node. This accepts, manages, and returns the results of the jobs on the compute nodes.

Messages from the qmaster daemon are logged under:

```
/cm/shared/apps/sge/current/default/spool/
```

On the associated compute nodes the execution log `messages` exists (alongside other job tracking files and directories) at:

`/cm/local/apps/sge/var/spool/node<`*number*`>/messages`

where `node<`*number*`>` is the node name, for example:

`node001, node002 ...`

9.5.3 Configuring And Running Torque

Torque is a resource manager controlling the jobs and compute nodes it talks with. Torque has its own built-in scheduler, but since this is quite basic, the open source Maui and the proprietary Moab schedulers are recommended alternatives.

Configuring Torque

The Torque package is installed, but not set up by default on DirectMon™. If it is not set up during installation (figure 2.21), it can be set up later on using the `wlm-setup` tool (section 9.3).

The execution daemon, `pbs_mom` is already in the software images by default and does not need to be installed, even if Maui or Moab are added.

The Torque services can be enabled and disabled via role assignment as described in section 9.4. Resources such as the number of GPUs are configured in that section too for the node or node category in order to set up the corresponding resource definitions in the Torque configuration files.

Torque software components are installed in `/cm/shared/apps/torque/current`, also referred to as the `PBS_HOME`. The `torque` environment module, which sets `$PBS_HOME` and other environment variables, must be loaded in order to submit jobs to Torque. The `man` pages for Torque are then accessible, with `$MANPATH` set to `$PBS_HOME/man`.

Torque documentation is available at the Adaptive Computing website at `http://www.adaptivecomputing.com/resources/docs/`, including various versions of the the Torque administrator manual.

Torque examples are available under `/cm/shared/examples/workload/torque/jobscripts/`

Installing The Maui Scheduler Package

If Maui is to be installed, the Maui scheduler source version 3.3.1 must be picked up from the Adaptive Computing website at `http://www.adaptivecomputing.com/support/download-center/maui-cluster-scheduler/` (registration required). The source file must be installed over the zero-sized placeholder file on the head node at `/usr/src/redhat/SOURCES/maui-3.3.1.tar.gz`.

Maui documentation is available at `http://docs.adaptivecomputing.com/maui/index.php`.

The RPM file is built from the source on the head node for DirectMon™ using:

```
rpmbuild -bb /usr/src/redhat/SPECS/maui.spec
```

and the installation is done with:

```
rpm -i /usr/src/redhat/RPMS/x86_64/maui-3.3.1-58_cm.x86_64.rpm
```

The exact version of the rpm file to install may differ from the version shown here. The version freshly generated by the `rpmbuild` process is what should be used.

CMDaemon needs to be aware the scheduler is Maui for nodes in a Torque server role. This can be configured using `wlm-setup` to enable the `torquemaui` option (as shown in section 9.3.1), or using `cmgui` to set the scheduler from the `Roles` tab (as shown in section 9.4.1), or using `cmsh` to assign the scheduler from within the assigned `torqueserver` role (as shown in section 9.4.2).

Installing The Moab Scheduler Package

Moab is not installed by default in DirectMon™. Moab and related products must be purchased from Adaptive Computing. DirectMon™ expects the init script for Moab to be located in the file `/etc/init.d/moab`. Other files such as the Moab binaries and libraries can be installed in `/cm/shared/apps/moab/<moab.version>` or any other preferred location.

The Moab install documentation should be followed carefully. Issues needing attention during the Moab installation, at least for the Moab version 7 series, include the following: if using Torque and if the `--with-torque=`<*path*> option is not specified, then library paths for Torque and Moab must be set in Moab's `ldconfig` configuration file.

This can be by appending them with, for example:

```
appsdir="/cm/shared/apps"
ldsodir="/etc/ld.so.conf.d"
echo $appsdir/torque/current/lib >> $ldsodir/moab.conf
echo $appsdir/moab/<version>/lib >> $ldsodir/moab.conf
ldconfig
```

Alternatively, the following lines can be added at the beginning of the Moab init.d script:

```
export LD_LIBRARY_PATH=/cm/shared/apps/torque/current/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/cm/shared/apps/moab/<version>/lib:$LD_LIBRARY_PATH
```

CMDaemon needs to be aware the scheduler is Moab for nodes in a Torque server role. This can be configured using `wlm-setup` to enable the `torquemoab` option (section 9.3.1), or using `cmgui` to set the scheduler from the `Roles` tab (as shown in section 9.4.1) or using `cmsh` to assign the scheduler from within the assigned `torqueserver` role (as shown in section 9.4.2).

Running Torque And Schedulers

The Torque resource manager runs the following daemons:

1. a `pbs_server` daemon. This handles submissions acceptance, and talks to the execution daemons on the compute nodes when sending and receiving jobs. It writes logs to the `/cm/shared/apps/torque/var/spool/server_logs` directory on its node. Queues for this service are configured with the `qmgr` command.

2. a `pbs_mom` execution daemon running on the nodes that are assigned the compute role. This accepts, manages, and returns the results of jobs on the compute nodes. It writes logs to the `/cm/local/apps/torque/var/spool/mom_logs` directory on the compute nodes.

3. a `trqauthd` authentication daemon, in version 4.0 and above. This service is used by Torque clients to authorize user connections to the pbs_server server daemon. When `torqueclient` or

torqueserver roles are assigned in cmsh/cmgui, the authentication service is added automatically to relevant nodes so that user commands can be executed on those nodes. This allows the schedulers health check to execute Torque user utilities on compute nodes.

If the torqueserver is running as an external server (pages 330, 333), then trqauthd must run on the external server too.

Job submission from a non-computing node: The trqauthd daemon is also needed by Torque users on a job submit node, if the submit node does not have a Torque role (torqueclient or torqueserver) assigned to it already. For example, some customized clusters use a category of node that is a login node. So, there may be login nodes named <*login01*>, <*login02*>, and so on. The login node does no computation itself, but users simply log into it to submit jobs. In that case, a minimal way to change the configuration of the node is as follows:

(a) The submit hosts are added to the Torque configuration database so that Torque knows about them:

Example

```
qmgr -c 'set server submit_hosts = <login01>'
qmgr -c 'set server submit_hosts += <login02>'
qmgr -c 'set server submit_hosts += <login03>'
...
```

The names become part of the qmgr database configuration. This configuration can be viewed by running:

```
qmgr -c "p s"
```

(b) The /etc/init.d/trqauthd script is copied from the head node to the login/submit node software image. Running imageupdate then places the script on the running login/submit nodes. Adding the trqauthd service to the list of services being monitored is recommended. Adding a service so that it is monitored and automatically restarted if needed is explained in section 4.12.

Jobs are however not be executed unless the scheduler daemon is also running. This typically runs on the head node and schedules jobs for compute nodes according to criteria set by the administrator. The possible scheduler daemons for Torque are:

- pbs_sched if Torque's built-in scheduler itself is used. It writes logs to the /cm/shared/apps/torque/var/spool/sched_logs directory.

- maui if the Maui scheduler is used. It writes logs to /cm/shared/apps/maui/var/spool/log.

- moab if the Moab scheduler is used. Log files are written to the spool directory. For example: /cm/shared/apps/moab/moab.version/spool/moab.log if Moab is installed in /cm/shared/apps/moab/<moab.version>.

9.5.4 Configuring And Running PBS Pro

Configuring PBS Pro

PBS Pro can be selected for installation during DirectMon™ installation, at the point when a workload manager must be selected (figure 2.21). It can also be installed later on, when the cluster is already set up. In either case, it is offered under a 90-day trial license.

To install and initialize PBS Pro after DirectMon has already been set up without PBS Pro, the wlm-setup tool (section 9.3) is used.

PBS Pro software components are then installed and initialized by default in /cm/shared/apps/pbspro/current, also referred to as the PBS_HOME. Users must load the pbspro environment module, which sets PBS_HOME and other environment variables, in order to use PBS Pro.

PBS Pro documentation is available at http://www.pbsworks.com/SupportDocuments.aspx.

By default, PBS Pro examples are available under the directory /cm/shared/examples/workload/pbspro/jobscripts/

Some PBS Pro configuration under DirectMon can be done using roles:

- In cmgui, the Roles tabbed pane allows the configuration of PBS Pro client and server roles.
  - For the PBS Pro server role, ticking and saving the check box state enables the role.
    - * Using the Advanced button for the server role, its installation path and server spool path can be specified.
  - For the PBS Pro client role, ticking and saving the check box state enables the role. The number of slots and GPUs can be specified, and all or just some of the queues can be selected.
    - * Using the Advanced button for the client role, its client spool path, properties, and GPU devices can be specified.
- In cmsh the client and server roles can be managed for the individual nodes in device mode, or managed for a node category in category mode.

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device roles ddnmon61
[ddnmon61->device[ddnmon61]->roles]% use pbsproserver
[ddnmon61->device[ddnmon61]->roles[pbsproserver]]% show
Parameter                      Value
-------------------------      ---------------------------------
External Server                no
Name                           pbsproserver
Provisioning associations      <0 internally used>
Revision
Type                           PbsProServerRole
```

```
Prefix                         /cm/shared/apps/pbspro/current
Spool                          /cm/shared/apps/pbspro/var/spool
...
[ddnmon61->device[ddnmon61]->roles[pbsproserver]]% set prefix \
                                    /srv/pbspro/current; commit
```

Further configuration of PBS Pro is done using its `qmgr` command and is covered in the PBS Pro documentation.

Running PBS Pro

PBS Pro runs the following three daemons:

1. a `pbs_server` daemon running, typically on the head node. This handles submissions acceptance, and talks to the execution daemons on the compute nodes when sending and receiving jobs. It writes logs to the `/cm/shared/apps/pbspro/var/spool/server_logs/` directory on its node. Queues for this service are configured with the `qmgr` command.

2. a `pbs_sched` scheduler daemon, also typically running on the head node. It writes logs to the `/cm/shared/apps/pbspro/var/spool/sched_logs/` directory.

3. a `pbs_mom` execution daemon running on each compute node. This accepts, manages, and returns the results of jobs on the compute nodes. It writes logs to `/cm/shared/apps/pbspro/var/spool/mom_logs/` on the head node, and to `/cm/local/apps/pbspro/var/spool/mom_logs/` on the compute nodes.

9.5.5 Installing, Configuring And Running openlava

Prerequisites To Installing openlava

In DirectMon DirectMon™, the RPM packages for version 2.0 of openlava must be picked up from the openlava site, at `http://www.openlava.org/download/download.html`, as a prerequisite to installation.

These packages are outside the standard YUM repositories accessible to the cluster. Installing standalone RPM packages onto the cluster is described generally in sections 10.2 and 10.4.

In the specific case of openlava, the packages can be installed directly from the package HTTP URL with the `rpm` command, or they can be downloaded first and then installed from their local path with the `yum install` command (standard `yum install` currently does not accept a package from a URL).

The openlava client and server packages can be installed on the nodes as follows:

- On a head node, the `openlava-server` package can be installed with `rpm` as:

  ```
  rpm -ivhp <package HTTP URL>
  ```

  Or, using `yum` instead of `rpm`:

  ```
  yum install <package path>
  ```

  To install to an existing failover setup, the installation must be done on the active head node.

- For the regular nodes, the `openlava-client` package goes into the appropriate software image. For example, for the default image, `default-image`, using `rpm`:

  ```
  rpm -ivhp <package HTTP URL> --root=/cm/images/default-image
  ```

  Or, using `yum` instead of `rpm`:

  ```
  yum --installroot=/cm/images/default-image install
  <package path>
  ```

  The `updateprovisioners` (section 6.2.4) and `imageupdate` (section 6.6.2) commands can be used to implement the changes on all running nodes.

Installing openlava

After package installation has been carried out, installing openlava on the cluster can be done as described in `wlm-setup` (section 9.3). The command can be run as:

```
wlm-setup -w openlava -s
```

Configuring openlava

After installation of openlava using `wlm-setup` (section 9.3), openlava software components are installed in `/cm/shared/apps/openlava/current`.

Documentation for openlava is available via `man` pages under `/cm/shared/apps/openlava/current/share/man`. Further documentation is available online at: `http://www.openlava.org/documentation/documentation.html`.

Reasonable defaults are used for openlava, but administrators familiar with openlava can reconfigure it by modifying the configuration files under `/cm/shared/apps/openlava/var/etc/`.

The openlava environment module can be set to automatically load for users as described in section 3.2.3.

Within the openlava module environment, running the openlava command `bhosts` displays something like:

```
[root@ddnmon61 ~]# bhosts
HOST_NAME     STATUS  JL/U  MAX  NJOBS    RUN  SSUSP  USUSP  RSV
node001           ok     -    2      0      0      0      0    0
node002           ok     -    2      0      0      0      0    0
node003      unavail     -    1      0      0      0      0    0
node004       closed     -    1      0      0      0      0    0
node005           ok     -    1      0      0      0      0    0
ddnmon61          ok     -    0      0      0      0      0    0
```

Running openlava

openlava can be disabled and enabled with the `wlm-setup tool` (section 9.3) during package installation itself. Alternatively, role assignment and role removal also enables and disables openlava from `cmgui` (sections 9.4.1 or `cmsh` (section 9.4.2).

All compute nodes (sometimes called passive openlava masters in openlava terminology) have the same set of daemons. One node (it can be a regular node or it can be a head node) takes on the role of active openlava master. It is enough to assign only the `openlavaserver` role to each node or node category that is running the daemons. The nodes used to submit an openlava job take on the `openlavaclient` role, but do not have the daemons running on them.

The openlava workload manager daemons are:

- daemons that can run on the active openlava master node:
  - `mbatchd`: The active openlava master batch daemon. Started by the batch daemon on a passive openlava master. Responsible for the overall state of `openlava` jobs. Receives job submissions, and information query requests. Manages jobs held in queues. Dispatches jobs to hosts as determined by `mbschd`. It is a part of the `openlava` service.
  - `mbschd`: The batch scheduler daemon. Works with `mbatchd`. Started by `mbatchd`. Makes scheduling decisions based on job requirements and policies.
- daemons that can run on both on the active and passive openlava master nodes):
  - `sbatchd`: The standard batch `openlava` daemon. It is a part of the `openlava` service.
  - `lim`: The load information manager daemon. Collects built-in load indices that reflect the load situations of CPU, memory, disk space, I/O, and interactive activities on individual hosts. It is a part of the `openlava` service.
  - `res`: The remote execution manager daemon. Executes user tasks on remote hosts. It is a part of the `openlava` service.
  - `pim`: The process information manager daemon. Is responsible for monitoring all jobs and monitoring every process created for all jobs running on the node.

`lim`, `res`, and `sbatchd` daemons maintain log files in:

```
/cm/local/apps/openlava/var/log
```

Batch accounting files are shared by all potential active openlava master nodes. The spools are stored in the `work` directory:

`/cm/shared/apps/openlava/var/work/logdir`.

9.5.6 Installing, Configuring, And Running LSF

IBM prefers to distribute LSF directly, which is why it is not available as an option in the installation carried out by DirectMon™ in Chapter 2.

Installing LSF

The workload manager LSF version 7 or 8 is integrated into DirectMon™ with the following steps:

1. The LSF compressed tar file that contains the LSF software is unpacked. The top level of the unpacked directory then has the installation file, `lsfinstall`. This should not be run yet.

In case of an existing failover setup, the installation is done on the active head node.

2. Some configuration changes are needed before running the `lsfinstall` script in order to create the right directories and paths during installation. The LSF `install.config` file, in the top level of the unpacked files, requires edits for at least the following entries:

   ```
   LSF_TOP="/cm/shared/apps/lsf/"
   LSF_CLUSTER_NAME="<cluster name>"
   LSF_MASTER_LIST="<head node list>"
   LSF_ENTITLEMENT_FILE="<license path>"
   ```

   Here:

   - The value of <*`cluster name`*> should not be the same as any hostname on the cluster.
   - `LSF_ENTITLEMENT_FILE` is `LSF_LICENSE` in LSF version 7.

3. The `lsfinstall` script is then run from the directory it was unpacked into:

   ```
   ./lsfinstall -f install.config
   ```

   and the instructions are followed.

4. - LSF is added to the `chkconfig` system on each LSF server host (by default the head node). This is done by running the `hostsetup` script, as root, from the `install` directory, by default at `/cm/shared/apps/lsf/<version>/install`:

     ```
     ./hostsetup --top=/cm/shared/apps/lsf/ --boot=y
     ```

     The same is run on the passive head node in case of an existing failover setup.

   - To verify that installation is proceeding as intended so far, the status can be checked with:

     ```
     [root@ddnmon61 ~]# /etc/init.d/lsf status
     Show status of the LSF subsystem
     lim (pid 21138) is running...
     res (pid 17381) is running...
     sbatchd (pid 17383) is running...
     ```

     while default queues can be seen by running (continuing in the same shell):

     ```
     [root@ddnmon61 ~]# . /cm/shared/apps/lsf/var/conf/profile.lsf
     [root@ddnmon61 ~]# bqueues
     ```

5. The LSF startup script is installed on to the software images, It should not be added via `hostsetup` or `chkconfig`. One way to copy over the files required is:

   ```
   for image in $(find /cm/images/ -mindepth 1 -maxdepth 1 -type d)
   do cp /etc/init.d/lsf $image/etc/init.d/; done
   ```

6. Optionally, the LSF environment can be added to `.bashrc` with:

   ```
   . /cm/shared/apps/lsf/var/conf/profile.lsf
   ```

7. The LSF master server role is added to the head node (`ddnmon61` in the following) with:

   ```
   cmsh -c "device roles ddnmon61; assign lsfserver; commit"
   ```

   In case of an existing failover setup, role assignment should be repeated on the passive head node, say `head2`.

   Example

   ```
   cmsh -c "device roles head2; assign lsfserver; commit"
   ```

8. The LSF client role is added to each node category containing LFS nodes. If the only category is `default`, then the command run is:

   ```
   cmsh -c "category roles default; assign lsfclient; set allqueues\
    yes; commit"
   ```

9. Additional NFS entries for the export path of LFS are configured for the head node and for LSF node categories. If the path is already NFS exported, for example by using the shared directory `/cm/shared`, then this step is unnecessary. In `cmgui` the NFS path can be set in the LSF client role from the "`FS Exports`" tab for a "`Head Node`", or "`Node Category`", or "`Nodes`" item (section 4.11.1).

10. The nodes are then rebooted, and the LSF command `bhosts` then shows a display like:

    Example

    ```
    [root@ddnmon61 ~]# bhosts
    HOST_NAME   STATUS  JL/U   MAX  NJOBS   RUN  SSUSP  USUSP   RSV
    ddnmon61        ok     -     2      0     0      0      0     0
    head2           ok     -     0      0     0      0      0     0
    node001         ok     -     1      0     0      0      0     0
    node002         ok     -     1      0     0      0      0     0
    node003    unavail     -     1      0     0      0      0     0
    node004     closed     -     1      0     0      0      0     0
    node005         ok     -     1      0     0      0      0     0
    ```

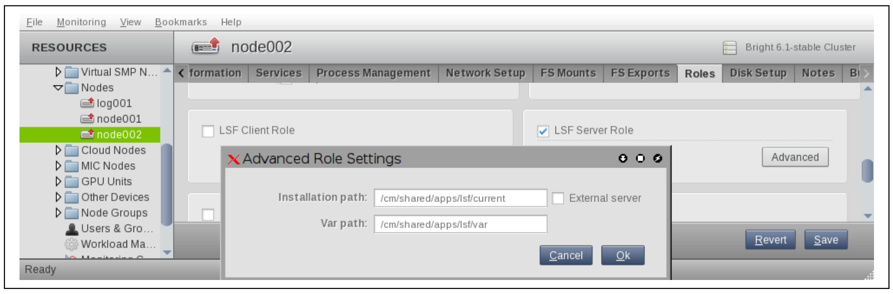
Configuring LSF

LSF server configuration: After installation, the following CMDaemon settings can be specified for the LSF server role:

- `prefix`: this sets the path to the root of the LSF installation.
- `var`: this sets the path to the `var` directory of LSF

These settings can be specified as follows:

- Within `cmgui`: For a particular category in the `Node Category` resource, or for a particular regular node in the `Nodes` resource, or for a head node in the `Head nodes` resource, the `Roles` tabbed pane can be selected. Within the `Roles` tabbed pane, in the LSF Server Role tile, clicking the `Advanced` button brings up a pop up window. This allows the `prefix` and `var` options to be specified. An example can be seen for a regular `node002` in the LSF server role in figure 9.7.



Installation path: /cm/shared/apps/lsf/current
External server
Var path: /cm/shared/apps/lsf/var

Figure 9.7: `cmgui` access to LSF path settings via roles tab

- Within `cmsh`: For a particular category in the `category` mode, or a particular device in the `device` mode, the `roles` submode is chosen. Within the `roles` submode, the `lsfserver` parameter can be set. The following example shows how to set the LSF `prefix` parameter for the `default` category.

  Example

```
[root@ddnmon61~]# cmsh
[ddnmon61]% category use default; roles
[ddnmon61->category[default]->roles]% use lsfserver
[ddnmon61->...[lsfserver]]% set prefix /cm/shared/apps/lsf/current2
[ddnmon61->...[lsfserver*]]% commit
```

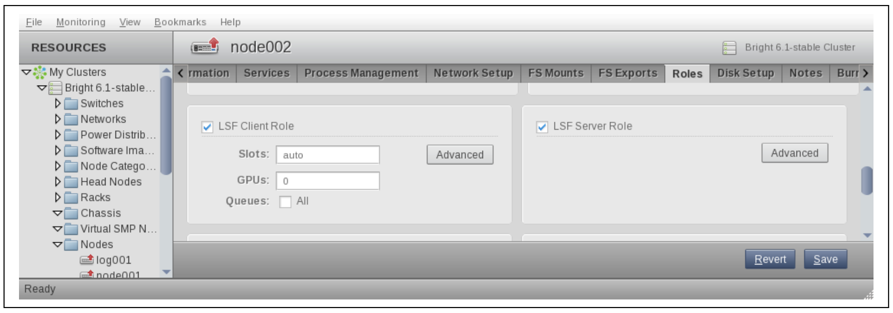
LSF client configuration: After installation, the following CMDaemon settings can be specified for the LSF client role:

- **Queues** By default, the LSF installation procedure that is listed starting on page 346 makes a node accept all queues in steps 7 and 8 by using "`set allqueues yes`". These default queues are displayed in figure 9.8.

  A restricted list of queues named "`qname1`", "`qname2`" and so on can be set using a command syntax like this instead:

```
set queues <qname1> [<qname2> ...]
```

  Alternatively, these, and other queues can be added or deleted using `cmgui` (section 9.6.2) or `cmsh` (section 9.7.2).



☑ LSF Client Role

Slots: auto

GPUs: 0

Queues: ☐ All

☑ LSF Server Role

Figure 9.8: `cmgui` access to LSF configuration options via roles tab

- **Options** The options that can be set via figure 9.8 are:
  - **Slots**: The number of CPUs per node. By default LSF tries to determine the value automatically. If the number of slots is set to `0`, then the node becomes an LSF submit host, so that no jobs can run on the host, but users can submit their jobs from this host using the LSF utilities.
  - **GPUs**: The number of GPUs per node
  - **Queues**: `All` queues.
  - **Advanced button**: This brings up a dialogue that allows GPU devices to be added.

  From `cmsh` these properties are accessible from within the appropriate node or category `roles` submode (section 9.4.2).

Failover: If LSF is set up before setting up failover (section 15.2) for the DirectMon head nodes, then nothing special or extra needs to be done when setting up failover for LSF.

Data-Aware Scheduling With `cmsub`: The ability to use `cmsub` to submit jobs to cloud nodes, as described in the chapter *Workload Management* in the DirectMon™ *User Manual*, is called *data-aware scheduling*. Data-aware scheduling is enabled for LSF as follows:

1. (a) If LSF is not under the control of CMDaemon because the directive `FreezeChangesToLSFConfig` (Appendix C) is active, and if the `cloudtransfers` queue does not exist in LSF, then the queue should be added manually into the following configuration file:

   `$LSF_ENVDIR/lsbatch/<clustername>/configdir/lsb.queues`

   The default value for `$LSF_ENVDIR` is `/cm/shared/apps/openlava/var/etc/`.

   (b) If LSF is controlled by CMDaemon, then the `cloudtransfers` queue is created automatically when a cloud provider is added. The parameter `PRE_EXEC` for the queue is set to the path of the script `/cm/local/apps/cmd/scripts/prolog.clouds`. The script runs when transferring input and output data during a user job, and has a similar function to the regular prolog scripts (page 354).

2. In order to run the cloud prolog script as the root user, /etc/lsf.sudoers must be configured. By default this file does not exist. Its ownership must be root. It must also have read-write permissions for root only, i.e. its Unix permissions can be set with chmod 600 /etc/lsd.sudoers. To execute the PRE_EXEC script as root, LSB_PRE_POST_EXEC_USER=root should be set in lsf.sudoers:

```
[root@ddnmon61 ~]$ echo "LSB_PRE_POST_EXEC_USER=\
root" >> /etc/lsf.sudoers
[root@ddnmon61 ~]$ chmod 600 /etc/lsf.sudoers
```

3. The changes are applied by restarting the hosts:

```
[root@ddnmon61 ~]$ badmin hrestart all
```

Further configuration: For further configuration the *Administering Platform LSF* manual provided with the LSF software should be consulted.

Running LSF

Role assignment and role removal enables and disables LSF from cmgui (sections 9.4.1) or cmsh (section 9.4.2).

An active LSF master (typically, but not necessarily on a head node) has the following LSF-related processes or services running on it:

Process/Service	Description
res	Remote Execution Server*
sbatchd	client batch job execution daemon*
mbatchd	master batch job execution daemon
eauth	External Authentication method
lim	Load Information Manager*
pim	Process Information Manager*
pem	Process Execution Manager*
vemkd	Platform LSF Kernel Daemon
egosc	Enterprise Grid Orchestrator service controller
mbschd	master batch scheduler daemon

*These services/processes run on compute nodes.

Non-active LSF-masters running as compute nodes run the processes marked with an asterisk only.

Logs for LSF processes and services are kept under /cm/shared/apps/lsf/log/ (or $LSF_TOP/log/ if the value of $LSF_TOP during installation is other than the recommended /cm/shared/apps/lsf/).

9.6 Using cmgui With Workload Management

Viewing the workload manager services from cmgui is described in section 9.4.3.

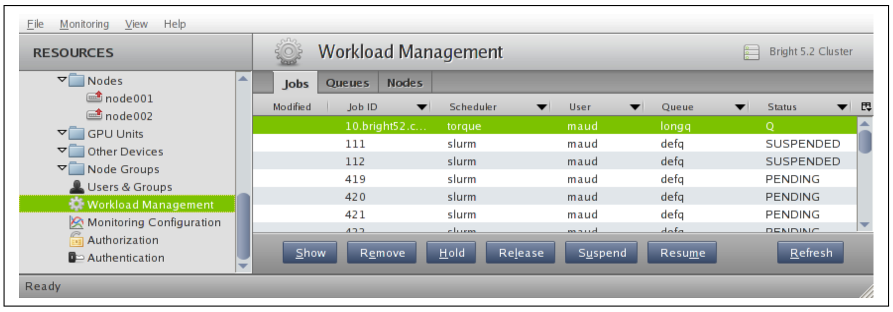
Selecting the DirectMon workload manager item from the resources tree displays tabs that let a cluster administrator change the states of:

- jobs
- queues
- nodes

These tabs are described next.

9.6.1 Jobs Display And Handling In `cmgui`

Selecting the `Jobs` tab displays a list of job IDs, along with the scheduler, user, queue, and status of the job (figure 9.9).

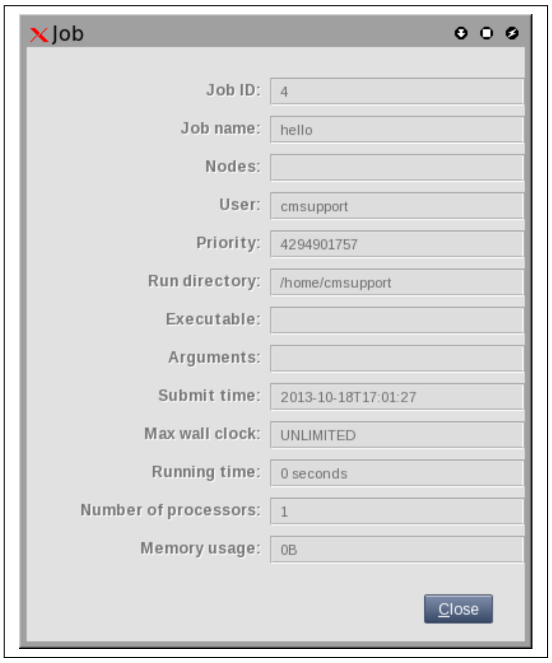


Modified	Job ID	Scheduler	User	Queue	Status
	10.bright52.c...	torque	maud	longq	Q
	111	slurm	maud	defq	SUSPENDED
	112	slurm	maud	defq	SUSPENDED
	419	slurm	maud	defq	PENDING
	420	slurm	maud	defq	PENDING
	421	slurm	maud	defq	PENDING

Figure 9.9: Workload Manager Jobs

Within the tabbed pane:

- The `Show` button allows further details of a selected job to be listed. For example, the details of a Slurm job are displayed as in figure 9.10:



Job ID:	4
Job name:	hello
Nodes:	
User:	cmsupport
Priority:	4294901757
Run directory:	/home/cmsupport
Executable:	
Arguments:	
Submit time:	2013-10-18T17:01:27
Max wall clock:	UNLIMITED
Running time:	0 seconds
Number of processors:	1
Memory usage:	0B

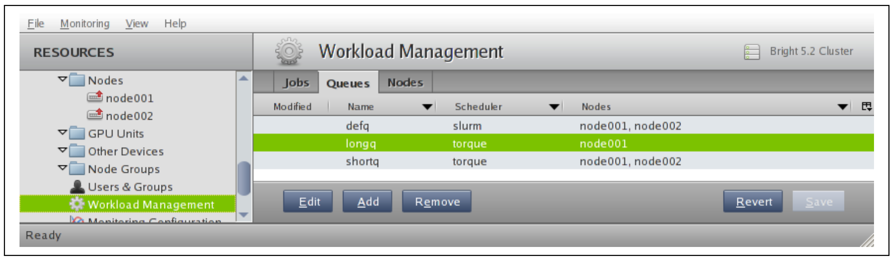
Figure 9.10: Workload Manager Selected Job Details

- The `Remove` button removes selected jobs from the queue.

- The `Hold` button stops selected queued jobs from being considered for running by putting them in a `Hold` state.
- The `Release` button releases selected queued jobs in the `Hold` state so that they are considered for running again.
- The `Suspend` button suspends selected running jobs.
- The `Resume` button allows selected suspended jobs to run again.
- The `Refresh` button refreshes the screen so that the latest available jobs list is displayed.

9.6.2 Queues Display And Handling In `cmgui`

Selecting the `Queues` tab displays a list of queues available, their associated scheduler, and the list of nodes that use each queue (figure 9.11).

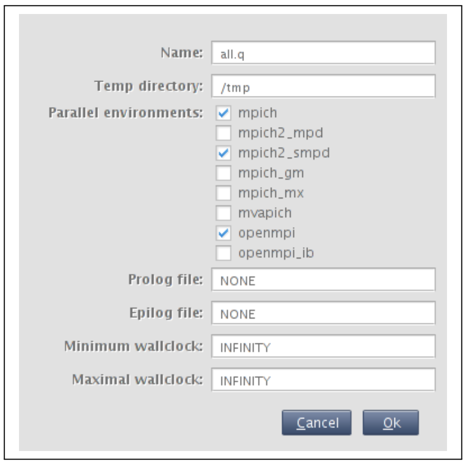


Modified	Name	Scheduler	Nodes
	defq	slurm	node001, node002
	longq	torque	node001
	shortq	torque	node001, node002

Figure 9.11: Workload Manager Queues

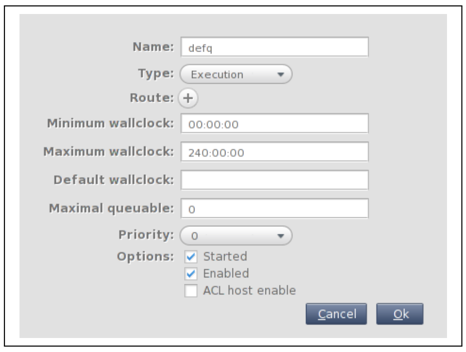
Within the tabbed pane:

- The `Edit` button allows an existing job queue of a workload manager to be edited. The particular values that can be edited for the queue depend upon the workload manager used (figures 9.12, 9.13 and 9.14).



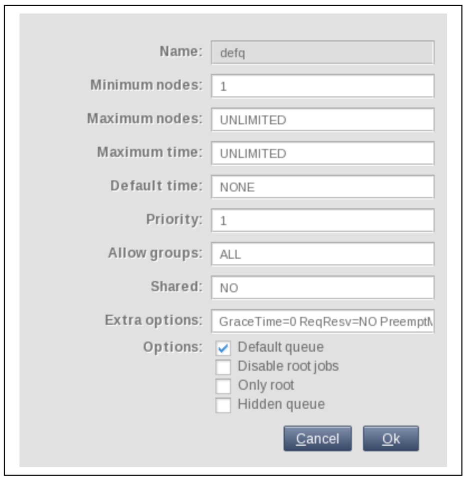
Name:	all.q
Temp directory:	/tmp
Parallel environments:	✓ mpich
	mpich2_mpd
	✓ mpich2_smpd
	mpich_gm
	mpich_mx
	mvapich
	✓ openmpi
	openmpi_ib
Prolog file:	NONE
Epilog file:	NONE
Minimum wallclock:	INFINITY
Maximal wallclock:	INFINITY

Figure 9.12: Workload Management Queues Edit Dialog For SGE



Name: defq
Type: Execution
Route: +
Minimum wallclock: 00:00:00
Maximum wallclock: 240:00:00
Default wallclock:
Maximal queuable: 0
Priority: 0
Options: ✓ Started
✓ Enabled
☐ ACL host enable

Figure 9.13: Workload Management Queues Edit Dialog For Torque And PBS Pro



Name: defq
Minimum nodes: 1
Maximum nodes: UNLIMITED
Maximum time: UNLIMITED
Default time: NONE
Priority: 1
Allow groups: ALL
Shared: NO
Extra options: GraceTime=0 ReqResv=NO PreemptM
Options: ✓ Default queue
☐ Disable root jobs
☐ Only root
☐ Hidden queue

Figure 9.14: Workload Management Queues Edit Dialog For Slurm

In the edit dialog:

– the generic names "`Minimum wallclock`" and "`Maximum wallclock`" correspond respectively to the soft and hard walltimes allowed for the jobs in the queue. Specifically, these are `s_rt` and `h_rt` in SGE, or `resources_default.walltime`, and `resources_max.walltime` in Torque and PBS Pro. The `Maximum time` parameter for Slurm corresponds to `Maximum wallclock` and sets Slurm's `MaxTime` value in `/etc/slurm/slurm.conf`.

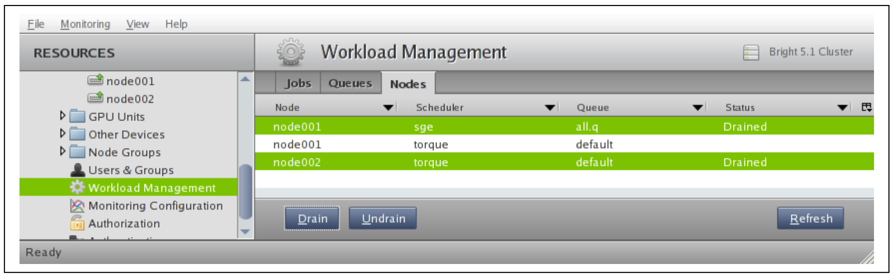
– The `Prolog` and `Epilog` files that can be specified in the dialog for SGE are CMDaemon-managed scripts run before and after the job is executed. A default global `Prolog` script is used by DirectMon if no CMDaemon-managed script exists. The global script ensures that a health check script flagged as a prejob script (section 11.4.3), is run before a job runs. Administrators creating their own `Prolog` script can refer to the global `Prolog` script (`/cm/local/apps/cmd/scripts/prolog`), to see how it calls the `prolog-healthchecker` script, which, in turn, executes a flagged health check script in its call to cmprejobcheck.

The Prolog and Epilog scripts for Torque and PBS Pro are set up for the software images, and their paths cannot be altered via DirectMon.

- The Add button allows a new job queue to be added to a workload manager.
- The Remove button removes a job queue from the workload manager.
- The Revert button reverts the Queues tabbed pane to its last saved state.
- The Save button saves the modified Queues tabbed pane.

9.6.3 Nodes Display And Handling In cmgui

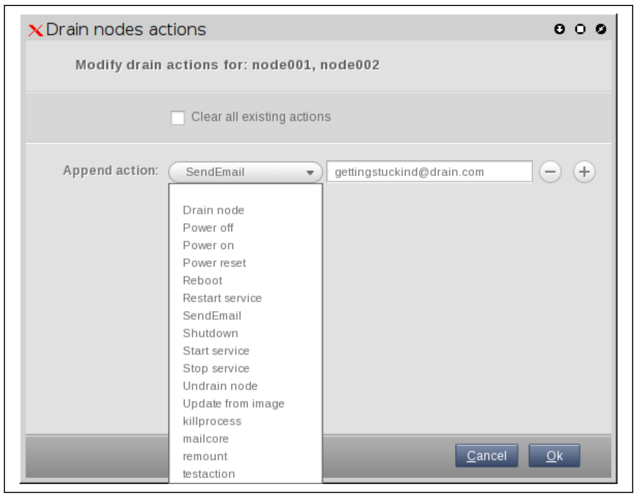
Selecting the Nodes tab displays a list of nodes, along with their schedulers, queues, and whether they are in a status of Drained or Undrained (figure 9.15).



Node	Scheduler	Queue	Status
node001	sge	all.q	Drained
node001	torque	default	
node002	torque	default	Drained

Figure 9.15: Node Drainage

- The Drain button acts on the selected node, scheduler, and queue combination. It opens up a dialog (figure 9.16).



Modify drain actions for: node001, node002

Append action: SendEmail gettingstuckind@drain.com

Figure 9.16: Node Drainage

  – If the default options are accepted, then clicking on the Ok button sets the node to a Drained state. This means that jobs no longer run on that node, for that workload manager.

  - If actions are appended to the drain command, from the set of actions described in Appendix H.3.1, then these are executed in order after the default drain action takes place for that node, for that workload manager.
  - If there are any existing actions, then ticking the `Clear all existing actions` checkbox means that they are cleared after the `OK` button is clicked

- The `Undrain` button unsets a `Drained` state, allowing jobs to start running for that combination.
- The `Refresh` button refreshes the screen so that the latest available state is displayed.

The `cmsh` commands for this functionality are described in section 9.7.3.

9.7 Using `cmsh` With Workload Management

9.7.1 Jobs Display And Handling In `cmsh`: `jobs` Mode

`jobs` Mode In `cmsh`: Top Level

At the top level of `jobs` mode, the administrator can view all jobs regardless of scheduler type with the `list` command:

Example

```
[ddnmon61->jobs]% list
Type         Job ID       User         Queue        Status
------------ ------------ ------------ ------------ ------------
SGEJob       620          maud         all.q        r
SGEJob       621          maud                      qw
TorqueJob    90.ddnmon61+ maud         hydroq       R
```

Also within the `jobs` mode, the `hold`, `release`, `suspend`, `resume`, `show`, and `remove` commands act on jobs when used with a specified scheduler type and job ID. Continuing with the example:

```
[ddnmon61->jobs]% suspend torque 90.ddnmon61.cm.cluster
Success
[ddnmon61->jobs]% list
Type         jobid        User         Queue        Status
------------ ------------ ------------ ------------ ------------
SGEJob       620          maud         all.q        r
SGEJob       621          maud                      qw
TorqueJob    90.ddnmon61+ maud         hydroq       S
```

While at the jobs mode top level, the suspended job here can be made to resume using `suspend`'s complementary command—`resume`. However, `resume` along with the other commands can also be executed within a `scheduler` submode, as is shown shortly.

`jobs` Mode In `cmsh`: The `scheduler` Submode

Setting the scheduler type sets the `scheduler` submode, and can be done thus (continuing with the preceding example):

```
[ddnmon61->jobs]% scheduler torque
[ddnmon61->jobs(torque)]%
```

The submode restriction can be unset with: `scheduler ""`.

The top level job mode commands executed within the `scheduler` submode then only apply to jobs running under that scheduler. The `list` and `resume` commands, for example, then only apply only to jobs running under torque (continuing with the example):

```
[ddnmon61->jobs(torque)]% list; !#no sge jobs listed now - only torque
Type          Job ID        User          Queue         Status
------------- ------------- ------------- ------------- -------------
TorqueJob     90.ddnmon61+ maud          hydroq        S
[ddnmon61->jobs(torque)]% resume 90.ddnmon61.cm.cluster; !#torque job
Success
[ddnmon61->jobs(torque)]% list; !#only torque jobs
Type          Job ID        User          Queue         Status
------------- ------------- ------------- ------------- -------------
TorqueJob     90.ddnmon61+ maud          hydroq        R
```

`jobs` Mode in `cmsh`: The `show` Command

The `show` command for a particular scheduler and job lists details of the job. Continuing with the preceding example:

```
[ddnmon61->jobs(torque)]% show 90.ddnmon61.cm.cluster;
Parameter                  Value
-------------------------- ---------------------------------------------
Arguments                  -q hydroq /home/maud/sleeper.sh
Executable
In queue
Job ID                     90.ddnmon61.cm.cluster
Job name                   sleeper.sh
Mail list
Mail options               a
Maximum wallclock time     02:00:00
Memory usage               0
Nodes                      node001
Notify by mail             yes
Number of processes        1
Priority                   0
Queue                      hydroq
Run directory              /home/maud
Running time               809
Status                     R
Stderr file                ddnmon61.cm.cluster:/home/maud/sleeper.sh.e90
Stdout file                ddnmon61.cm.cluster:/home/maud/sleeper.sh.o90
Submission time            Fri Feb 18 12:49:01 2011
Type                       TorqueJob
User                       maud
```

9.7.2 Job Queues Display And Handling In `cmsh`: `jobqueue` Mode

Properties of scheduler job queues can be viewed and set in `jobqueue` mode.

`jobqueue` Mode In `cmsh`: Top Level

If a `scheduler` submode is not set, then the `list`, `qstat`, and `listpes` commands operate, as is expected, on all queues for all schedulers.

At the top level of `jobqueue` mode:

- `list` lists the queues associated with a scheduler.

  Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% jobqueue
[ddnmon61->jobqueue]% list
Type         Name
------------ ------------------------
sge          all.q
torque       default
torque       hydroq
torque       longq
torque       shortq
```

- `qstat` lists statistics for the queues associated with a scheduler.

  Example

```
[ddnmon61->jobqueue]% qstat
======================= sge ========================
Queue        Load      Total     Used      Available
------------ --------- --------- --------- ---------
all.q        0.1       1         0         1
====================== torque ======================
Queue        Running   Queued    Held      Waiting
------------ --------- --------- --------- ---------
default      0         0         0         0
hydroq       1         0         0         0
longq        0         0         0         0
shortq       0         0         0         0
====================== pbspro ======================
Queue        Running   Queued    Held      Waiting
------------ --------- --------- --------- ---------
```

- `listpes` lists the parallel environment available for schedulers

  Example

  (some details elided)

```
[ddnmon61->jobqueue]% listpes
Scheduler    Parallel Environment
------------ ------------------------
sge          make
sge          mpich
...
sge          openmpi_ib
```

- `scheduler` sets the `scheduler` submode

  Example

```
[ddnmon61->jobqueue]% scheduler torque
Working scheduler is torque
[ddnmon61->jobqueue(torque)]%
```

  The submode can be unset using: `scheduler ""`

jobqueue Mode In cmsh: The scheduler Submode

If a scheduler submode is set, then commands under jobqueue mode operate only on the queues for that particular scheduler. For example, within the torque submode of jobqueue mode, the list command shows only the queues for torque.

Example

```
[ddnmon61->jobqueue]% list
Type         Name
------------ ------------------------
sge          all.q
torque       default
torque       longq
torque       shortq
[ddnmon61->jobqueue]% scheduler torque
Working scheduler is torque
[ddnmon61->jobqueue(torque)]% list
Type         Name
------------ ------------------------
torque       default
torque       longq
torque       shortq
```

jobqueue Mode In cmsh: Other Object Manipulation Commands

The usual object manipulation commands of section 3.5.3 work at the top level mode as well as in the scheduler submode:

Example

```
[ddnmon61->jobqueue]% list torque
Type         Name
------------ ------------------------
torque       default
torque       longq
torque       shortq
[ddnmon61->jobqueue]% show torque longq
Parameter                      Value
------------------------------ ------------------------------------------
Maximal runtime                23:59:59
Minimal runtime                00:00:00
Queue type                     Execution
Routes
Type                           torque
name                           longq
nodes                          node001.cm.cluster node002.cm.cluster
[ddnmon61->jobqueue]% get torque longq maximalruntime
23:59:59
[ddnmon61->jobqueue]%
[ddnmon61->jobqueue]% scheduler torque
Working scheduler is torque
[ddnmon61->jobqueue(torque)]% list
Type         Name
------------ ------------------------
torque       default
torque       longq
torque       shortq
[ddnmon61->jobqueue(torque)]% show longq
Parameter                      Value
------------------------------ ------------------------------------------
Maximal runtime                23:59:59
Minimal runtime                00:00:00
Queue type                     Execution
Routes
Type                           torque
name                           longq
nodes                          node001.cm.cluster node002.cm.cluster
[ddnmon61->jobqueue(torque)]% get longq maximalruntime
23:59:59
[ddnmon61->jobqueue(torque)]% use longq
[ddnmon61->jobqueue(torque)->longq]% show
Parameter                      Value
------------------------------ ------------------------------------------
Maximal runtime                23:59:59
Minimal runtime                00:00:00
Queue type                     Execution
Routes
Type                           torque
name                           longq
nodes                          node001.cm.cluster node002.cm.cluster
[ddnmon61->jobqueue(torque)->longq]% get maximalruntime
23:59:59
```

9.7.3 Nodes Drainage Status And Handling In `cmsh`

The node drainage displayed using `cmgui` (section 9.6.3) has `cmsh` equivalents that are described in this section.

Running the `device` mode command `drainstatus` displays if a specified node is in a `Drained` state or not. In a `Drained` state jobs are not allowed to start running on that node.

Running the `device` mode command `drain` puts a specified node in a `Drained` state:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device
[ddnmon61->device]% drainstatus
Node                     Queue                    Status
------------------------ ------------------------ ----------------
node001                  workq
node002                  workq
[ddnmon61->device]% drain node001
Node                     Queue                    Status
------------------------ ------------------------ ----------------
node001                  workq                    Drained
```

The `undrain` command unsets the `Drained` state so that jobs may start running on the node again.

The `drain`, `undrain`, and `drainstatus` commands have the same options. The options can make the command apply to not just one node, but to a list of nodes, a group of nodes, a category of nodes, or to a chassis. Continuing the example:

```
[ddnmon61->device]% drain -c default; !# for a category of nodes
Node                     Queue                    Status
------------------------ ------------------------ ----------------
node001                  workq                    Drained
node002                  workq                    Drained
```

The help text for each command indicates the syntax:

Example

```
[root@ddnmon61 ~]# cmsh -c "device help drain"
Name: drain - Drain a set of nodes

Usage: drain [OPTIONS/node]

Options: -n, --nodes node(list)
              List of nodes, e.g. node001..node015,node20..node028,node\
030 or ^/some/file/containing/hostnames

         -g, --group group(list)
              Include all nodes that belong to the node group, e.g. tes\
tnodes or test01,test03

         -c, --category category(list)
              Include all nodes that belong to the category, e.g. defau\
lt or default,gpu

         -r, --rack rack(list)
              Include all nodes that are located in the given rack, e.g\
rack01 or rack01..rack04

         -h, --chassis chassis(list)
              Include all nodes that are located in the given chassis, \
e.g chassis01 or chassis03..chassis05

         -s,--setactions <actions>
             set drain actions, actions already set will be removed (co\
mma separated)

         -a,--appendactions <actions>
             append drain actions to already existing drain actions (co\
mma separated)

         -e,--removeactions <actions>
             remove drain actions

         -l, --clearactions
              remove all drain actions

         -i, --listactions
              list all drain actions
```

9.7.4 Launching Jobs With cm-launcher

Some MPI distributions occasionally leave processes behind that cannot be killed from the workload manager. To prevent this situation from occurring, DirectMon provides the `cm-launcher` wrapper for the `mpirun` command, which tracks and kills such processes. The tracking relies on knowing what processes the workload manager launches, so it can only run from within a suitable workload manager. Currently, suitable workload managers are Torque or SGE.

In the following job script examples, instead of having users use:

Example

```
mpirun <...>
```

which may not have a job clean up properly after it ends, users can use:

Example

```
cm-launcher mpirun <...>
```

which cleans up properly after the job ends. Here, `<...>` is used to indicate the mix of options, programs and arguments used with `mpirun`.

For Torque `cm-launcher` can be used if the default Torque `epilogue` script provided by the DirectMon Torque package is present, at `/cm/local/apps/torque/var/spool/mom_priv/epilogue`.

For SGE the procedure is as follows:

- A symlink to `cm-launcher` is created to the `<arch>` SGE functions directory library

  ```
  ln -s /cm/shared/apps/sge/var/cm/cm-launcher /cm/shared/apps/sge\
  /current/bin/<arch>
  ```

- SGE's `epilog` (spelled without the "-ue" ending) script is set either for a particular queue, or globally.

  – To set it for a particular queue, for example, for the default queue `all.q`, the following `cmsh` commands can be run:

    Example

    ```
    [root@ddnmon61 ~]# cmsh
    [ddnmon61]% jobqueue
    [ddnmon61->jobqueue]% scheduler sge
    Working scheduler is sge
    [ddnmon61->jobqueue(sge)]% set all.q epilog /cm/shared/apps/sge\
    /var/cm/epilog
    [ddnmon61->jobqueue*(sge)]% commit
    Successfully committed 1 JobQueues
    [ddnmon61->jobqueue(sge)]%
    ```

  – To set it globally for all queues, that is, not just the queue `all.q` but all the other queues as well, the following SGE configuration command is used:

    ```
    qconf -mconf global
    ```

    This starts up an editor acting on the `global` configuration setting. The epilog line entry is modified to:

    ```
    epilog                /cm/shared/apps/sge/var/cm/epilog
    ```

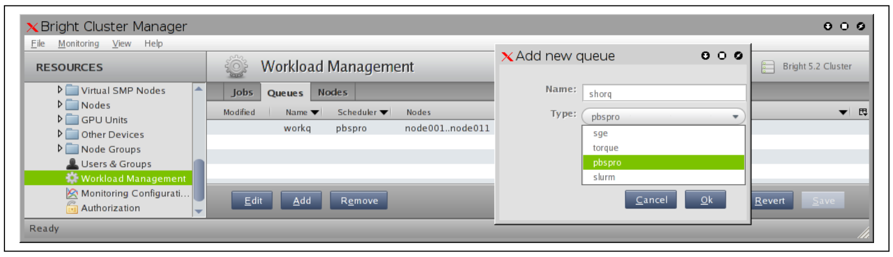
9.8 Examples Of Workload Management Assignment

9.8.1 Setting Up A New Category And A New Queue For It

Suppose a new node with processor optimized to handle Shor's algorithm is added to a cluster that originally has no such nodes. This merits a new category, `shornodes`, so that administrators can configure more such new nodes efficiently. It also merits a new queue, `shorq`, so that users are aware that they can submit suitably optimized jobs to this category.

A New Category And A New Queue With `cmgui`

To create a new queue, the Workload Management item is selected, and the `Queues` tab selected. The `Add` button is used to associate a newly created queue with a scheduler and add it to the workload manager. The modification is then saved (figure 9.17).

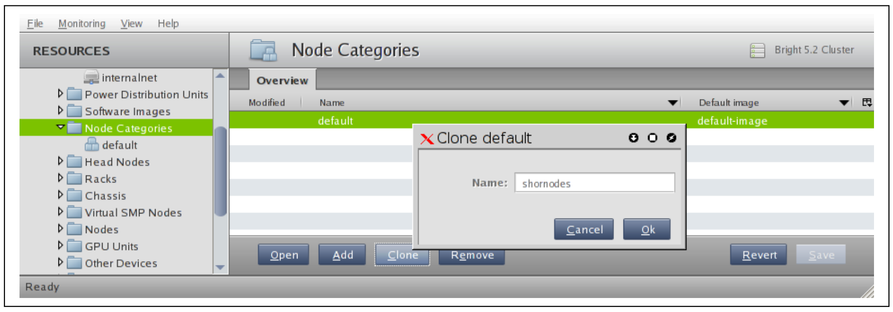


Modified	Name	Scheduler	Nodes
	workq	pbspro	node001..node011

Name: shorq
Type: pbspro
sge
torque
pbspro
slurm

Figure 9.17: Adding A New Queue Via `cmgui`

A useful way to create a new category is to simply clone the old `default` category over to a new category, and then change parameters in the new category to suit the new machine (figure 9.18).



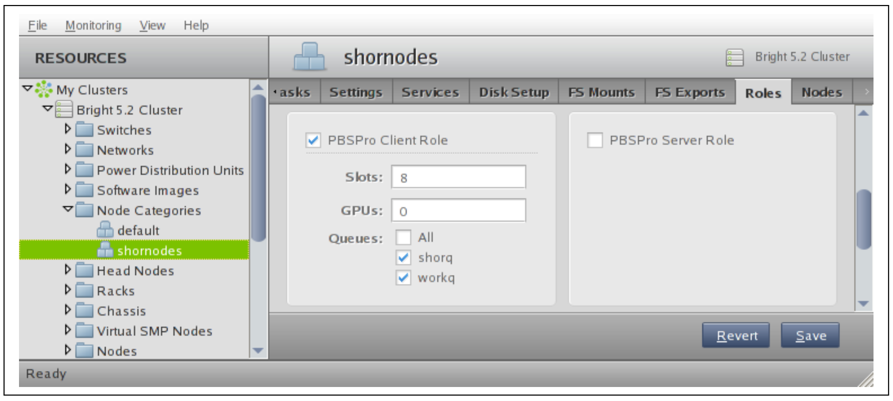
Modified	Name	Default image
	default	default-image

Name: shornodes

Figure 9.18: Cloning A New Category Via `cmgui`

Having cloned and saved the category, called `shornodes` in the example of figure 9.18, the configuration of the category may be altered to suit the new machine, perhaps by going into the settings tab and altering items there.

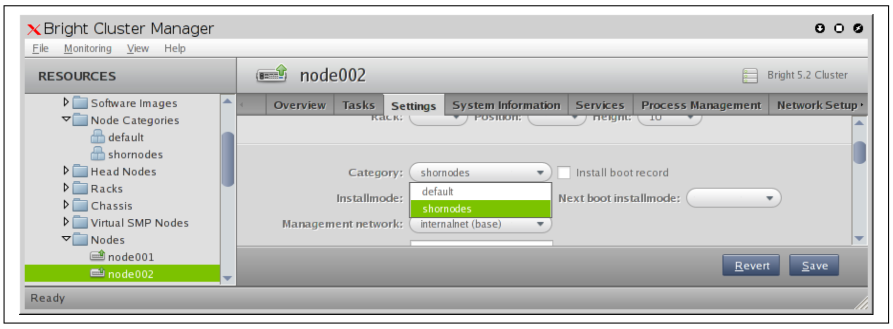
Next, the queue is set for this new category, `shornodes`, by going into the `Roles` tabbed pane of that category, selecting the appropriate workload manager client role and queues, and saving the setting (figure 9.19).



PBSPro Client Role: ✓
Slots: 8
GPUs: 0
Queues: All ☐, shorq ✓, workq ✓
PBSPro Server Role: ☐

Figure 9.19: Setting A Queue For A New Category Via `cmgui`

Finally, a node in the `Nodes` folder that is to be placed in the new `shornodes` category must be placed there by changing the category value of that node in its `Settings` tab (figure 9.20).



Category: shornodes (options: default, shornodes)
Install boot record: ☐
Installmode:
Next boot installmode:
Management network: internalnet (base)

Figure 9.20: Setting A New Category To A Node Via `cmgui`

A New Category And A New Queue With `cmsh`

The preceding example can also be configured in `cmsh` as follows:

The new queue can be added from within `jobqueue` mode, for the workload manager. For example, if Slurm is enabled:

```
[ddnmon61]% jobqueue add shorq
[ddnmon61->jobqueue*(slurm)->shorq*]% commit
```

The new category, `shornodes`, can be created by cloning an old category, such as `default`:

```
[ddnmon61->jobqueue(slurm)->shorq]% category
[ddnmon61->category]% clone default shornodes
[ddnmon61->category*[shornodes*]]% commit
```

Then, going into the roles tab, appropriate queues and workload manager roles can be set and committed for that category:

```
[ddnmon61->category[shornodes]]% roles
[ddnmon61->category[shornodes]->roles]% use slurmclient
[ddnmon61->category[shornodes]->roles[slurmclient]]% append queues shorq
[ddnmon61->category[shornodes*]->roles*]% commit
```

The nodes belonging to the shornodes category can then be placed by going into `device` mode to select the nodes to be placed in that category. For example:

```
[ddnmon61->category[shornodes]->roles]% device use node002
[ddnmon61->device[node002]]% set category shornodes
[ddnmon61->device*[node002*]]% commit
```

9.8.2 Setting Up A Prejob Health Check

How It Works

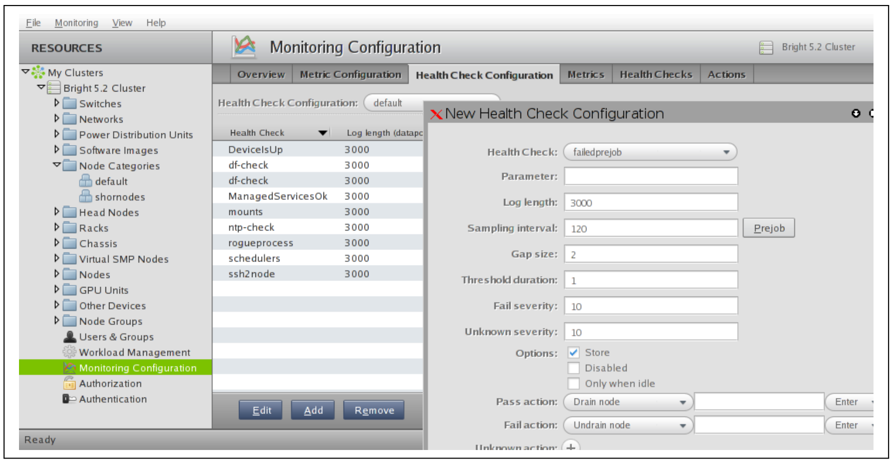
Health checks (section 11.2.4) by default run as scheduled tasks over regular intervals. They can optionally be configured to run as prejob health checks, that is, before a job is run. If the response to a prejob health check is PASS, then it shows that the node is displaying healthy behavior for that particular health aspect.

If the response to a prejob health check is FAIL, then it implies that the node is unhealthy, at least for that aspect. A consequence of this may be that a job submitted to the node may fail, or may not even be able to start. To disallow passing a job to such unhealthy nodes is therefore a good policy, and so for a cluster in the default configuration, the action (section 11.2.2) taken defaults to putting the node in a Drained state (sections 9.6.3 and 9.7.3), with DirectMon arranging a rescheduling of the job.

A node that has been put in a Drained state with a health check is not automatically undrained. The administrator must clear such a state manually.

Configuration Using cmgui

To configure the monitoring of nodes as a prejob health check in cmgui, the Monitoring Configuration resource item is selected, and the Health Check Configuration tabbed pane is opened. The default resource is chosen as a value for Health Check Configuration, and the Add button is clicked on to add the health check via a dialog (figure 9.21). In the dialog, the Health Check script value is set to the chosen health check. If the health check is already listed, then it can be edited as needed. The Sampling interval is set to prejob, which automatically sets the Fail action to the Drain node action, when the configuration is saved. After saving these settings, any node that is not in the Drained state in the default resource gets a pre-job check when a job is scheduled for the node, and the pre-job check puts the node in a Drained state if it is unhealthy.



Health Check Configuration: default

Health Check	Log length (datapo
DevicesUp	3000
df-check	3000
df-check	3000
ManagedServicesOk	3000
mounts	3000
ntp-check	3000
rogueprocess	3000
schedulers	3000
ssh2node	3000

New Health Check Configuration

Health Check:	failedprejob
Parameter:	
Log length:	3000
Sampling interval:	120
Gap size:	2
Threshold duration:	1
Fail severity:	10
Unknown severity:	10
Options:	✓ Store; ☐ Disabled; ☐ Only when idle
Pass action:	Drain node; Enter
Fail action:	Undrain node; Enter
Unknown action:	

Figure 9.21: Configuring A Prejob Healthcheck Via `cmgui`

Configuration Using `cmsh`

To configure a prejob health check with `cmsh`, the `healthconf` submode (section 11.7.4) is entered, and the prejob health script object used. In the following example, where some text has been elided, the object is the `smart` script:

Example

```
[ddnmon61% monitoring setup healthconf default
[ddnmon61->monitoring->setup[default]->healthconf]% use smart
[ddnmon61->...->healthconf[smart]]% set checkinterval prejob
set checkinterval prejob
```

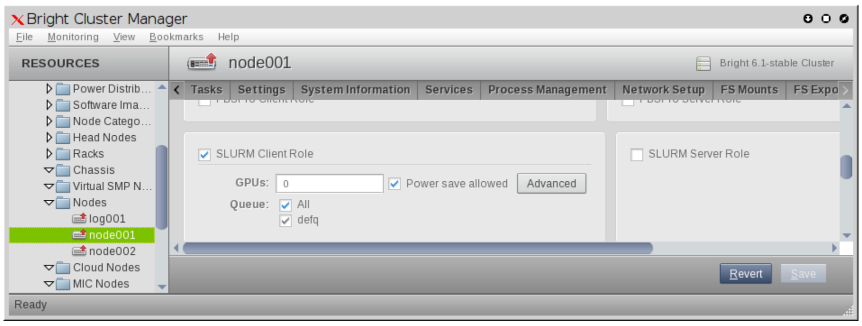
The `failactions` value automatically switches to "`enter: Drain node()`" when the value for the `checkinterval` parameter of the health check is set to `prejob`.

9.9 Power Saving Features

9.9.1 Slurm

Slurm provides a power saving mechanism that allows non-head nodes in a client role to go into a power save mode. The power saving is disallowed by default. It can be allowed at the category or node level as follows:

- in `cmgui` within the `Nodes` resource or the `Node Categories` resource: Within the resource, the `Roles` tabbed pane is selected, the `Power save allowed` checkbox can be ticked, and the setting saved. For example, figure 9.22 illustrates this for `node001` specified within the `Nodes` resource.



SLURM Client Role ✓
GPUs: 0 Power save allowed ✓ Advanced
Queue: All ✓ defq ✓
SLURM Server Role

Figure 9.22: Allowing Power Save In Slurm With `cmgui`

- in `cmsh` within the `category` mode, or in the `device` mode: Within the mode for the specific category or device, the `roles` submode must be chosen. The `slurmclient` role is used, the value for `powersavingallowed` set to `yes`, and the setting is committed.

  Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device roles node001
[ddnmon61->device[node001]->roles]% set slurmclient powersaving\
allowed yes; commit
```

Additional power saving configuration file parameters are described in the Slurm power saving guide at `https://computing.llnl.gov/linux/slurm/power_save.html`

10 Post-Installation Software Management

Some time after DirectMon has been installed, administrators may wish to manage other software on the cluster. This means carrying out software management actions such as installation, removal, updating, version checking, and so on.

Since DirectMon is built on top of an existing Linux distribution, it is best that the administrator use distribution-specific package utilities for software management.

Packages managed by the distribution are hosted by distribution repositories. SUSE and Red Hat enterprise distributions require the purchase of their license in order to access their repositories.

Packages managed by the DirectMon are hosted by the DDN repository. Access to the DDN repositories also requires a license (section 4.1).

There may also be software that the administrator would like to install that is outside the default packages collection. These could be source files that need compilation, or packages in other repositories.

A software image (section 3.1.2) is the file system that a node picks up from a provisioner (a head node or a provisioning node) during provisioning. A subtopic of software management on a cluster is software image management—the management of software on a software image. By default, a node uses the same distribution as the head node for its base image along with necessary minimal, cluster-mandated changes. A node may however deviate from the default, and be customized by having software added to it in several ways.

This chapter covers the techniques of software management for the cluster.

Section 10.1 describes the naming convention for a DirectMon RPM package.

Section 10.2 describes how an RPM package is managed for the head node.

Section 10.3 describes how a kernel RPM package can be managed on a head node or image.

Section 10.4 describes how an RPM package can be managed on the software image.

Section 10.5 describes how a software other than an RPM package can be managed on a software image.

Section 10.6 describes how custom software images are created that are completely independent of the existing software image distribution and version.

10.1 DirectMon RPM Packages And Their Naming Convention

Like the distributions it runs on top of, DirectMon uses RPM (RPM Package Manager) packages. An example of a DirectMon RPM package is:

```
mpich-ge-gcc-64-1.2.7-116_cm.x86_64.rpm
```

The file name has the following structure:

*package-version-revision*_cm*x*.*y*.*architecture*.rpm where:

- *package* (mpich-ge-gcc-64) is the name of the package
- *version* (1.2.7) is the version number of the package
- *revision* (116) is the revision number of the package
- *x.y* () is the version of DirectMon for which the RPM was built
- *architecture* (x86_64) is the architecture for which the RPM was built

To check whether DDN or the distribution has provided a file that is already installed on the system, the package it has come from can be identified using "rpm -qf".

Example

```
[root@ddnmon61 bin]# rpm -qf /usr/bin/zless
gzip-1.3.12-18.el6.x86_64
[root@ddnmon61 bin]# rpm -qf /cm/local/apps/cmd/sbin/cmd
cmdaemon--13383_cm.x86_64
```

In the example, /usr/bin/zless is supplied by the distribution, while /cm/local/apps/cmd/sbin/cmd is supplied by DDN, as indicated by the "_cm" in the nomenclature.

As an aside, it should be noted that using a distribution package means less work for DDN developers. System administrators should therefore be aware that the DDN version of a package is provided and used for a reason. Replacing the DDN version with a distribution version can result in subtle and hard-to-trace problems in the cluster, and DDN cannot offer support for a cluster that is in such a state.

More information about the RPM Package Manager is available at http://www.rpm.org.

10.2 Managing Packages On The Head Node

10.2.1 Managing RPM Packages On The Head Node

Once DirectMon has been installed, distribution packages and DirectMon software packages can be managed using the rpm command-line utility.

A more convenient way of managing RPM packages is to use either YUM or zypper. Both of these tools are repository and package managers. The zypper tool is recommended for use by the SUSE 11 distribution, while YUM is recommended for use by the other distributions that DirectMon supports. YUM is not installed by default in SUSE 11, and it is better not to install and use it with SUSE 11 unless the administrator is familiar with configuring YUM.

For YUM and zypper, the following commands list all available packages:

```
yum list
```
or
```
zypper packages
```

For zypper, the short command option `pa` can also be used instead of `packages`.

The following commands can install a new package called <*package name*>:

```
yum install <package name>
```
or
```
zypper in <package name>
```

All installed packages can be updated with:

```
yum update
```
or
```
zypper refresh; zypper up    #refresh mostly not needed
```

DDN maintains YUM and zypper repositories at:

`http://updates.ddn.com/yum`

and updates are fetched by YUM and zypper for DirectMon packages from there by default, to overwrite older package versions by default.

Accessing the YUM repositories manually (i.e. not through YUM or zypper) requires a username and password. Authentication credentials are provided upon request. For more information on this, `support@ddn.com` should be contacted.

The repository managers use caches to speed up their operations. Occasionally these caches may need flushing to clean up the index files associated with the repository. This is done with:

```
yum clean all
```
or
```
zypper clean -a
```

As an extra protection to prevent DirectMon installations from receiving malicious updates, all DirectMon packages are signed with the DDN GPG public key (`0x5D849C16`), installed by default in `/etc/pki/rpm-gpg/RPM-GPG-KEY-cm` for Red Hat, Scientific Linux, and CentOS packages. The DDN public key is also listed in Appendix B.

The first time YUM or zypper are used to install updates, the user is asked whether the DDN public key should be imported into the local RPM database. Before answering with a "Y", yum users may choose to compare the contents of /etc/pki/rpm-gpg/RPM-GPG-KEY-cm with the key listed in Appendix B to verify its integrity. Alternatively, the key may be imported into the local RPM database directly, using the following command:

```
rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-cm
```

Installation of the following third party packages, most of which are repackaged by DDN for installation purposes, is described in Chapter 13. These are installed mostly on the head node, but some packages work as a server and client process, with the clients installed and running on the regular nodes:

- Modules (section 13.1)
- Shorewall (section 13.2)
- Compilers (section 13.3):
  - GCC (section 13.3.1)
  - Intel Compiler Suite (section 13.3.2)
  - PGI High-Performance Compilers (section 13.3.3)
  - AMD Open64 Compiler Suite (section 13.3.4)
  - FLEXlm License Daemon (section 13.3.5)
- Intel Cluster Checker (section 13.4)
- CUDA (section 13.5)
- Lustre (section 13.7)
- ScaleMP (section 13.8)

Exclusion of packages on the head node can be carried out as explained in section 10.3.2, where the kernel package is used as an example for exclusion.

10.2.2 Managing Non-RPM Software On The Head Node

Sometimes a package is not packaged as an RPM package by DDN or by the distribution. In that case, the software can be usually be treated as for installation onto a standard distribution. There may be special considerations on placement of components that the administrator may feel appropriate due to the particulars of a cluster configuration.

For example, for compilation and installation of the software, some consideration may be made of the options available on where to install parts of the software within the default shared filesystem. A software may have a compile option, say --prefix, that places an application <*application*> in a directory specified by the administrator. If the administrator decides that <*application*> should be placed in the shared directory, so that everyone can access it, the option could then be specified as: "--prefix=/cm/shared/apps/<*application*>".

Other commonly provided components of software for the applications that are placed in shared may be documentation, licenses, and examples. These may similarly be placed in the directories /cm/shared/docs, /cm/shared/licenses, and /cm/shared/examples. The placement may be done with a compiler option, or, if that is not done or not possible, it could be done by modifying the placement by hand later. It is not obligatory to do the change of placement, but it helps with cluster administration to stay consistent as packages are added.

Module files (sections 3.2 and 13.1) may sometimes be provided by the software, or created by the administrator to make the application work for users easily with the right components. The directory /cm/shared/modulefiles is recommended for module files to do with such software.

To summarize the above considerations on where to place software components, the directories under /cm/shared that can be used for these components are:

```
/cm/shared/
|-- apps
|-- docs
|-- examples
|-- licenses
`-- modulefiles
```

10.3 Kernel Management On A Head Node Or Image

Care should be taken when updating a head node or software image. This is particularly true when custom kernel modules compiled against a particular kernel version are being used.

10.3.1 Installing A Standard Distribution Kernel

A standard distribution kernel is treated almost like any other package in a distribution. This means that for head nodes, installing a standard kernel is done according to the normal procedures of managing a package on a head node (section 10.2), while for regular nodes, installing a standard distribution kernel onto a regular node is done according to the normal procedures of managing an RPM package inside an image (section 10.4).

An example standard kernel package name is "kernel-2.6.18-274.3.1.el5". The actual one suited to a cluster varies according to the distribution used. Packages with names that begin with "kernel-devel-" are development packages that can be used to compile custom kernels, and are not required when installing standard distribution kernels.

When upgrading a kernel, an extra consideration for a head or software image is that third-party drivers may need to be re-installed or rebuilt for that kernel. This is necessary in the case of OFED drivers used instead of the standard distribution drivers. Details on re-installing or rebuilding such OFED drivers are given in section 13.6.

When installing a kernel, extra considerations for software images are:

- The kernel must also be explicitly set in CMDaemon (section 10.3.3) before it may be used by the regular nodes.

- Some GRUB-related errors with a text such as "`grubby fatal error: unable to find a suitable template`" typically show up during the installation of a kernel package in the software image. These occur due to a failure to find the partitions when running the post-install scripts in the chroot environment. They can simply be ignored because the nodes do not boot from partitions configured by GRUB.
- The ramdisk must be regenerated using the `createramdisk` command (section 10.4.3).

As is standard for Linux, both head or regular nodes must be rebooted to use the new kernel.

10.3.2 Excluding Kernels And Other Packages From Updates

Specifying A Kernel Or Other Package For Update Exclusion

Sometimes it may be desirable to exclude the kernel from updates on the head node.

- When using `yum`, to prevent an automatic update of a package, the package is listed after using the `--exclude` flag. So, to exclude the kernel from the list of packages that should be updated, the following command can be used:

  ```
  yum --exclude kernel update
  ```

  To exclude a package such as `kernel` permanently from all YUM updates, without having to specify it on the command line each time, the package can instead be excluded inside the repository configuration file. YUM repository configuration files are located in the `/etc/yum.repos.d` directory, and the packages to be excluded are specified with a space-separated format like this:

  ```
  exclude = <package 1> <package 2> ...
  ```

- The `zypper` command can also carry out the task of excluding the kernel package from getting updated when updating. To do this, the kernel package is first locked (prevented from change) using the `addlock` command, then the `update` command is run, and finally the kernel package is unlocked again using the `removelock` command:

  ```
  zypper addlock kernel
  zypper update
  zypper removelock kernel
  ```

Specifying A Repository For Update Exclusion

Sometimes it is useful to exclude an entire repository from an update on the head node. For example, the administrator may wish to exclude updates to the parent distribution, and only want updates for the cluster manager to be pulled in. In that case, a construction like the following may be used to specify that only the repository IDs matching the glob `cm*` are used, from the repositories in `/etc/yum.repos.d/`:

```
[root@ddnmon61 ~]# yum repolist
...
repo id              repo name            status
base                 CentOS-6 - Base     6,356+11
cm-rhel6-         Cluster Manager    316+13
cm-rhel6--updates Cluster Manager       514
extras               CentOS-6 - Extras         14
updates              CentOS-6 - Updates       391
repolist: 7,591
[root@ddnmon61 ~]# yum --disablerepo=* --enablerepo=cm* update
```

Blocking Major OS Updates With The `cm-dist-limit-<DIST>` package

Sometimes DirectMon may install the package:

```
cm-dist-limit-<DIST>
```

where `<DIST>` is `rhel6.3`, `rhel6.4`, `sl6.3`, `sl6.4`, `centos6.3`, or `centos6.4`.

When the package is installed, it deliberately prevents `yum update` from working if the update is going to install packages that require a higher version of `cm-dist-limit-<DIST>`. That is, it is a locking mechanism.

For example, when trying to upgrade from SL6.3 to SL6.4, and if `cm-dist-limit-sl6.3` is installed, then the only way to update the operating system to SL6.4, is to remove the `cm-dist-limit-sl6.3` package. Any dependent packages will also be removed automatically by YUM. A subsequent `yum update` installs new versions of the removed packages, and the `cm-dist-limit-sl6.4` package is installed automatically.

The reason for the `cm-dist-limit-<DIST>` packages is to prevent the OS from upgrading. This is helpful if some packages are not compatible with upgrades.

Currently, DirectMon only uses the `cm-dist-limit-<DIST>` package to block `intel-mic-*` package updates. The `intel-mic-*` packages, used by MICs, are not compatible with a `yum update` from an OS based on RHEL6.3 that updates the OS to RHEL6.4.

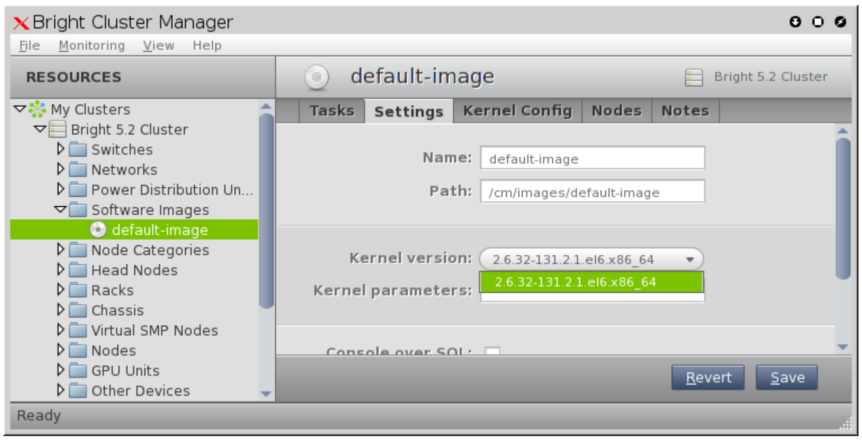
10.3.3 Updating A Kernel In A Software Image

A kernel can be updated in the software image in several ways, including as a package installation using the chroot environment (section 10.4).

Parent distributions derived from RHEL6 and above are by default configured, by the distribution itself, so that only up to 3 kernel images are kept during YUM updates. The distribution default value is however overridden by a DirectMon default value, so that kernel images are never removed during YUM updates, by default.

For a software image, if the kernel is updated by the package manager, then the kernel is not used on reboot until it is explicitly enabled with either `cmgui` or `cmsh`.

- To enable it in `cmgui`, the `Software Images` resource is selected, and the specific image item is selected. The `Settings` tabbed pane for that particular software image is opened, the new kernel version is selected from the "`Kernel version`" drop-down menu, and the `Save` button is clicked. Saving the version builds a new initial ramdisk containing the selected kernel (figure 10.1).



Name: default-image
Path: /cm/images/default-image
Kernel version: 2.6.32-131.2.1.el6.x86_64
Kernel parameters:
Console over SOL:

Figure 10.1: Updating A Software Image Kernel In `cmgui`

- To enable the updated kernel from `cmsh`, the `softwareimage` mode is used. The `kernelversion` property of a specified software image is then `set` and committed:

  Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% softwareimage
[ddnmon61]->softwareimage% use default-image
[ddnmon61->softwareimage[default-image]]% set kernelversion 2.6.32-
131.2.1.el6.x86_64
[ddnmon61->softwareimage*[default-image*]]% commit
```

10.3.4 Setting Kernel Options For Software Images

A standard kernel can be booted with special options that alter its functionality. For example, a kernel can boot with `apm=off`, to disable Advanced Power Management, which is sometimes useful as a workaround for nodes with a buggy BIOS that may crash occasionally when it remains enabled.

To enable booting with this kernel option setting in `cmgui`, the "`Software Images`" resource is selected, and the specific image item is selected (figure 10.1). The `Settings` tabbed pane for that particular software image is opened, and the "`Kernel parameters`" value is set to `apm=off`.

In `cmsh` the value of "`kernel parameters`" in `softwareimage` mode for the selected image can be set as in the following example:

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% softwareimage
[ddnmon61]->softwareimage% use default-image
[ddnmon61->softwareimage[default-image]]% set kernelparameters apm=off
[ddnmon61->softwareimage*[default-image*]]% commit
```

Often kernel options load up modules and their parameters. Making module loading persist after reboot and setting module loading order is covered in section 6.3.2

Some kernel options may require changes to be made in the BIOS settings in order to function.

10.3.5 Kernel Driver Modules

DDN provides some packages which install new kernel drivers or update kernel drivers. Such packages generally require the `kernel-devel` package. In this section, the `kernel-devel-check` utility is first described, followed by the various drivers that DDN provides.

Kernel Driver Modules: `kernel-devel-check` Compilation Check

The distribution's `kernel-devel` package is required to compile kernel drivers for its kernel. It must be the same version and release as the kernel running on the node.

To check the head node and software images for the installation status of the `kernel-devel` package, the DirectMon utility `kernel-devel-check` is run from the head node:

Example

```
[root@mycluster ~]# kernel-devel-check
Head node: mycluster
  Found kernel development rpm package kernel-devel-2.6.32-131.2.1.el6.\
x86_64

Software image: default-image
  No kernel development directories found, probably no kernel developme\
nt package installed.
  Kernel development rpm package kernel-devel-2.6.32-131.2.1.el6.x86_64\
not found
  If needed, try to install the kernel development package with:
  # chroot /cm/images/default-image yum install kernel-devel-2.6.32-131\
.2.1.el6.x86_64

Software image: default-image1
  No kernel development directories found, probably no kernel developme\
nt package installed.
  Kernel development rpm package kernel-devel-2.6.32-131.2.1.el6.x86_64\
 not found
  If needed, try to install the kernel development package with:
  # chroot /cm/images/default-image1 yum install kernel-devel-2.6.32-13\
1.2.1.el6.x86_64
```

As suggested by the output of `kernel-devel-check`, running a command on the head node such as:

```
[root@mycluster ~]# chroot /cm/images/default-image1 yum install kernel\
-devel-2.6.32-131.2.1.el6.x86_64
```

installs a `kernel-devel` package, to the software image called `default-image1` in this case. The package version suggested corresponds to the kernel version set for the image, rather than necessarily the latest one that the distribution provides.

Kernel Driver Modules: Improved Intel Wired Ethernet Drivers

Improved Intel wired Ethernet drivers—what they are: The standard distributions provide Intel wired Ethernet driver modules as part of the kernel they provide. DDN provides an improved version of the drivers with its own `intel-wired-ethernet-drivers` package. The package contains more recent versions of the Intel wired Ethernet kernel drivers: `e1000`, `e1000e`, `igb`, `igbvf`, `ixgbe` and `ixgbevf`. They often work better than standard distribution modules when it comes to performance, features, or stability.

Improved Intel wired Ethernet drivers—replacement mechanism: The improved drivers can be installed on all nodes.

For head nodes, the standard Intel wired Ethernet driver modules on the hard drive are overwritten by the improved versions during package installation. Backing up the standard driver modules before installation is recommended, because it may be that some particular hardware configurations are unable to cope with the changes, in which case reverting to the standard drivers may be needed.

For regular nodes, the standard distribution wired Ethernet drivers are not overwritten into the provisioner's software image during installation of the improved drivers package. Instead, the standard driver modules are removed from the kernel and the improved modules are loaded to the kernel during the `init` stage of boot.

For regular nodes in this "unwritten" state, removing the improved drivers package from the software image restores the state of the regular node, so that subsequent boots end up with a kernel running the standard distribution drivers from on the image once again. This is useful because it allows a very close-to-standard distribution to be maintained on the nodes, thus allowing better distribution support to be provided for the nodes.

If the software running on a fully-booted regular node is copied over to the software image, for example using the "`Grab to image`" button (section 10.5.2), this will write the improved driver module into the software image. Restoring to the standard version is then no longer possible with simply removing the improved drivers packages. This makes the image less close-to-standard, and distribution support is then less easily obtained for the node.

Thus, after the installation of the package is done on a head or regular node, for every boot from the next boot onwards, the standard distribution Intel wired Ethernet drivers are replaced by the improved versions for fully-booted kernels. This replacement occurs before the network and network services start. The head node simply boots from its drive with the new drivers, while a regular node initially starts with the kernel using the driver on the software image, but then if the driver differs from the improved one, the driver is unloaded and the improved one is compiled and loaded.

Improved Intel wired Ethernet drivers—installation: The drivers are compiled on the fly on the regular nodes, so a check should first be done that the `kernel-devel` package is installed on the regular nodes (section 10.3.5).

If the regular nodes have the `kernel-devel` package installed, then the following `yum` commands are issued on the head node, to install the package on the head node and in the `default-image`:

Example

```
[root@mycluster ~]# yum install intel-wired-ethernet-drivers
[root@mycluster ~]# chroot /cm/images/default-image
[root@mycluster /]# yum install intel-wired-ethernet-drivers
```

For SUSE, the equivalent `zypper` commands are used ("`zypper in`" instead of "`yum install`").

Kernel Driver Modules: CUDA Driver Installation

CUDA drivers are drivers the kernel uses to manage GPUs. These are compiled on the fly for nodes with GPUs in DirectMon. The details of how this is done is covered in the CUDA software section (section 13.5).

Kernel Driver Modules: OFED Stack Installation

By default, the distribution provides the OFED stack used by the kernel to manage the InfiniBand or RDMA interconnect. Installing a DirectMon repository OFED stack to replace the distribution version is covered in section 13.6. Some guidance on placement into initrd for the purpose of optional InfiniBand-based node provisioning is given in section 6.3.3.

10.4 Managing An RPM Package In A Software Image And Running It On Nodes

10.4.1 Installing From Head Via `chroot`: Installing Into The Image

Managing RPM packages (including the kernel) inside a software image is most easily done while on the head node, using a chroot mechanism with `rpm`, `yum`, or `zypper`.

The `rpm` command supports the `--root` flag. To install an RPM package inside the default software image while on the head node, the command is used as follows:

Example

```
rpm --root /cm/images/default-image -ivh /tmp/libxml2-2.6.16-6.x86_64.rpm
```

YUM and zypper implement the same functionality in slightly different ways. For example, all packages in the default image are updated with:

```
yum --installroot=/cm/images/default-image update
```

or

```
zypper --root /cm/images/default-image up
```

With the `chroot` command, the same result is accomplished by first chrooting into an image, and subsequently executing `rpm`, `yum`, or `zypper` commands without `--root` or `--installroot` arguments.

Excluding Packages And Repositories From The Image

Sometimes it may be desirable to exclude a package from an image.

- If using `yum --installroot`, then to prevent an automatic update of a package, the package is listed after using the `--exclude` flag. For example, to exclude the kernel from the list of packages that should be updated, the following command can be used:

  ```
  yum --installroot=/cm/images/default-image --exclude kernel update
  ```

  To exclude a package such as `kernel` permanently from all YUM updates, without having to specify it on the command line each time, the package can instead be excluded inside the repository configuration file of the image. YUM repository configuration files are located in the `/cm/images/default-image/etc/yum.repos.d` directory, and the packages to be excluded are specified with a space-separated format like this:

  ```
  exclude = <package 1> <package 2> ...
  ```

- The `zypper` command can also carry out the task of excluding a package from getting updated when during update. To do this, the package is first locked (prevented from change) using the `addlock` command, then the `update` command is run, and finally the package is unlocked again using the `removelock` command. For example, for the kernel package:

  ```
  zypper --root /cm/images/default-image addlock kernel
  zypper --root /cm/images/default-image update
  zypper --root /cm/images/default-image removelock kernel
  ```

- Sometimes it is useful to exclude an entire repository from an update to the image. For example, the administrator may wish to exclude updates to the base distribution, and only want DirectMon updates to be pulled into the image. In that case, a construction like the following may be used to specify that, for example, from the repositories listed in `/cm/images/default-image/etc/yum.repos.d/`, only the repositories matching the pattern `cm*` are used:

  ```
  [root@ddnmon61 ~]# cd /cm/images/default-image/etc/yum.repos.d/
  [root@ddnmon61 yum.repos.d]# yum --installroot=/cm/images/defaul\
  t-image --disablerepo=* --enablerepo=cm* update
  ```

10.4.2 Installing From Head Via `chroot`: Updating The Node

Rebooting the nodes that use the software images then has those nodes start up with the new images. Alternatively, the nodes can usually simply be updated without a reboot (section 6.6), if no reboot is required by the underlying Linux distribution.

10.4.3 Installing From Head Via `rpm --root`, `yum --installroot` Or `chroot`: Possible Issues

- The update process with YUM or zypper on an image will fail to start if the image is being provisioned by a provisioner at the time. The administrator can either wait for provisioning requests to finish, or can ensure no provisioning happens by setting the image to a locked state (section 6.4.7), before running the update process. The image can then be updated. The administrator normally unlocks the image after the update, to allow image maintenance by the provisioners again.

Example

```
[root@ddnmon61 ~]# cmsh -c "softwareimage use default-image; set lo\
cked yes; commit"
[root@ddnmon61 ~]# yum --installroot /cm/images/default-image update
[root@ddnmon61 ~]# cmsh -c "softwareimage use default-image; set lo\
cked no; commit"
```

- The `rpm --root` or `yum --installroot` command can fail if the versions between the head node and the version in the software image differ significantly. For example, installation from a Scientific Linux 5 head node to a Red Hat 6 software image is not possible with those commands, and can only be carried out with `chroot`.

- While installing software into a software image with an `rpm --root`, `yum --installroot` or with a `chroot` method is convenient, there can be issues if daemons start up in the image.

  For example, installation scripts that stop and re-start a system service during a package installation may successfully start that service within the image's chroot jail and thereby cause related, unexpected changes in the image. Pre- and post- (un)install scriptlets that are part of RPM packages may cause similar problems.

  DDN's RPM packages are designed to install under chroot without issues. However packages from other repositories may cause the issues described. To deal with that, the cluster manager runs the `chrootprocess` health check, which alerts the administrator if there is a daemon process running in the image. The `chrootprocess` also checks and kills the process if it is a `crond`.

- For some package updates, the distribution package management system attempts to modify the ramdisk image. This is true for kernel updates, many kernel module updates, and some other packages. Such a modification is designed to work on a normal machine. For a regular node on a cluster, which uses an extended ramdisk, the attempt does nothing.

  In such cases, a new ramdisk image must nonetheless be generated for the regular nodes, or the nodes will fail during the ramdisk loading stage during start-up (section 6.8.4).

  The ramdisk image for the regular nodes can be regenerated manually, using the `createramdisk` command (section 6.3.2).

- Trying to work out what is in the image from under chroot must be done with some care.

  For example, under chroot, running "`uname -a`" returns the kernel that is currently running—that is the kernel outside the chroot. This is typically not the same as the kernel that will load on the node from the filesystem under chroot. It is the kernel in the filesystem under chroot that an unwary administrator may wrongly expect to detect on running the `uname` command under chroot.

To find the kernel version that is to load from the image, the software image kernel version property (section 10.3.3) can be inspected using the cluster manager with:

Example

```
cmsh -c "softwareimage; use default-image; get kernelversion"
```

10.5 Managing Non-RPM Software In A Software Image And Running It On Nodes

Sometimes, packaged software is not available for a software image, but non-packaged software is. This section describes the installation of non-packaged software onto a software image in these two cases:

1. copying only the software over to the software image (section 10.5.1)
2. placing the software onto the node directly, configuring it until it is working as required, and syncing that back to the software image using DirectMon's special utilities (section 10.5.2)

As a somewhat related aside, completely overhauling the software image, including changing the base files that distinguish the distribution and version of the image is also possible. How to manage that kind of extreme change is covered separately in section 10.6.

However, this current section (10.5) is about modifying the software image with non-RPM software while staying within the framework of an existing distribution and version.

In all cases of installing software to a software image, it is recommend that software components be placed under appropriate directories under `/cm/shared` (which is actually outside the software image).

So, just as in the case for installing software to the head node in section 10.2.2, appropriate software components go under:

```
/cm/shared/
|-- apps
|-- docs
|-- examples
|-- licenses
`-- modulefiles
```

10.5.1 Managing The Software Directly On An Image

The administrator may choose to manage the non-packaged software directly in the correct location on the image.

For example, the administrator may wish to install a particular software to all nodes. If the software has already been prepared elsewhere and is known to work on the nodes without problems, such as for example library dependency or path problems, then the required files can simply be copied directly into the right places on the software image.

The `chroot` command may also be used to install non-packaged software into a software image. This is analogous to the `chroot` technique for installing packages in section 10.4:

Example

```
cd /cm/images/default-image/usr/src
tar -xvzf /tmp/app-4.5.6.tar.gz
chroot /cm/images/default-image
cd /usr/src/app-4.5.6
./configure --prefix=/usr
make install
```

Whatever method is used to install the software, after it is placed in the software image, the change can be implemented on all running nodes by running the `updateprovisioners` (section 6.2.4) and `imageupdate` (section 6.6.2) commands.

10.5.2 Managing The Software Directly On A Node, Then Syncing Node-To-Image

Why Sync Node-To-Image?

Sometimes, typically if the software to be managed is more complex and needs more care and testing than might be the case in section 10.5.1, the administrator manages it directly on a node itself, and then makes an updated image from the node after it is configured, to the provisioner.

For example, the administrator may wish to install and test an application from a node first before placing it in the image. Many files may be altered during installation in order to make the node work with the application. Eventually, when the node is in a satisfactory state, and possibly after removing any temporary installation-related files on the node, a new image can be created, or an existing image updated.

Administrators should be aware that until the new image is saved, the node loses its alterations and reverts back to the old image on reboot.

The node-to-image sync can be seen as the converse of the image-to-node sync that is done using `imageupdate` (section 6.6.2).

The node-to-image sync discussed in this section is done using the "`Grab to image`" or "`Synchronize image`" button in `cmgui`, or using the "`grabimage`" command with appropriate options in `cmsh`. The sync automatically excludes network mounts and parallel filesystems such as Lustre and GPFS, but includes any regular disk mounted on the node itself.

Some words of advice and a warning are in order here

- The cleanest, and recommended way, to change an image is to change it directly in the node image, typically via changes within a chroot environment (section 10.5.1).

- Changing the deployed image running on the node can lead to unwanted changes that are not obvious. While many unwanted changes are excluded because of the `excludelistgrab*` lists during a node-to-image sync, there is a chance that some unwanted changes do get captured. These changes can lead to unwanted or even buggy behavior. The changes from the original deployed image should therefore be scrutinized with care before using the new image.

- For scrutiny, the command:

  ```
  vimdiff <(cd image1; find . | sort) <(cd image2; find . | sort)
  ```

run from /cm/images/ shows the changed files for image directories image1 and image2, with uninteresting parts folded away.

Node-To-Image Sync Using cmgui

In cmgui, saving the node state to a new image is done by selecting a node from the Nodes resource, and then selecting the Tasks tab. The "Software image" section of the tab has these two buttons that can carry out the sync from node to software image (figure 10.2):

1. The "Grab to image" button. This opens up a dialog offering a new image to sync to. This button creates a new image, wiping out whatever (if anything) is in the selected image, except for a list of excluded items.

   The excluded items are specified in the "Exclude list grabbing to a new image" list, available under the "Node Categories" resource, under the Settings tab. The exclude list is known as excludelistgrabnew (page 386) in cmsh.

2. The "Synchronize image" button. This does a sync from the node back to the software image that the node is provisioned with, using evaluation based on file change detection between the node and the image. It is thus a synchronization to the already existing software image that is currently in use by the node.

   The items that it excludes from the synchronization are specified in the "Exclude list image grab" list, available under the "Node Categories" resource, under the Settings tab. This exclude list is known as excludelistgrab (page 386) in cmsh.

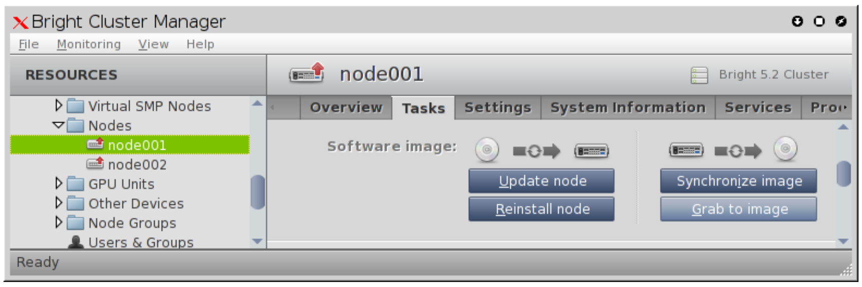


Figure 10.2: Synchronizing From A Node To A Software Image In cmgui

The exclude lists are there to ensure, among other things, that the configuration differences between nodes are left alone for different nodes with the same image. The exclude lists are simple by default, but they conform in structure and patterns syntax in the same way that the exclude lists detailed in section 6.4.7 do, and can therefore be quite powerful.

Both buttons ask for reconfirmation of the action before going ahead with it. The "Grab to image" button in addition also allows selection of the image to be used for syncing. In the dialog for both buttons, a dry-run checkbox is marked by default to allow the administrator to see what would happen during the node-to-image sync, without actually having the files written over to the image. Logs of the run are viewable by clicking on the "Provisioning Log" button, also in the Tasks tab. If the administrator is sure that the run works as wanted, then the node-to-image sync can be done with the dry-run checkbox unmarked.

The images that are available for selection with the "Grab to image" button can be existing images, while the image that the Synchronize image button syncs to is the existing image. If such existing images are known to work well with nodes, then overwriting them with a new image on a production system may be reckless. A wise administrator who has prepared a node that is to write an image would therefore follow a process similar to the following instead of simply overwriting an existing image:

1. A new image is created into which the node state can be written. This is easiest to do by using the "Software Images" resource to clone a new image. The node state with the software installed on it would then be saved using the "Grab to image"button, and choosing the cloned image name as the image to save it to.

2. A new category is then cloned from the old category, and within the new category the old image is changed to the new image. This is easiest to do from the Overview tab of the "Node Categories" resource. Here the original category of the node is selected and cloned. The cloned category is then selected, opened and the old image within it—in the "Software image" section—is changed to the new, cloned, image, and the result is saved.

3. Some nodes are set to the new category to test the new image. This is done from the Nodes resource, selecting the node, and from its Settings tab, changing its category to the new one, and saving the change.

4. The nodes in the new category are made to pick up and run their new images. This can be done with a reboot.

5. After sufficient testing, all remaining nodes can be moved to using the new image, and the old image is removed if no longer needed.

Node-To-Image Sync Using cmsh

The preceding cmgui method can alternatively be carried out using cmsh commands. The cmsh equivalent to the "Synchronize image" button is the grabimage command, available from device mode. The cmsh equivalent to the "Grab to image" button is the grabimage -i command, where the -i option specifies the new image it will write to. As before, that image must be created or cloned beforehand.

Cloning an image for use, setting a category of nodes that will use it, and then synchronizing a node that has the new software setup over to the new image on the provisioner might be carried out as follows via cmsh:

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% softwareimage
[ddnmon61->softwareimage]% clone default-image default-image1
[ddnmon61->softwareimage*[default-image1]]% commit
[ddnmon61->softwareimage[default-image1]]% category
[ddnmon61->category]% clone default default1
[ddnmon61->category*[default1*]]% commit
[ddnmon61->category[default1]]% set softwareimage default-image1
[ddnmon61->category*[default1*]]% commit
```

```
[ddnmon61->category[default1]]% device
[ddnmon61->device]% grabimage -w -i default-image1 node001
[ddnmon61->device]%
Mon Jul 18 16:13:00 2011 [notice] ddnmon61: Provisioning started on\
 node node001
[ddnmon61->device]%
Mon Jul 18 16:13:04 2011 [notice] ddnmon61: Provisioning completed on\
 node node001
```

The `grabimage` command without the `-w` option simply does a dry-run so that the user can see in the provisioning logs what should be grabbed, without having the changes actually carried out. Running `grabimage -w` instructs CMDaemon to really write the image.

When writing out the image, two exclude lists may be used:

- The `excludelistgrabnew` object. This is used with `grabimage` with the `-i` option. The list can be accessed and edited under `cmsh`, in its `category` mode for a node image, as the `excludelistgrabnew` object. It corresponds to the "`Exclude list grabbing to a new image`" exclusion list associated with the "`Grab to image`" button (page 384) in `cmgui`.
- The `excludelistgrab` object. This is used with the `grabimage` command, run without the `-i` option. The list can be accessed and edited under `cmsh`, in its `category` mode for a node image, as the `excludelistgrab` object. It corresponds to the "`Exclude list image grab`" exclusion list associated with the "`Synchronize image`" button (page 384) in `cmgui`.

10.6 Creating A Custom Software Image

By default, the software image used to boot non-head nodes is based on the same version and release of the Linux distribution as used by the head node. However, sometimes an image based on a different distribution or a different release from that on the head node may be needed.

A custom software image is created typically by building an entire filesystem image from a regular node. The node, which is never a head node, is then called the *base host*, with the term "base" used to indicate that it has no additional cluster manager packages installed. The distribution on the base host, is called the *base distribution* and is a selection of packages derived from the *parent distribution* (Red Hat, Scientific Linux etc). A *base distribution package* is a package or rpm that is directly provided by the vendor of the parent distribution which the base distribution is based on, and is not provided by DirectMon.

Creating a custom software image consists of two steps. The first step (section 10.6.1) is to create a *base (distribution) archive* from an installed base host. The second step (section 10.6.2) is to create the image from the base archive using a special utility, `cm-create-image`.

10.6.1 Creating A Base Distribution Archive From A Base Host

Structure Of The Base Distribution Archive

The step of creating the base distribution archive is done by creating an archive structure containing the files that are needed by the non-head node.

The filesystem that is archived in this way can differ from the special way that a Linux distribution unpacks and installs its filesystem on to a machine. This is because the distribution installer often carries out extra changes, for example in GRUB boot configuration. The creation of the base distribution archive is therefore a convenience to avoid working with the special logic of a distribution installer, which will vary across distributions and versions. Instead, the filesystem and contents of a node on which this parent distribution is installed—i.e. the end product of that logic—is what is dealt with.

The archive can be a convenient and standard `tar.gz` file archive (sometimes called the "base tar"), or, taking the step a little further towards the end result, the archive can be a fully expanded archive file tree.

Repository Access Considerations When Intending To Build A Base Distribution Archive

For convenience, the archive should be up-to-date. So, the base host used to generate the base distribution archive should ideally have updated files. If, as is usual, the base host is a regular node, then it should ideally be up to date with the repositories that it uses. Therefore running `yum update` or `zypper up` on the base host image, and then provisioning the image to the base host, is recommended in order to allow the creation of an up-to-date base distribution archive.

However sometimes updates are not possible or desirable for the base host. This means that the base host archive that is put together from the base host filesystem is an un-updated archive. The custom image that is to be created from the archive must then be also be created without accessing the repositories, in order to avoid dependency problems with the package versions. Exclusion of access to the repositories is possible by specifying options to the `cm-create-image` command, and is described in section 10.6.2.

Examples Of How To Build A Base Distribution Archive

In the following example, a base distribution `tar.gz` archive `/tmp/BASEDIST.tar.gz` is created from the base host `basehost64`:

Example

```
ssh root@basehost64 \
"tar -cz \
--exclude /etc/HOSTNAME --exclude /etc/localtime \
--exclude /proc --exclude /lost+found --exclude /sys \
--exclude /root/.ssh --exclude /var/lib/dhcpcd/* \
--exclude /media/floppy --exclude /etc/motd \
--exclude /root/.bash_history --exclude /root/CHANGES \
--exclude /etc/udev/rules.d/*persistent*.rules \
--exclude /var/spool/mail/* --exclude /rhn \
--exclude /etc/sysconfig/rhn/systemid --exclude /tmp/* \
--exclude /var/spool/up2date/* --exclude /var/log/* \
--exclude /etc/sysconfig/rhn/systemid.save \
--exclude /root/mbox --exclude /var/cache/yum/* \
--exclude /etc/cron.daily/rhn-updates /" > /tmp/BASEDIST.tar.gz
```

Or alternatively, a fully expanded archive file tree can be created from `basehost64` by `rsync`ing to an existing directory (here it is `/cm/images/new-image`):

Example

```
rsync -av --numeric-ids \
--exclude='/etc/HOSTNAME' --exclude='/etc/localtime' --exclude='/proc'\
--exclude='/lost+found' --exclude='/sys' --exclude='/root/.ssh' \
--exclude='/var/lib/dhcpcd/*' --exclude='/media/floppy' \
--exclude='/etc/motd' --exclude='/root/.bash_history' \
--exclude='/root/CHANGES' --exclude='/var/spool/mail/*'\
--exclude='/etc/udev/rules.d/*persistent*.rules' \
--exclude='/rhn' --exclude='/etc/sysconfig/rhn/systemid' \
--exclude='/etc/sysconfig/rhn/systemid.save' --exclude='/tmp/*' \
--exclude='/var/spool/up2date/*' --exclude='/var/log/*' \
--exclude='/root/mbox' --exclude='/var/cache/yum/*' \
--exclude='/etc/cron.daily/rhn-updates' \
root@basehost64:/ /cm/images/new-image/
```

The first step, that of building the base archive, is now done.

10.6.2 Creating The Software Image With `cm-create-image`

The second step, that of creating the image from the base archive, now needs to be done. This uses the `cm-create-image` utility, which is part of the `cluster-tools` package.

The `cm-create-image` utility uses the base archive as the base for creating the image. By default, it expects that the base distribution repositories be accessible just in case files need to be fetched from a repository package.

Thus, when the `cm-create-image` utility is run with no options, the image created mostly picks up the software only from the base archive. However, the image picks up software from the repository packages:

- if it is required as part of a dependency, or
- if it is specified as part of the package selection file (page 392).

If a repository package file is used, then it should be noted that the repository package files may be more recent compared with the files in the base archive. This can result in an image with files that are perhaps unexpectedly more recent in version than what might be expected from the base archive, which may cause compatibility issues. To prevent this situation, the `--exclude` option (section 10.2) can be used to exclude updates for the packages that are not to be updated.

Repository access can be directly to the online repositories provided by the distribution, or it can be to a local copy. For RHEL, online repository access can be activated by registering with the Red Hat Network, as described in appendix M.1, and for SUSE, online repository access can be activated by registering with Novell, as described in appendix M.2. An offline repository can be constructed as described in section 10.6.3.

Usage Of The `cm-create-image` Command

The usage information for `cm-create-image` lists options and examples:

```
USAGE: cm-create-image <OPTIONS1> [OPTIONS2]

OPTIONS1:
```

```
---------
-a | --fromarchive <archive>   Create software image from archive file
                               of supported base distribution. Supported
                               file formats are .tar, .tgz, .tar.gz,
                               .tbz, and .tar.bz2. The extension must
                               match the format.
-d | --fromdir   <dir path>    Create software image from existing
                               directory that already has valid base
                               distribution.
-h | --fromhost  <hostname>    Create software image from running host
-n | --imagename <name>        Name of software image to create in
                               cluster management daemon database.

OPTIONS2:
---------
-i | --imagedir <dir name>     Name of directory to be created in
                               /cm/images.
                               Contents of archive file are extracted
                               into this directory (default: name
                               specified with -n).
-r | --recreate                Recreate directory specified by -i or
                               default, if it exists.
                               Default behavior: directory is overwritten
-s | --skipdist                Skip install of base distribution packages
-e | --excludecmrepo           Do not copy default cluster manager repo
                               files.  (Use this option when the repo
                               files have been already modified in the
                               image directory, and hence must not be
                               overwritten.)
-f | --forcecreate             Force non-interactive mode
-u | --updateimage             If image specified by -n already exists,
                               then it is updated, with the new parameters
-b | --basedistrepo <file>     Use this as the repo configuration file
                               for fetching required distribution
                               packages (cannot be used with -e).
-c | --cmrepo <file>           Use this as the repo configuration file
                               for fetching required cluster manager
                               packages (cannot be used with -e).
-m | --minimal                 Install only a minimal set of packages,
                               relevant for the cluster management
                               daemon to function. Use with -s option, to
                               also prevent additional distribution
                               packages from being installed.
-w | --hwvendor                Install hardware vendor specific packages.
                               Valid choices are: %s
-l | --resolvconf              resolv.conf to be used inside image dir,
                               during image creation (by default
                               /etc/resolv.conf from head node is used)
-x | --excludecm       <list>  List of CM packages to exclude (comma-
                               separated)
-j | --excludedist     <list>  List of distribution packages to exclude
                               (comma separated)
-q | --excludehwvendor <list>  List of hardware vendor packages to
                               exclude (comma-separated)
```

```
EXAMPLES:
---------
1. cm-create-image -a /tmp/RHEL5.tar.gz -n rhel5-image
2. cm-create-image -a /tmp/RHEL5.tar.gz -n rhel5-image -i /cm/images/te\
st-image
3. cm-create-image -d /cm/images/SLES11-image -n sles11-image
4. cm-create-image -h node001 -n node001-image
5. cm-create-image -a /tmp/RHEL5.tar.gz -n rhel5-image -i /cm/images/new\
-image -r
6. cm-create-image -a /tmp/RHEL5.tar.gz -n rhel5-image -i /cm/images/new\
-image -u
7. cm-create-image -d /cm/images/new-image -n bio-image -s -e
8. cm-create-image -d /cm/images/new-image -n bio-image -s -b /tmp/rhel5\
-updates.repo
9. cm-create-image -d /cm/images/new-image -n bio-image -s -c /tmp/cm-rh\
el5.repo
10. cm-create-image -d /cm/images/new-image -n bio-image -s -m
```

Explanations Of The Examples In Usage Of `cm-create-image`
Explanations of the 10 examples in the usage text follow:

1. In the following, a base distribution archive file, `/tmp/RHEL5.tar.gz`, is written out to a software image named `rhel5-image`:

   ```
   cm-create-image --fromarchive /tmp/RHEL5.tar.gz --imagename rhel5-\
   image
   ```

   The image with the name `rhel5-image` is created in the CMDaemon database, making it available for use by `cmsh` and `cmgui`. If an image with the above name already exists, then `/cm/create-image` will exit and advise the administrator to provide an alternate name.

   By default, the image name specified sets the directory into which the software image is installed. Thus here the directory is `/cm/images/rhel5-image/`.

2. Instead of the image getting written into the default directory as in the previous item, an alternative directory can be specified with the `--imagedir` option. Thus, in the following, the base distribution archive file, `/tmp/RHEL5.tar.gz` is written out to the `/cm/images/test-image` directory. The software image is given the name `rhel5-image`:

   ```
   cm-create-image --fromarchive /tmp/RHEL5.tar.gz --imagename rhel5-\
   image --imagedir /cm/images/test-image
   ```

3. If the contents of the base distribution file tree have been transferred to a directory, then no extraction is needed. The `--fromdir` option can then be used with that directory. Thus, in the following, the archive has already been transferred to the directory `/cm/images/SLES11-image`, and it is that directory which is then used to place the image under a directory named `/cm/images/sles11-image/`. Also, the software image is given the name `sles11-image`:

```
cm-create-image --fromdir /cm/images/SLES11-image --imagename sles\
11-image
```

4. A software image can be created from a running node using the `--fromhost` option. This option makes `cm-create-image` behave in a similar manner to `grabimage` (section 10.5.2) in `cmsh`. It requires passwordless access to the node in order to work. Nodes outside the cluster can also be used. An image named `node001-image` can then be created from a running node named `node001` as follows:

   ```
   cm-create-image --fromhost node001 --imagename node001-image
   ```

   By default the image goes under the `/cm/images/node001-image/` directory.

5. If the destination directory already exists, the `--recreate` option can be used to recreate the existing directory. The administrator should be aware that this means removal of the content of any existing directory of the same name. Thus, in the following, the content under `/cm/images/new-image/` is deleted, and new image content is taken from the base distribution archive file, `/tmp/RHEL5.tar.gz` and then placed under `/cm/images/new-image/`. Also, the software image is given the name `rhel5-image`:

   ```
   cm-create-image --fromarchive /tmp/RHEL5.tar.gz --imagename rhel5-\
   image --imagedir /cm/images/new-image --recreate
   ```

   If the `--recreate`, option is not used, then the contents are simply overwritten, that is, the existing directory contents are copied over by the source content. It also means that old files on the destination directly may survive unchanged because the new source may not have filenames matching those.

6. The destination directory can also just be updated without removing the existing contents, by using the option `--updateimage`. The option is almost the same as the “contents are simply overwritten” behavior described in example 5, but it actually works like an `rsync` command. Thus, in the following, the base distribution archive file, `/tmp/RHEL5.tar.gz`, is used to update the contents under the directory `/cm/images/new-image/`. The name of the image is also set to `rhel5-image`.

   ```
   cm-create-image --fromarchive /tmp/RHEL5.tar.gz --imagename rhel5-\
   image --imagedir /cm/images/new-image --updateimage
   ```

7. With the default DirectMon, the head node provisions a software image based on the parent distribution to the other nodes. The software image which runs on the nodes provides a selection of distribution packages from the parent distribution.

   The default software image is thus a selection of Red Hat packages, if the head node uses Red Hat, or a selection of SUSE packages if the head node uses SUSE, and so on. The other packages for the software image are supplied by DDN.

When creating a custom software image, and if using the `--skipdist` flag with `cm-create-image`, then DirectMon packages are added to the software image, but no parent distribution packages are added. Thus in the following, the packages made available to `cm-create-image` in the directory `/cm/images/new-image`, are installed into the image named `bio-image`; however, no packages matching parent distribution packages are installed (because of the `--skipdist` option). Furthermore, transfer of the packages takes place only if they are newer than the files already in the `bio-image` image (because of the `--updateimage` option):

```
cm-create-image --fromdir /cm/images/new-image --imagename bio-imag\
e --skipdist --updateimage
```

So, only DirectMon packages are updated to the image `bio-image` in the directory `/cm/images/bio-image`.

8. The `--basedistrepo` flag is used together with a `.repo` file. The file defines the base distribution repository for the image. The file is copied over into the repository directory of the image, (`/etc/yum.repos.d/` for Red Hat and similar, or `/etc/zypp/repos.d/` for SLES).

9. The `--cmrepo` flag is used together with a `.repo` file. The file defines the cluster manager repository for the image. The file is copied over into the repository directory of the image, (`/etc/yum.repos.d/` for Red Hat and similar, or `/etc/zypp/repos.d/` for SLES).

10. The `--minimal` flag tells `cm-create-image` to install only a limited number of DDN packages and its dependencies, which are required for the cluster management daemon to run the node. This also means that no DDN configuration RPMS will be installed, and all existing system configuration files will remain untouched. The minimal package selection list is read from `/cm/local/apps/cluster-tools/config/minimal`. The word “minimal” primarily refers to DirectMon, and it is still required to use the `--skipdist` option explained earlier, in order to prevent additional parent distribution packages from being installed. This is because “minimal” in the context of the parent distribution can vary a lot depending on the requirements of different users, and is beyond the scope of `cm-create-image`.

```
cm-create-image --fromdir /cm/images/new-image --imagename bio-imag\
e --skipdist --minimal
```

Package Selection Files In `cm-create-image`

Regarding explanation 7 in the preceding explanations text, the selection of packages on the head node is done using a *package selection file*.

Package selection files are available in `/cm/local/apps/cluster-tools/config/`. For example, if the base distribution of the software image being created is CentOS5, then the configuration file used is:

```
/cm/local/apps/cluster-tools/config/CENTOS5-config-dist.xml
```

The package selection file is made up of a list of XML elements, specifying the name of the package, architecture and image type. For example:

```
...
<package image="slave" name="apr" arch="x86_64"/>
<package image="slave" name="apr-util" arch="x86_64"/>
<package image="slave" name="atk-devel" arch="x86_64"/>
<package image="slave" name="autoconf" arch="noarch"/>
...
```

The minimal set of packages in the list defines the minimal distribution that works with DirectMon, and is the base-distribution set of packages, which may not work with some features of the distribution or DirectMon. To this minimal set the following packages may be added to create the custom image:

- Packages from the standard repository of the parent distribution. These can be added to enhance the custom image or to resolve a dependency of DirectMon. For example, in the (parent) Red Hat distribution, packages can be added from the (standard) main Red Hat channel to the base-distribution.
- Packages from outside the standard repository, but still from inside the parent distribution. These can be added to enhance the custom image or to resolve a dependency of DirectMon. For example, outside the main Red Hat channel, but still within the parent distribution, there is a supplementary packages channel (Red Hat 5) or an optional packages channel (Red Hat 6). Packages from these optional/supplementary channels can be added to the base-distribution to enhance the capabilities of the image or resolve dependencies of DirectMon. Section 13.5.1 considers an example of such a dependency for the CUDA package.

Unless the required distribution packages and dependencies are installed and configured, particular features of DirectMon, such as CUDA, cannot work correctly or cannot work at all.

The package selection file also contains entries for the packages that can be installed on the head (`image="master"`) node. Therefore non-head node packages must have the `image="slave"` attribute.

Kernel Module Selection By `cm-create-image`

For an image created by `cm-create-image`, with a distribution *<dist>*, the default list of kernel modules to be loaded during boot are read from the file `/cm/local/apps/cluster-tools/config/`*<dist>*`-slavekernelmodules`.

<dist> can take the value `CENTOS5`, `CENTOS6`, `RHEL5`, `RHEL6`, `SL5`, `SL6`, `SLES11`, `SLES11sp1`, `SLES11sp2`, or `OEL5`.

If custom kernel modules are to be added to the image, they can be added to this file.

Output And Logging During A `cm-create-image` Run

The `cm-create-image` run goes through several stages: validation, sanity checks, finalizing the base distribution, copying DirectMon repository files, installing distribution package, finalizing image services, and installing DirectMon packages. An indication is given if any of these stages fail.

Further detail is available in the logs of the `cm-create-image` run, which are kept in `/var/log/cmcreateimage.log.`<*image name*>, where <*image name*> is the name of the built image.

Default Image Location

The default-image is at `/cm/images/default-image`, so the image directory can simply be kept as `/cm/images/`.

During a `cm-create-image` run, the `--imagedir` option allows an image directory for the image to be specified. This must exist before the option is used.

More generally, the full path for each image can be set:

- In `cmgui` in the "`Software Images`" resource, by filling in the box for `Path` in the `Settings` tabbed pane for the image
- In `cmsh` within `softwareimage` mode, for example:

```
[ddnmon61->softwareimage]% set new-image path /cm/higgs/new-images
```

- At the system level, the images or image directory can be symlinked to other locations for organizational convenience

Workload Manager Reconfiguration On The Custom Regular Node Image

After a custom regular node image has been created by `cm-create-image`, and the workload manager on the custom node image is to be managed by DirectMon, then the manager typically needs to be reconfigured. This can be done by running the `wlm-setup` utility using the `--image` option with the path to the custom node image (section 9.3).

10.6.3 Configuring Local Repositories For Linux Distributions, And For The DirectMon Package Repository, For A Software Image

Using local instead of remote repositories can be useful in the following cases:

- for clusters that have restricted or no internet access.
- for the RHEL and SUSE Linux distributions, which are based on a subscription and support model, and therefore do not have free access to their repositories.
- for creating a custom image with the `cm-create-image` command introduced in section 10.6.2, using local base distribution repositories.

The administrator can choose to access an online repository provided by the distribution itself via a subscription as described in appendix M.

Another way to set up a repository is to set it up as a local repository, which may be offline, or perhaps set up as a locally-controlled proxy with occasional, restricted, updates from the distribution repository.

In the three procedures that follow, the first two procedures explain how to create and configure a local offline SLES zypper or RHEL YUM repository for the subscription-based base distribution packages. These first two procedures assume that the corresponding ISO/DVD has been purchased/downloaded from the appropriate vendors. The third procedure then explains how to create a local offline YUM repository from the DirectMon ISO for CentOS so that a cluster that is completely offline still has a complete and consistent repository access.

Thus, a summary list of what these procedures are about is:

- Setting up a local repository for SLES (page 395)
- Setting up a local repository for RHEL (page 395)
- Setting up a local repository for CentOS and DDN from the DirectMon ISO for CentOS (page 396)

Configuring Local Repositories For SLES For A Software Image

For SLES11 SP0, SLES11 SP1, and SLES11 SP2, the required packages are spread across two DVDs, and hence two repositories must be created. Assuming the image directory is `/cm/images/sles11sp1-image`, while the names of the DVDs are `SLES-11-SP1-SDK-DVD-x86_64-GM-DVD1.iso` and `SLES-11-SP1-DVD-x86_64-GM-DVD1.iso`, then the contents of the DVDs can be copied as follows:

```
mkdir /mnt1 /mnt2
mkdir /cm/images/sles11sp1-image/root/repo1
mkdir /cm/images/sles11sp1-image/root/repo2
mount -o loop,ro SLES-11-SP1-SDK-DVD-x86_64-GM-DVD1.iso /mnt1
cp -ar /mnt1/* /cm/images/sles11sp1-image/root/repo1/
mount -o loop,ro SLES-11-SP1-DVD-x86_64-GM-DVD1.iso  /mnt2
cp -ar /mnt2/* /cm/images/sles11sp1-image/root/repo2/
```

The two repositories can be added for use by `zypper` in the image, as follows:

```
chroot /cm/images/sles11sp1-image
zypper addrepo /root/repo1 "SLES11SP1-SDK"
zypper addrepo /root/repo2 "SLES11SP1"
exit (chroot)
```

Configuring Local Repositories For RHEL For A Software Image

For RHEL distributions, the procedure is almost the same. The required packages are contained in one DVD.

```
mkdir /mnt1
mkdir /cm/images/rhel-image/root/repo1
mount -o loop,ro RHEL-DVD1.iso /mnt1
cp -ar /mnt1/* /cm/images/rhel-image/root/repo1/
```

The repository is added to YUM in the image, by creating the repository file `/cm/images/rhel-image/etc/yum.repos.d/rhel-base.repo` with the following contents:

```
[base]
name=Red Hat Enterprise Linux $releasever - $basearch - Base
baseurl=file:///root/repo1/Server
gpgcheck=0
enabled=1
```

Configuring Local Repositories For CentOS And DDN For A Software Image

Mounting the ISOs The variable `$imagedir` is assigned as a shortcut for the software image that is to be configured to use a local repository:

```
imagedir=/cm/images/default-image
```

If the ISO is called `bright-centos.iso`, then its filesystem can be mounted by the root user on a new mount, `/mnt1`, as follows:

```
mkdir /mnt1
mount -o loop bright-centos.iso /mnt1
```

The head node can then access the ISO filesystem.

The same mounted filesystem can also be mounted with the `bind` option into the software image. This can be done inside the software image by the root user, in the same relative position as for the head node, as follows:

```
mkdir $imagedir/mnt1
mount -o bind /mnt1 $imagedir/mnt1
```

This allows an operation run under the `$imagedir` in a chroot environment to access the ISO filesystem too.

Creating YUM repository configuration files: YUM repository configuration files can be created:

- **for the head node:** A repository configuration file

  ```
  /etc/yum.repos.d/cm6.1-dvd.repo
  ```

  can be created, for example, for a release tagged with a <*subminor*> number tag, with the content:

  ```
  [ddn-repo]
  name=DirectMon DVD Repo
  baseurl=file:///mnt1/data/cm-rpms/-<subminor>
  enabled=1
  gpgcheck=1
  exclude = slurm* pbspro* sge* torque* cm-hwloc
  ```

- **for the regular node image:** A repository configuration file

  ```
  $imagedir/etc/yum.repos.d/cm6.1-dvd.repo
  ```

  can be created. This file is in the image directory, but it has the same content as the previous head node yum repository configuration file.

Verifying that the repository files are set up right: To verify the repositories are usable on the head node, the YUM cache can be cleaned, and the available repositories listed:

```
[root@ddnmon61 ~]# yum clean all
[root@ddnmon61 ~]# yum repolist -v
bright-repo                 Bright Cluster Manager DVD Repo
...
```

To carry out the same verification on the image, these commands can be run with `yum --installroot=$imagedir` substituted in place of just `yum`.

The ISO repository should show up, along with any others that are accessible. Connection attempts that fail to reach a network-based or local repositories display errors. If those repositories are not needed, they can be disabled from within their configuration files.

10.6.4 Creating A Custom Image From The Local Repository

After having created the local repositories for SLES, RHEL or CentOS (section 10.6.3), a custom software image based on one of these can be created. For example, for CentOS, in a directory given the arbitrary name `offlineimage`:

```
cm-create-image -d $imagedir -n offlineimage -e -s
```

The `-e` option prevents copying the default cluster manager repository files on top of the image being created, since they may have been changed by the administrator from their default status. The `-s` option prevents installing additional base distribution packages that might not be required.

11 Cluster Monitoring

The DirectMon monitoring framework lets a cluster administrator:

- inspect monitoring data to the required level for existing resources;
- configure gathering of monitoring data for new resources;
- see current and past problems or abnormal behavior;
- notice trends that help the administrator predict likely future problems;
- handle current and likely future problems by
  - triggering alerts;
  - taking action if necessary to try to improve the situation or to investigate further.

Powerful features are accessible within an intuitive monitoring framework, and customized complex setups can be constructed to suit the requirements of the administrator.

In this chapter, the monitoring framework is explained with the following approach:

1. A basic example is first presented in which processes are run on a node. These processes are monitored, and are acted on when a threshold is exceeded.
2. With this easy-to-understand example as the base, the various features and associated functionality of the DirectMon monitoring framework are described and discussed in depth. These include visualization of data, concepts, configuration, monitoring customization and `cmsh` use.

11.1 A Basic Example Of How Monitoring Works

In this section, a minimal basic example of monitoring a process is set up. The aim is to present a simple overview that covers a part of what the monitoring framework is capable of handling. The overview gives the reader a structure to keep in mind, around which further details are fitted and filled in during the coverage in the rest of this chapter.

In the example, a user runs a large number of pointless CPU-intensive processes on a head node which is normally very lightly loaded. An administrator would then want to monitor user mode CPU load usage, and stop such processes automatically when a high load is detected (figure 11.1).

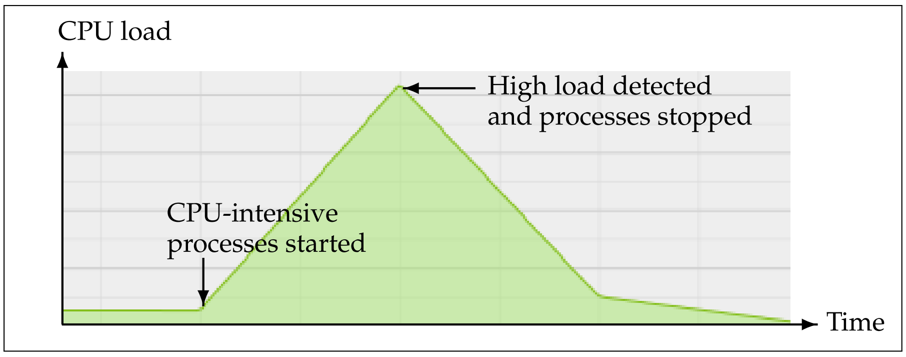


Figure 11.1: Monitoring Basic Example: CPU-intensive Processes Started, Detected And Stopped

The basic example illustrates a (very contrived) way for the DirectMon monitoring framework to be used to do that.

11.1.1 Before Using The Framework—Setting Up The Pieces

Running A Large Number Of Pointless CPU-Intensive Processes

One way to simulate a user running pointless CPU-intensive processes is to run several instances of the standard Unix utility, `yes`. The `yes` command sends out an endless number of lines of "y" texts. It is usually used to answer prompts for confirmation.

8 subshell processes are run in the background from the command line on the head node, with `yes` output sent to `/dev/null` as follows:

```
for i in {1..8}; do ( yes > /dev/null &); done
```

Running "`mpstat 2`" shows usage statistics for each processor, updating every 2 seconds. It shows that `%user`, which is user mode CPU usage, and which is reported as `CPUUser` in the DirectMon metrics, is close to 100% on an 8-core or less head node when the 8 subshell processes are running.

Setting Up The Kill Action

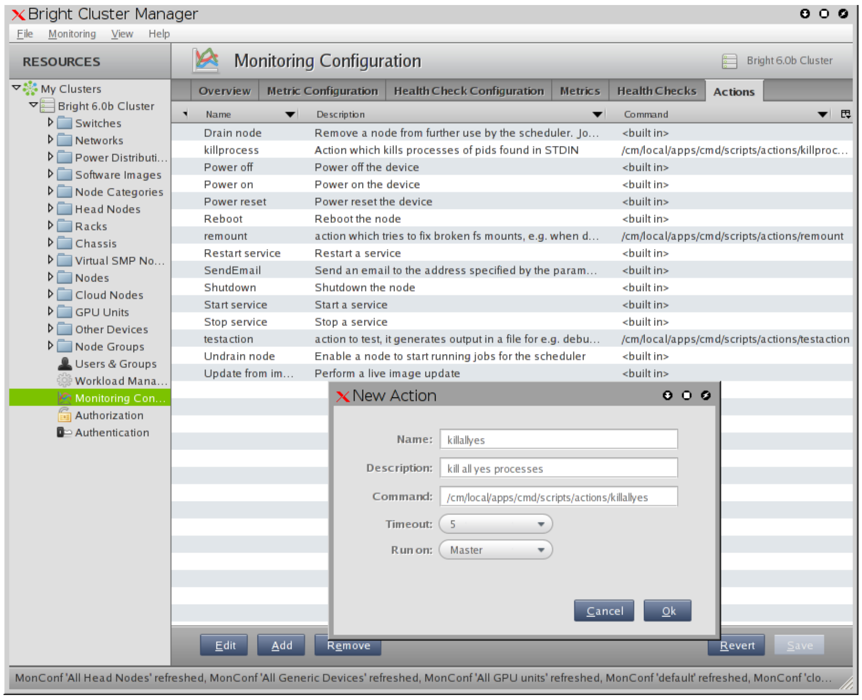
To stop the pointless CPU-intensive `yes` processes, the command "killall yes" is used. It is made a part of a script `killallyes`:

```
#!/bin/bash
killall yes
```

and made executable with a `chmod 700 killallyes`. For convenience, it may be placed in the `/cm/local/apps/cmd/scripts/actions` directory where other action scripts also reside.

11.1.2 Using The Framework

Now that the pieces are in place, `cmgui`'s monitoring framework is used to add the action to its action list, and then set up a threshold level that triggers the action:



Name	Description	Command
Drain node	Remove a node from further use by the scheduler. Jo...	<built in>
killprocess	Action which kills processes of pids found in STDIN	/cm/local/apps/cmd/scripts/actions/killproc...
Power off	Power off the device	<built in>
Power on	Power on the device	<built in>
Power reset	Power reset the device	<built in>
Reboot	Reboot the node	<built in>
remount	action which tries to fix broken fs mounts, e.g. when d...	/cm/local/apps/cmd/scripts/actions/remount
Restart service	Restart a service	<built in>
SendEmail	Send an email to the address specified by the param...	<built in>
Shutdown	Shutdown the node	<built in>
Start service	Start a service	<built in>
Stop service	Stop a service	<built in>
testaction	action to test, it generates output in a file for e.g. debu...	/cm/local/apps/cmd/scripts/actions/testaction
Undrain node	Enable a node to start running jobs for the scheduler	<built in>
Update from im...	Perform a live image update	<built in>

New Action

Name: killallyes
Description: kill all yes processes
Command: /cm/local/apps/cmd/scripts/actions/killallyes
Timeout: 5
Run on: Master

MonConf 'All Head Nodes' refreshed, MonConf 'All Generic Devices' refreshed, MonConf 'All GPU units' refreshed, MonConf 'default' refreshed, MonConf 'clo...

Figure 11.2: `cmgui` Monitoring Configuration: Adding An Action

Adding The Action To The Actions List

From the resources tree of `cmgui`, `Monitoring Configuration` is selected, and then the `Actions` tab is selected. A list of currently available actions is displayed. A new action is added by entering the following values in the `Add` dialog (figure 11.2):

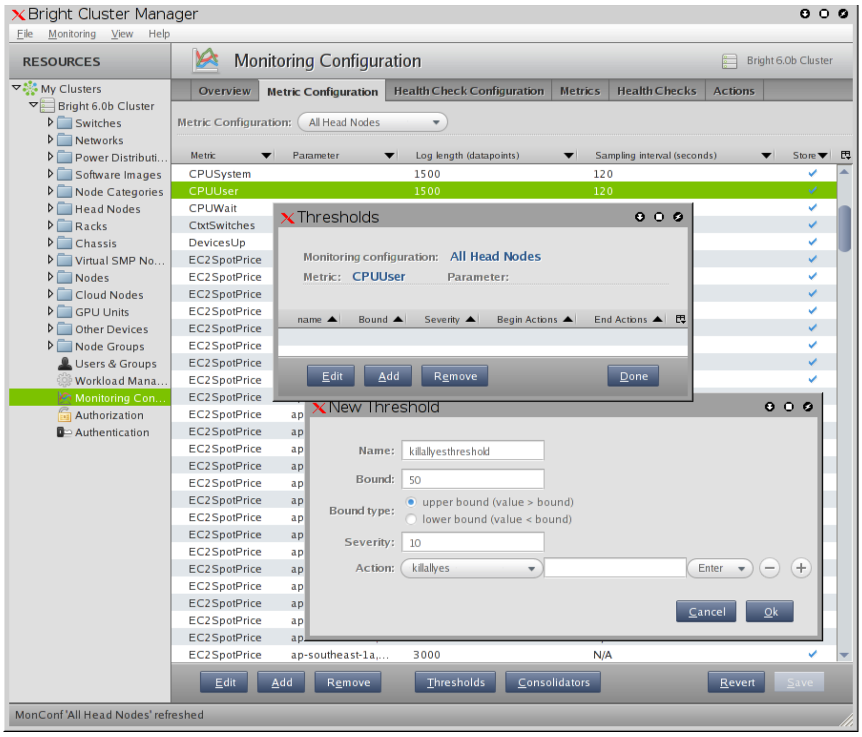
- action name: `killallyes`
- description: `kill all yes processes`
- command: `/cm/local/apps/cmd/scripts/actions/killallyes`

The `Save` button adds the action `killallyes` to the list of possible actions, which means that the action can now be used throughout the monitoring framework.

Setting Up The Threshold Level For CPUUser On The Head Node(s)

Continuing on, the `Metric Configuration` tab is selected. Then within the selection box options for `Metric Configuration`, `All Head Nodes` is selected to confine the metrics being measured to the head node(s). The metric `CPUUser`, which is a measure of the user mode CPU usage as a percentage, is selected. The `Thresholds` button is clicked on to open a `Thresholds` dialog. Within the `Thresholds` dialog the `Add` button is clicked to open up a "`New Threshold`" dialog. Within the "`New Threshold`" dialog (figure 11.3), these values are set:

- threshold name: `killallyesthreshold`
- (upper) bound: `50`
- action name (first selection box in the action option): `killallyes`
- action state option (next selection box in the action option): `Enter`



Metric Configuration: All Head Nodes

Metric	Parameter	Log length (datapoints)	Sampling interval (seconds)	Store
CPUSystem		1500	120	✓
CPUUser		1500	120	✓
CPUWait				✓
CtxtSwitches				✓
DevicesUp				✓
EC2SpotPrice				✓
EC2SpotPrice				✓
EC2SpotPrice				✓
EC2SpotPrice				✓
EC2SpotPrice				✓
EC2SpotPrice				✓
EC2SpotPrice				✓
EC2SpotPrice				✓
EC2SpotPrice				
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap			
EC2SpotPrice	ap-southeast-1a,...	3000	N/A	✓

Thresholds

Monitoring configuration: All Head Nodes
Metric: CPUUser Parameter:

New Threshold

Name: killallyesthreshold
Bound: 50
Bound type: [✓ upper bound (value > bound)] lower bound (value < bound)
Severity: 10
Action: killallyes Enter

MonConf 'All Head Nodes' refreshed

Figure 11.3: cmgui Monitoring Configuration: Setting A Threshold

Clicking on Ok exits the "New Threshold" dialog, clicking on Done exits the Thresholds dialog, and clicking on Save saves the threshold setting associated with CPUUser on the head node.

The Result

In the preceding section, an action was added, and a threshold was set up with the monitoring framework.

With a default installation on a newly installed cluster, the measurement of CPUUser is done every 120s (the edit dialog of figure 11.23 shows how to edit this value). The basic example configured with the defaults thus monitors if CPUUser on the head node has crossed the bound of 50% every 120s.

If CPUUser is found to have entered, that is crossed over from below the value and gone into the zone beyond 50%, then the framework runs the killallyes script, killing all running yes processes. Assuming the system is trivially loaded apart from these yes processes, the CPUUser metric value then drops to below 50%.

After an Enter threshold condition has been met for a sample, the first sample immediately after that does not ever meet the Enter threshold condition, because an Enter threshold crossing condition requires the previous sample to be below the threshold.

The second sample can only launch an action if the Enter threshold condition is met and if the preceding sample is below the threshold.

Other non-yes CPU-intensive processes running on the head node can also trigger the killallyes script. Since the script only kills yes processes, leaving any non-yes processes alone, it would in such a case run unnecessarily. This is a deficiency due to the contrived and simple nature of the basic example being illustrated here, and is of no real concern.

11.2 Monitoring Concepts And Definitions

A discussion of the concepts of monitoring, along with definitions of terms used, is appropriate at this point. The features of the monitoring framework covered later on in this chapter will then be understood more clearly.

11.2.1 Metric

In the basic example of section 11.1, the metric value considered was `CPUUser`, measured at regular time intervals of 120s.

A metric is a property of a device that can be monitored. It has a numeric value and can have units, unless it is unknown, i.e. has a null value. Examples are:

- temperature (value in degrees Celsius, for example: 45.2 °C);
- load average (value is a number, for example: 1.23);
- free space (value in bytes, for example: 12322343).

A metric can be a built-in, which means it is an integral part of the monitoring framework, or it can be a standalone script.

The word metric is often used to mean the script or object associated with a metric as well as a metric value. The context makes it clear which is meant.

11.2.2 Action

In the basic example of section 11.1, the action script is the script added to the monitoring system to kill all yes processes. The script runs when the condition is met that `CPUUser` crosses 50%.

An *action* is a standalone script or a built-in command that is executed when a condition is met. This condition can be:

- health checking (section 11.2.4);
- threshold checking (section 11.2.3) associated with a metric (section 11.2.1);
- state flapping (section 11.2.9).

11.2.3 Threshold

In the basic example of section 11.1, a threshold is set to 50% of `CPUUser`, and an action set so that crossing this threshold runs the `killallyes` script.

A *threshold* is a particular value in a sampled metric. A sample can cross the threshold, thereby entering or leaving a zone that is demarcated by the threshold.

A threshold can be configured to launch an action (section 11.2.2) according to threshold crossing conditions. The "`New Threshold`" dialog of `cmgui` (figure 11.3) has three action launch configuration options:

1. `Enter`: if the sample has entered into the zone and the previous sample was not in the zone

2. `Leave`: if the sample has left the zone and the previous sample was in the zone
3. `During`: if the sample is in the zone, and the previous sample was also in the zone.

A threshold zone also has a settable severity (section 11.2.6) associated with it. This value is processed for the AlertLevel metric (section 11.2.7) when an action is triggered by a threshold event.

11.2.4 Health Check

A *health check* value is a state. It is the response to running a health check script at a regular time interval, with as possible response values: `PASS`, `FAIL`, or `UNKNOWN`. The state is recorded in the CMDaemon database, and in `cmgui` an overview can be seen in the `Overview` tab for the device, in the `Health Status` section.

Examples of health checks are:

- checking if the hard drive still has enough space left on it and returning `PASS` if it has;
- checking if an NFS mount is accessible, and returning `FAIL` if it is not;
- checking if `CPUUser` is below 50%, and returning `PASS` if it is;
- checking if the `cmsh` binary is found, and returning `UNKNOWN` if it is not.

A health check has a settable severity (section 11.2.6) associated with a `FAIL` or `UNKNOWN` response. The severity can be set by appending a value to the response. For example, `FAIL 30` or `UNKNOWN 10`, as is done in the `hpraid` health check (`/cm/local/apps/cmd/scripts/healthchecks/hpraid`).

Severity values are processed for the AlertLevel metric (section 11.2.7) when the health check runs.

A health check can also launch an action based on any of the response values, similar to the way that an action is launched by a metric with a threshold condition.

11.2.5 Conceptual Overview: Health Checks Vs Threshold Checks

A health check is quite similar to a threshold state check with a metric. Conceptually, however, they are intended to differ as follows:

- A threshold state check works with numeric values.

  A health check on the other hand works with a response state of `PASS`, `FAIL`, or `UNKNOWN`.
- Threshold-checking does not specifically store a direct history of whether the threshold condition was met or not—it just calls the action script right away as its response. Admittedly, the associated metric data values are still kept by the monitoring framework, so that establishing if a threshold has been crossed historically is always possible with a little effort.

A health check on the other hand stores its PASS/FAIL/UNKNOWN responses for the monitoring framework, making it easily accessible for viewing by default.

- The threshold-checking mechanism is intended to be limited to doing a numerical comparison of a metric value with a threshold value

  A health check on the other hand has more general checking capabilities.

With some inventiveness, a health check can be made to do the function of a metric's threshold/action sequence (as well as the other way round).

The considerations above should help decide what the appropriate tool (health check or metric threshold check) should be for the job.

11.2.6 Severity

Severity is a positive integer value that the administrator assigns to a threshold-crossing event or to a health check status event. It takes one of these 5 suggested values:

Value	Name	Icon	Description
0	info	ℹ	informational message
10	notice	ℹ	normal, but significant, condition
20	warning	⚠	warning conditions
30	error	⚠	error conditions
40	alert	❗	action must be taken immediately

By default the value is 10. Severity levels are used in the AlertLevel metric (section 11.2.7). They can also be set by the administrator in the return values of health check scripts (section 11.2.4) and during event bucket processing (section 11.6.2).

11.2.7 AlertLevel

AlertLevel is a special metric. It is not sampled, but it is re-calculated when an event with an associated Severity (section 11.2.6) occurs. There are two types of AlertLevel metrics:

1. *AlertLevel (max)*, which is simply the maximum severity of the latest value of all the events. The aim of this metric is to alert the administrator to the severity of the *most important* issue.

2. *AlertLevel (sum)* which is the *sum* of the latest severity values of all the events. The aim of this metric is to alert the administrator to the *overall severity* of issues.

11.2.8 InfoMessages

InfoMessages are optional messages that inform the administrator of the reason for a health status change, or give a reason for a particular metric value in the cluster. These show up in the Overview tab of nodes, in the Health Status section for metrics and for health checks.

Metric or health check scripts can use file descriptor 3 within their scripts to write an InfoMessage:

Example

```
echo "Drive speed unknown: Reverse polarity" >&3
```

11.2.9 Flapping

Flapping, or *State Flapping*, is when a state transition (section 11.2.10) occurs too many times over a number of samples. In the basic example of section 11.1, if the `CPUUser` metric crossed the threshold zone 7 times within 12 samples (the default values for flap detection), then it would by default be detected as flapping. A flapping alert would then be recorded in the event viewer, and a flapping action could also be launched if configured to do so. Flapping configuration for `cmgui` is covered for thresholds crossing events in section 11.4.2, when the metric configuration tab's `Edit` and `Add` dialogs are explained; and also covered for health check state changes in section 11.4.3, when the health check configuration tab's `Edit` and `Add` dialogs are explained.

11.2.10 Transition

A state transition is:

- a health check state change (for example, changing from `PASS` to `FAIL`, or from `FAIL` to `UNKNOWN`);
- a metric threshold (section 11.2.3) crossing event. This is only valid for values that `Enter` or `Leave` the threshold zone.

11.2.11 Conceptual Overview: `cmgui`'s Main Monitoring Interfaces

Monitoring information is presented in several places in `cmgui` for convenience during everyday use. The conceptual overview in figure 11.4 covers a commonly seen layout in `cmgui`, showing 4 monitoring-related viewing areas for the cluster administrator. These are:

1. **Visualization**

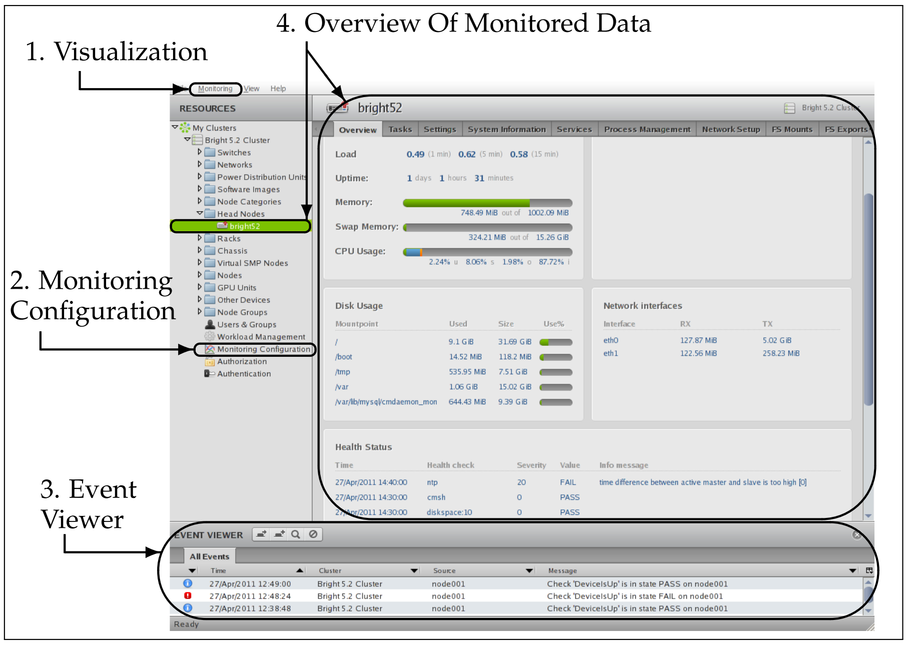
   Visualization of monitoring data is made available from `cmgui`'s monitoring menu, and launches a new window. Graphs are generated from metrics and health checks data, and these graphs are viewed in various ways within window panes.

   The use of the visualization tool is covered in section 11.3 using typical data from `CPUUser` from the basic example of section 11.1.

2. **Monitoring Configuration**

   Selecting the `Monitoring Configuration` resource in `cmgui` from the `Resources` list on the left hand side of the DirectMon displays the monitoring configuration pane on the right hand side. Within this pane, the following tabs show up:

   - `Overview`: an overview of enabled actions
   - `Metric Configuration`: allows configuration of device categories with metrics



Load 0.49 (1 min) 0.62 (5 min) 0.58 (15 min)

Uptime: 1 days 1 hours 31 minutes

Memory: 748.49 MiB out of 1002.09 MiB

Swap Memory: 324.21 MiB out of 15.26 GiB

CPU Usage: 2.24% u 8.06% s 1.98% o 87.72% i

Disk Usage

Mountpoint	Used	Size	Use%
/	9.1 GiB	31.69 GiB	
/boot	14.52 MiB	118.2 MiB	
/tmp	535.95 MiB	7.51 GiB	
/var	1.06 GiB	15.02 GiB	
/var/lib/mysql/cmdaemon_mon	644.43 MiB	9.39 GiB	

Network interfaces

Interface	RX	TX
eth0	127.87 MiB	5.02 GiB
eth1	122.56 MiB	258.23 MiB

Health Status

Time	Health check	Severity	Value	Info message
27/Apr/2011 14:40:00	ntp	20	FAIL	time difference between active master and slave is too high [0]
27/Apr/2011 14:30:00	cmsh	0	PASS	
27/Apr/2011 14:30:00	diskspace:10	0	PASS	

	Time	Cluster	Source	Message
●	27/Apr/2011 12:49:00	Bright 5.2 Cluster	node001	Check 'DeviceIsUp' is in state PASS on node001
🟥	27/Apr/2011 12:48:24	Bright 5.2 Cluster	node001	Check 'DeviceIsUp' is in state FAIL on node001
●	27/Apr/2011 12:38:48	Bright 5.2 Cluster	node001	Check 'DeviceIsUp' is in state PASS on node001

Figure 11.4: `cmgui` Conceptual Overview - Monitoring Types

- `Health Check Configuration`: allows configuration of device categories with health checks
- `Metrics`: allows configuration of metrics for devices
- `Health Checks`: allows configuration of health checks for devices
- `Actions`: allows actions to be set to run from metric thresholds and health check results

Some parts of `Monitoring Configuration` were used in the basic example of section 11.1 to set up the threshold for `CPUUser`, and to assign the action. It is covered more thoroughly in section 11.4.

3. **Event Viewer**

   The *Event Viewer* displays events that are seen on the cluster(s) within a pane of `cmgui`. How the events are presented is configurable, with tools that allow filtering based on dates, clusters, nodes or a text string; and widgets that allow rearranging the sort order or detaching the pane.

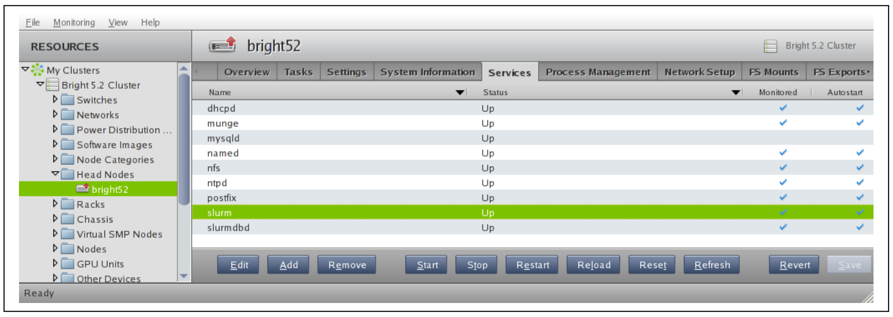
4. **Overview Of Monitored Data**

   A dashboard in a car conveys the most important relevant information at a glance and attracts attention to items that are abnormal and merit further investigation.

   The same idea lies behind the `Overview` tab of DirectMon. This gives a dashboard view based on the monitored data for a particular device such as a switch, a cluster (probably the most useful overview, and therefore also the default when first connecting to the cluster with `cmgui`), a node, a GPU unit, and so on.

   Neighboring tabs often allow a closer look at issues noticed in the `Overview`, and also sometimes a way to act on them.

For example, if jobs are not seen in the `Overview` tab, then the administrator may want to look at the neighboring `Services` tab (figure 11.5), and see if the workload manager is running. The `Services` tab (section 4.12.2) allows the administrator to manage a service such as the workload manager.



Name	Status	Monitored	Autostart
dhcpd	Up	✓	✓
munge	Up	✓	✓
mysqld	Up		
named	Up	✓	✓
nfs	Up	✓	✓
ntpd	Up	✓	✓
postfix	Up	✓	✓
slurm	Up	✓	✓
slurmdbd	Up	✓	✓

Figure 11.5: `cmgui`: Device Services Tab

11.3 Monitoring Visualization With `cmgui`

The `Monitoring` option in the menu bar of `cmgui` (item 1 in figure 11.4) launches an intuitive visualization tool that should be the main tool for getting a feel of the system's behavior over periods of time. With this tool the measurements and states of the system are viewed. Graphs for metrics and health checks can be looked at in various ways: for example, the graphs can be zoomed in and out on over a particular time period, the graphs can be laid out on top of each other or the graphs can be laid out as a giant grid. The graph scale settings can also be adjusted, stored and recalled for use the next time a session is started.

An alternative to `cmgui`'s visualization tool is the command-line `cmsh`. This has the same functionality in the sense that data values can be selected and studied according to configurable parameters with it (section 11.8). The data values can even be plotted and displayed on graphs with `cmsh` with the help of Unix pipes and graphing utilities. However, the strengths of monitoring with `cmsh` lie elsewhere: `cmsh` is more useful for scripting or for examining pre-decided metrics and health checks rather than a quick visual check over the system. This is because `cmsh` needs more familiarity with options, and is designed for text output instead of interactive graphs. Monitoring with `cmsh` is discussed in sections 11.7 and 11.8.

How `cmgui` is used for visualization is now described.

11.3.1 The Monitoring Window

The `Monitoring` menu is selected from the menu bar of `cmgui` and a cluster name is selected.

The `Monitoring` window opens (figure 11.6). The resources in the cluster are shown on the left side of the window. Clicking on a resource opens or closes its subtree of metrics and health checks.

The subsequent sections describe ways of viewing and changing resource settings. After having carried out such modifications, saving and loading a settings state can be done from options in the `File` menu.

Figure 11.6 shows the different resources of the head node, with the

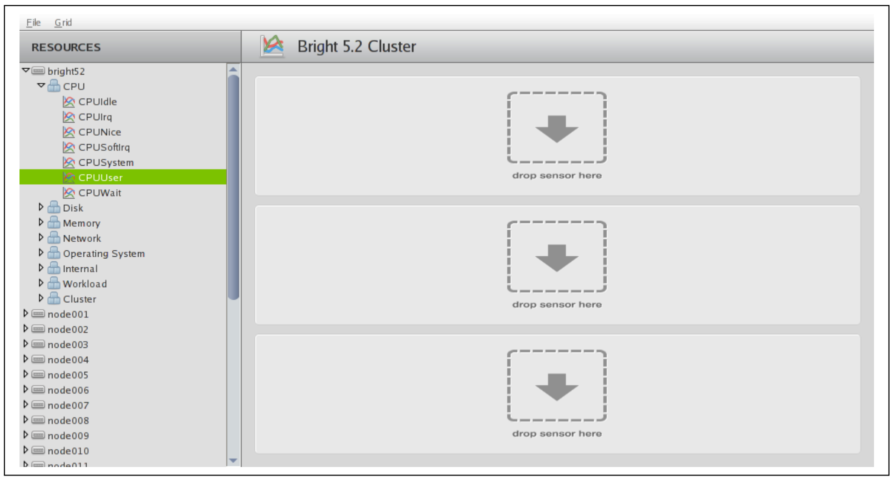


Figure 11.6: `cmgui` Monitoring Window: Resources View

CPU resource subtree opened up in turn to show its metrics and health checks. Out of these, the `CPUUser` metric (for user CPU usage) is shown selected for further display.

To display this metric, the selection is drag-and-dropped onto one of the 3 panes which has the text "`drop sensor here`".

11.3.2 The Graph Display Pane

Figure 11.7 shows the monitoring window after such a drag-and-drop. The graph of the metric `CPUUser` is displayed over 20 minutes (10th November 2010 08:04 to 08:24). On the y-axis the unit used by the metric is shown (0% to about 100%). This example is actually of data gathered when the basic example of 11.1 was run, and shows `CPUUser` rising as a number of `yes` processes are run, and falling when they end.

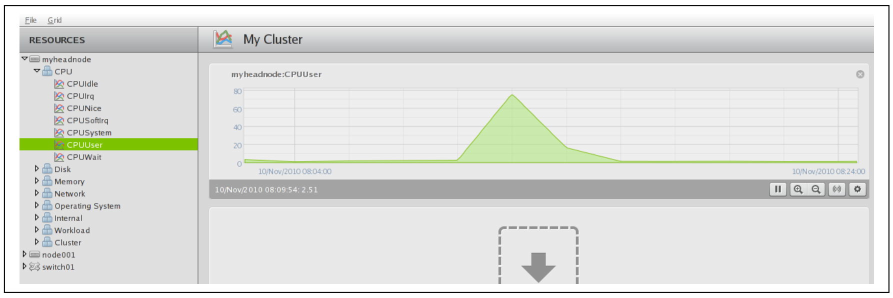


Figure 11.7: `cmgui` Monitoring Window: Graph Display Pane

Features of graph display panes are (figure 11.8):

1. **The close widget** which erases all graphs on the drawing pane when it is clicked. (Individual graphs are removed in the settings dialog discussed in section 11.3.5.)

2. **The (time, measurement) data values** in the graph are displayed on the graph toolbar by hovering the mouse cursor over the graph.

3. **The graph view adjustment buttons** are:

   - **play/pause**: by default the graph is refreshed with new data

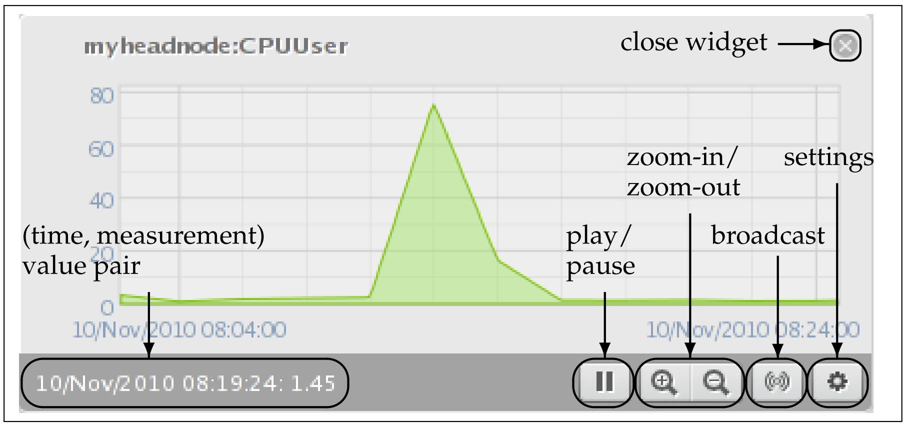


Figure 11.8: Graph Display Pane: Features

every 2 minutes. This is disabled and resumed by clicking on the pause/play button on the graph toolbar.

- **zoom-out/zoom-in**: Clicking on one of the magnifying glasses zooms-in or zooms-out on the graph in time. This way data values can be shown, even from many months ago. Zooming in with mouse gestures is also possible and is discussed in section 11.3.4.
- **broadcast**: A time-scale synchronizer. Toggling this button to a pressed state for one of the graphs means that scale changes carried out via magnifying glass zooms (preceding bullet point) or via mouse gestures (section 11.3.4) are done over all the other graph display panes too so that their x-ranges match. This is useful for large numbers of nodes.
- **settings**: Clicking on this button opens a dialog window to modify certain aspects of the graph. The settings dialog is discussed in section 11.3.5.

4. **A grid of graph display panes** can be laid out by using the `Grid` menu option of the main `Monitoring Pane` (figure 11.6). Among the menu options of the `Grid` menu (figure 11.9) are:

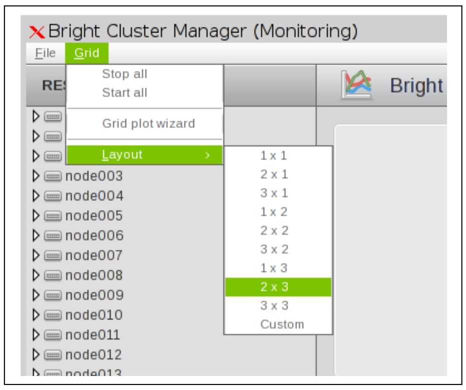


Figure 11.9: Grid Menu Options

(a) **Layout**: a grid of dimensions $x \times y$ can be selected or specified with the `Layout` option. With the `Layout` option, metrics need to be added manually to each grid unit.

   (b) **Grid plot wizard**: For larger grids it is tedious to allocate devices to a grid and manually fill in the grid units with metrics. The `Grid plot wizard` can take care of the tedious aspects, and is described in section 11.3.3.

5. **Multiple graphs** are drawn in a single graph display pane by repeating the drag and drop for different metrics. For example, adding the CPUIdle metric with a drag-and-drop to the `CPUUser` graph of figure 11.7 gives a result as seen in figure 11.10, where both graphs lie on the same axis in the top pane.

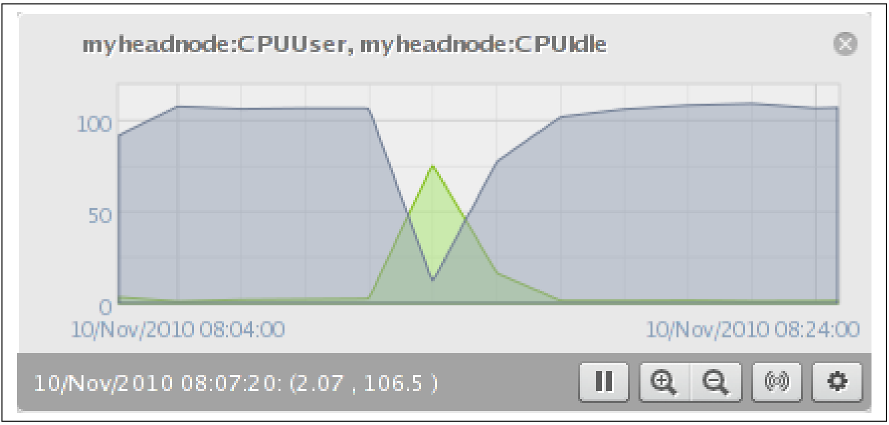

myheadnode:CPUUser, myheadnode:CPUIdle
100
50
0
10/Nov/2010 08:04:00
10/Nov/2010 08:24:00
10/Nov/2010 08:07:20: (2.07 , 106.5)

Figure 11.10: Graph Display Pane: Multiple Graphs On One Pane

6. **Boxplots** (sometimes called box-and-whisker plots) are used to draw graphs for samples carried out over the interval for a particular time on a resource item that has multiple devices in it. For example, in the resource item `node001`, `CPUUser` is sampled and plotted for a single node for set time intervals, and so it is displayed as a line plot. However, for the resource item "`All nodes`", `CPUUser` is sampled and plotted for all nodes for set time intervals, and so it is displayed as a boxplot (figure 11.11):

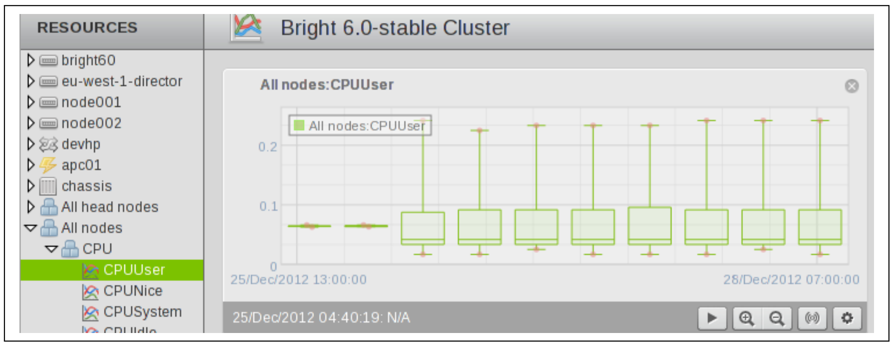


Figure 11.11: Graph Display Pane: Boxplots

Resource items that have multiple devices are indicated by starting with the text "`All`". For example: "`All nodes`", "`All head nodes`", and so on.

The boxplot displayed shows the lowest and highest values at the whisker ends. Typically, the lower and upper quartile values are the box borders, and the median value is the middle divider of the box. In addition, outlier points can be displayed. If there are only two outlier points, they are the same as the lowest and highest values. Outlier points and how to set them are described in section 11.3.5.

The `dumpstatistics` command in `cmsh` is somewhat similar in function to boxplots in `cmgui`

11.3.3 Using The Grid Wizard

Within the `Monitoring` window (section 11.3.1), the `Grid plot wizard` sets up a grid for devices selected by the administrator, and allows metrics to be added automatically to each grid unit.

The first screen of the wizard allows devices to be selected from the group of all devices, and placed in a group of devices that are to have their metrics plotted (figure 11.12).

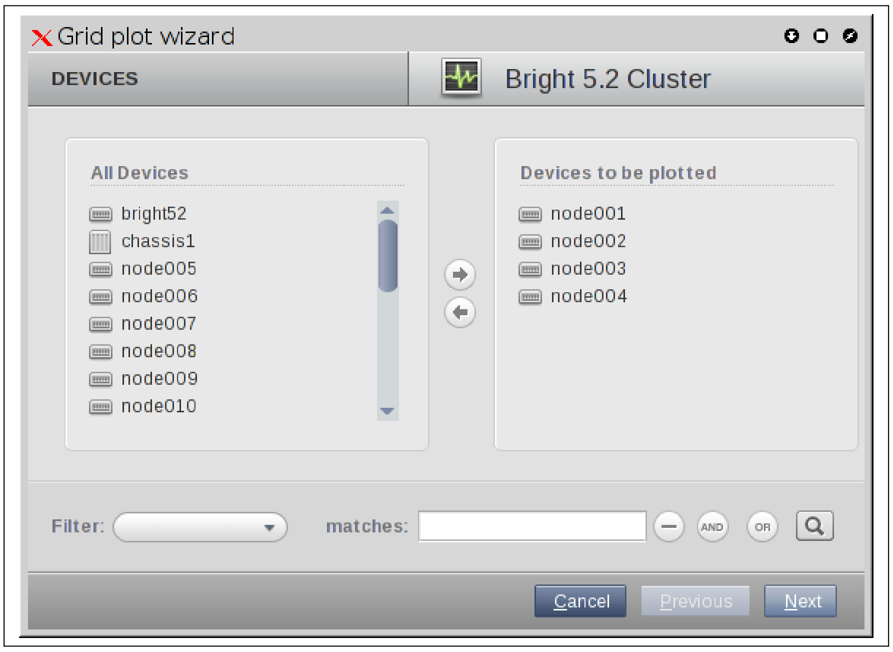


Figure 11.12: Grid Wizard: Devices Selection

The next screen of the wizard allows metrics to be drag-and-dropped from the available metrics into a group of metrics that are to be displayed for the devices in the previous screen (figure 11.13).

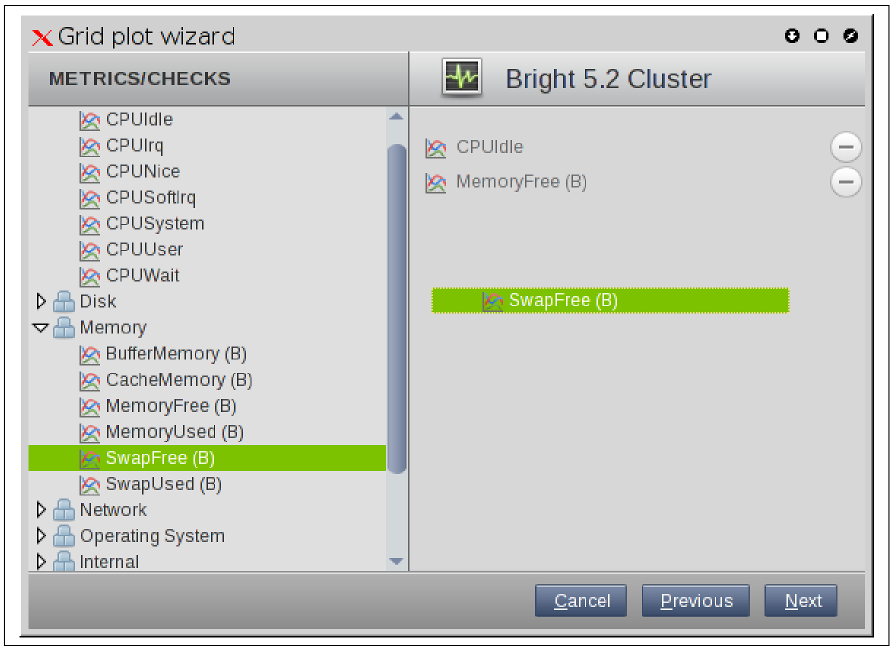
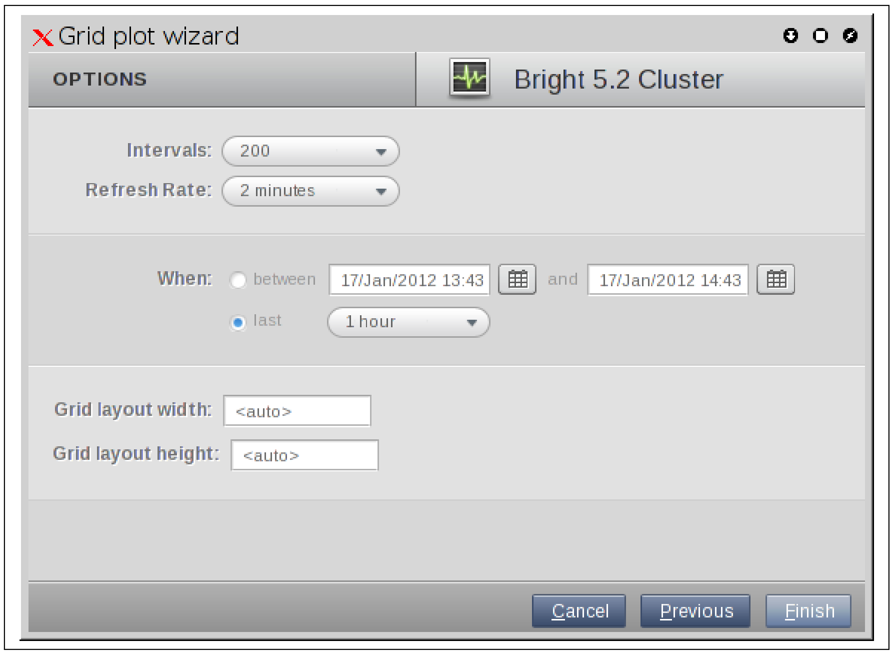


Figure 11.13: Grid Wizard: Metrics Drag-And-Drop

The last screen of the wizard allows several display options to be set for the selected devices and their metrics (figure 11.14).



Intervals: 200
Refresh Rate: 2 minutes
When: ○ between 17/Jan/2012 13:43 and 17/Jan/2012 14:43
● last 1 hour
Grid layout width: <auto>
Grid layout height: <auto>

Figure 11.14: Grid Wizard: Display Options

One of these options is the specification of the layout width and height for the displayed grid of selected devices. For example, four nodes could be laid out in a grid of 4 wide by 1 high, 2 wide by 2 high, or 1 wide by 4 high. The meanings of the remaining display options are described in section 11.3.5.

Once the `Finish` button of the last screen is clicked, a graph display pane is shown with a grid of graphs (figure 11.15).

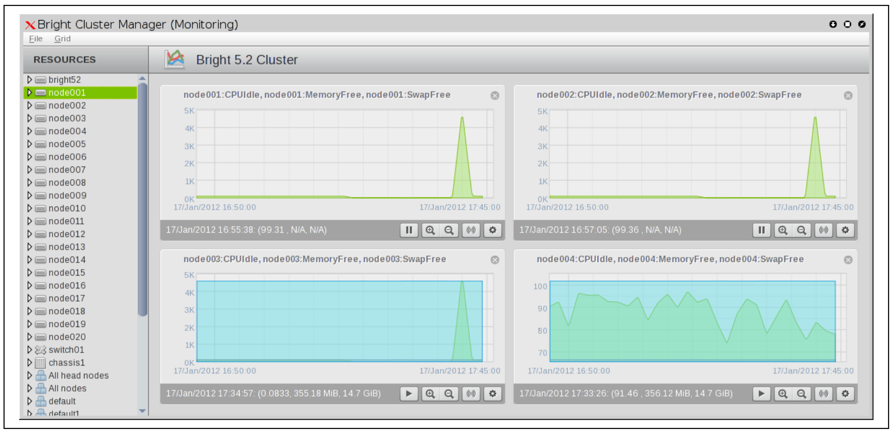

Bright Cluster Manager (Monitoring)
File Grid
RESOURCES
Bright 5.2 Cluster
node001:CPUIdle, node001:MemoryFree, node001:SwapFree
node002:CPUIdle, node002:MemoryFree, node002:SwapFree
node003:CPUIdle, node003:MemoryFree, node003:SwapFree
node004:CPUIdle, node004:MemoryFree, node004:SwapFree

Figure 11.15: Grid Wizard: Grid Result

11.3.4 Zooming In With Mouse Gestures

Besides using a magnifying glass button there are two other ways to zoom in on a graph, based on intuitive mouse gestures:

X-Axis Zoom

The first way to zoom in is to draw a horizontal line across the graph by holding the left mouse button down on the graph. A guide line shows up while doing this (figure 11.16):

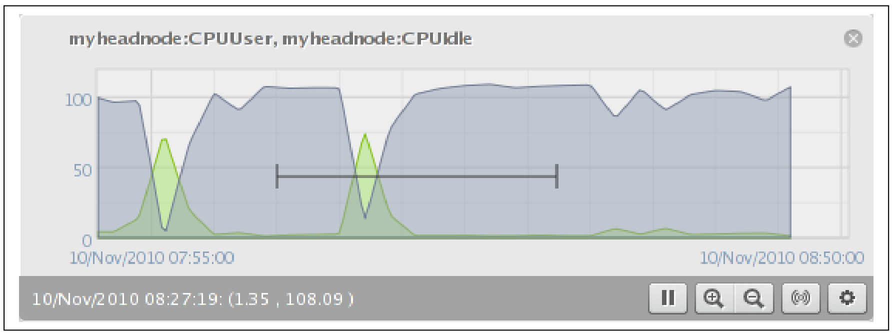

myheadnode:CPUUser, myheadnode:CPUIdle
100
50
0
10/Nov/2010 07:55:00
10/Nov/2010 08:50:00
10/Nov/2010 08:27:19: (1.35 , 108.09)

Figure 11.16: Graph Display Pane: X-axis Zoom Start

The x-axis range covered by this line is zoomed in on when the mouse button is released (figure 11.17):

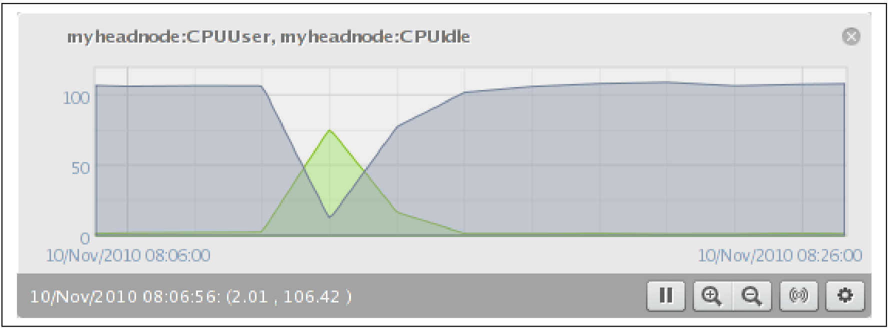

myheadnode:CPUUser, myheadnode:CPUIdle
100
50
0
10/Nov/2010 08:06:00
10/Nov/2010 08:26:00
10/Nov/2010 08:06:56: (2.01 , 106.42)

Figure 11.17: Graph Display Pane: X-axis Zoom Finish

Box Zoom

The second way to zoom in is to draw a box instead of a line across the graph by holding the left mouse button down and drawing a line diagonally across the data instead of horizontally. A guide box shows up (figure 11.18):

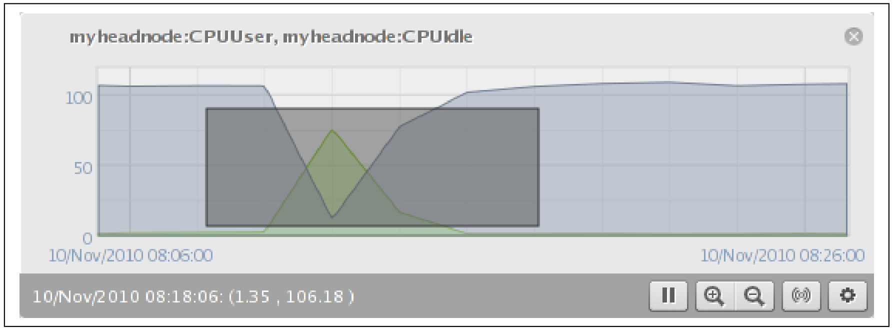

myheadnode:CPUUser, myheadnode:CPUIdle
100
50
0
10/Nov/2010 08:06:00
10/Nov/2010 08:26:00
10/Nov/2010 08:18:06: (1.35 , 106.18)

Figure 11.18: Graph Display Pane: Box Zoom Start

This is zoomed into when the mouse button is released (figure 11.19):

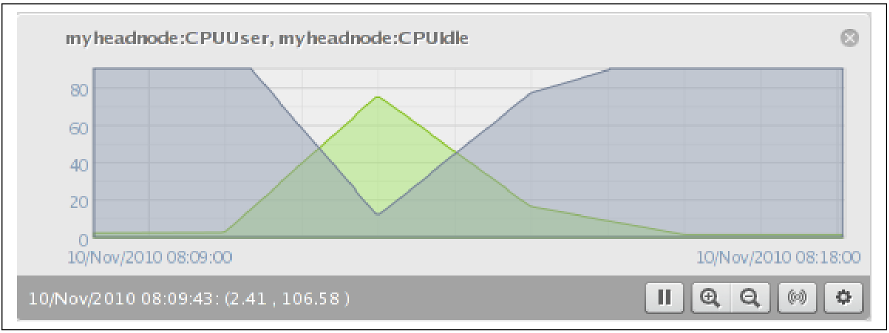
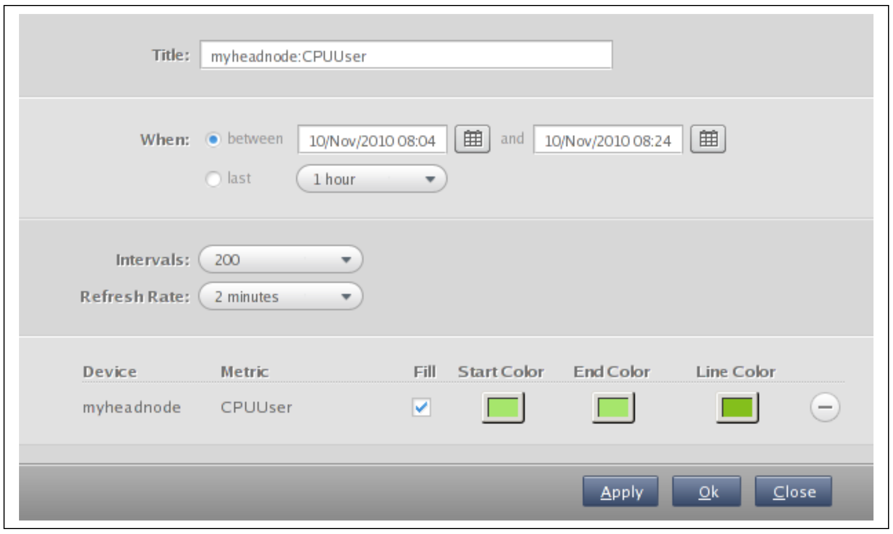

myheadnode:CPUUser, myheadnode:CPUIdle
80
60
40
20
0
10/Nov/2010 08:09:00
10/Nov/2010 08:18:00
10/Nov/2010 08:09:43: (2.41 , 106.58)

Figure 11.19: Graph Display Pane: Box Zoom Finish

11.3.5 The Graph Display Settings Dialog

Clicking on the settings button in the graph display pane (figure 11.8) opens up the graph display pane settings dialog (figure 11.20):



Title: myheadnode:CPUUser

When: [✓ between] 10/Nov/2010 08:04 and 10/Nov/2010 08:24

last 1 hour

Intervals: 200

Refresh Rate: 2 minutes

Device	Metric	Fill	Start Color	End Color	Line Color
myheadnode	CPUUser	✓	🟩	🟩	🟩

Figure 11.20: Graph Display Pane Settings Dialog

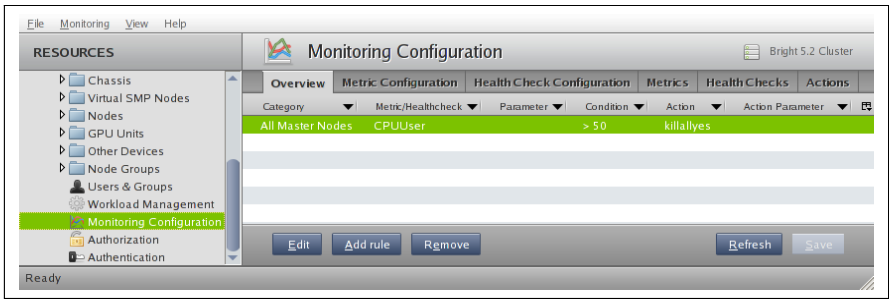
This allows the following settings to be modified:

- the `Title` shown at the top of the graph;
- over `When` the x-range is displayed;
- The existence and positioning of the `Legend`;
- the `Intervals` value. This is the number of intervals (by default 200) used to draw the graph. For example, although there may be 2000 data points available during the selected period, by default only 200 are used, with each of the 200 an average of 10 real data points. This mechanism is especially useful for smoothing out noisier metrics to give a better overview of behavior.
- The `Refresh Interval`, which sets how often the graph is recreated;
- the visual layout of the graphs, which can be adjusted so that:
  - `Color` aspects of each graph are changed in the row of settings for that graph;
  - Each graph is deleted from its pane with the ⊖ button at the end of the row of settings for that graph;
  - The number of points representing `Outliers` can be set for each boxplot (page 411) displayed. The points are chosen based on their distance being a maximum from the median, with the order in which they are chosen being alternately above and below the median.

11.4 Monitoring Configuration With `cmgui`

This section is about the configuration of monitoring for health checks and metrics, along with setting up the actions which are triggered from a health check or a metric threshold check.

Selecting `Monitoring Configuration` from the resources section of `cmgui` makes the following tabs available (figure 11.21):



Category	Metric/Healthcheck	Parameter	Condition	Action	Action Parameter
All Master Nodes	CPUUser		> 50	killallyes	

Figure 11.21: `cmgui` Monitoring Configuration Tabs

- `Overview` (displays as the default)
- `Metric Configuration`
- `Health Check Configuration`
- `Metrics`
- `Health Checks`
- `Actions`

The tabs are now discussed in detail.

11.4.1 The Overview Tab

The `Overview` tab of figure 11.21 shows an overview of custom threshold actions and custom health check actions that are active in the system. Each row of conditions in the list that decides if an action is launched is called a rule. Only one rule is on display in figure 11.21, showing an overview of the metric threshold action settings which were set up in the basic example of section 11.1.

The `Add rule` button runs a convenient wizard that guides an administrator in setting up a condition, and thereby avoids having to go through the other tabs separately.

The `Remove` button removes a selected rule.

The `Edit` button edits aspects of a selected rule. It opens a dialog that edits a metric threshold configuration or a health check configuration. These configuration dialog options are also accessible from within the `Metric Configuration` and `Health Check Configuration` tabs.

The `Revert` button reverts a modified state of the tab to the last saved state.

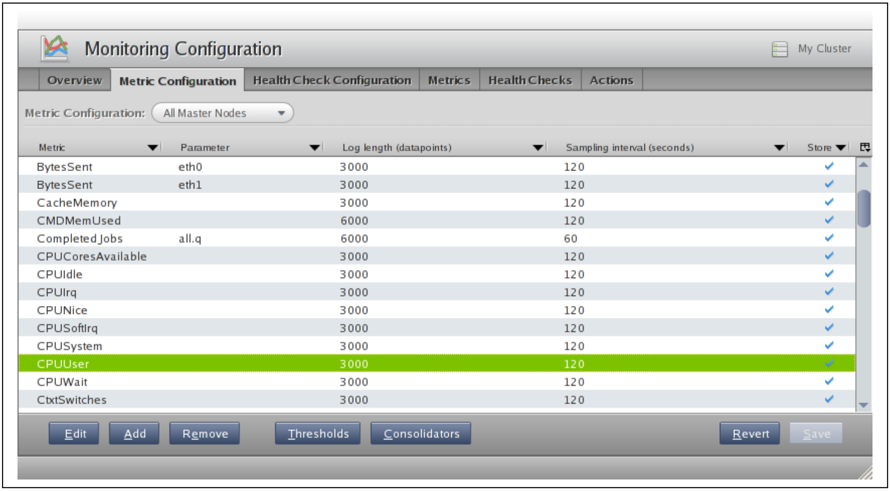
The `Save` button saves a modified state of the tab.

11.4.2 The Metric Configuration Tab

The `Metric Configuration` tab allows device categories to be selected for the sampling of metrics. Properties of metrics related to the taking of samples can then be configured from this tab for the selected device category. These properties are the configuration of the sampling parameters themselves (for example, frequency and length of logs), but also the configuration of related properties such as thresholds, consolidation, actions launched when a threshold is crossed, and actions launched when a metric state is flapping.

The `Metric Configuration` tab is initially a blank tab until the device category is selected by using the `Metric Configuration` selection box. The selection box selects the device category from a list of built-in categories and user-defined node categories (node categories are introduced in section 3.1.3). On selection, the metrics of the selected device category are listed in the `Metric Configuration` tab. Properties of the metrics related to sampling are only available for configuration and manipulation after the metrics list displays. Handling metrics in this manner, via groups of devices, is slightly awkward for just a few machines, but for larger clusters it keeps administration scalable and thus manageable.

Figure 11.22 shows an example of the `Metric Configuration` tab after `All master nodes` is chosen as the device category. This corresponds to the basic example of section 11.1, where `All master nodes` was the device category chosen because it was the `CPUUser` metric on a master node that was to be monitored. Examples of other device categories that could be chosen are `All ethernet switches`, if Ethernet switches are to have their metrics configured; or `All Power Distribution Units`, if power distribution units are to have their metrics configured.



Metric Configuration: All Master Nodes

Metric	Parameter	Log length (datapoints)	Sampling interval (seconds)	Store
BytesSent	eth0	3000	120	✓
BytesSent	eth1	3000	120	✓
CacheMemory		3000	120	✓
CMDMemUsed		6000	120	✓
CompletedJobs	all.q	6000	60	✓
CPUCoresAvailable		3000	120	✓
CPUIdle		3000	120	✓
CPUIrq		3000	120	✓
CPUNice		3000	120	✓
CPUSoftIrq		3000	120	✓
CPUSystem		3000	120	✓
CPUUser		3000	120	✓
CPUWait		3000	120	✓
CtxtSwitches		3000	120	✓

Figure 11.22: `cmgui` Monitoring: Metric Configuration Display After Category Selection

With the screen displaying a list of metrics as in figure 11.22, the metrics in the `Metric Configuration` tab can now be configured and manipulated. The buttons used to do this are: `Edit`, `Add`, `Remove`, `Thresholds`, `Consolidators`, `Revert` and `Save`.

The `Save` button saves as-yet-uncommitted changes made via the `Add` or `Edit` buttons.

The `Revert` button discards unsaved edits made via the `Edit` button. The reversion goes back to the last save.

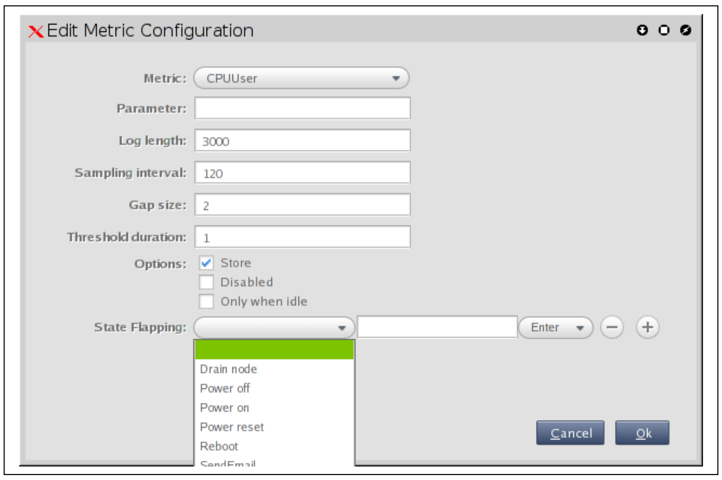
The `Remove` button removes a selected metric from the metrics listed.

The remaining buttons, `Edit`, `Add`, `Thresholds` and `Consolidators`, open up options dialogs. These options are now discussed.

Metric Configuration Tab: Edit And Add Options

The Metric Configuration tab of figure 11.22 has Add and Edit buttons. The Add button opens up a dialog to add a new metric to the list, and the Edit button opens up a dialog to edit a selected metric from the list. The dialogs allow logging options for a metric to be set or adjusted. For example, a new metric could be set for sampling by adding it to the device category from the available list of all metrics, or the sampling frequency could be changed on an existing metric, or an action could be set for a metric that has a tendency to flap.

The Edit and Add dialogs for a metric have the following options (figure 11.23):


Edit Metric Configuration

Metric:	CPUUser
Parameter:	
Log length:	3000
Sampling interval:	120
Gap size:	2
Threshold duration:	1
Options:	✓ Store
	Disabled
	Only when idle
State Flapping:	

Figure 11.23: cmgui Monitoring: Metric Configuration Tab Edit Dialog

- Metric: The name of the metric.
- Parameter: Values that the metric script is designed to handle. For example:
  - the metric FreeSpace tracks the free space left in a file system, and is given a mount point such as / or /var as a parameter;
  - the metric BytesRecv measures the number of bytes received on an interface, and takes an interface name such as eth0 or eth1 as a parameter.

  For CPUUser, the parameter field is disallowed in the Metric tab, so values here are ignored.
- Log length: The maximum number of raw data samples that are stored for the metric. 3000 by default.
- Sampling interval: The time between samples. 120s by default.
- Gap size: The number of missing samples allowed before a null value is stored as a sample value. 2 by default.
- Threshold duration: Number of samples in the threshold zone before a threshold event is decided to have occurred. 1 by default.
- Options checkboxes:

  - `Store`: If ticked, the metric data values are saved to the database. Note that any threshold checks are still done, whether the samples are stored or not.
  - `Disabled`: If ticked, the metric script does not run, and no threshold checks are done for it. If `Store` is also ticked, no value is stored.
  - `Only when idle`: If ticked, the metric script is only run when the system is idling. A resource-hungry metric burdens the system less this way.

- `State Flapping`: The first selection box decides what action to launch if state flapping is detected. The next box is a plain text-entry box that allows a parameter to be passed to the action. The third box is a selection box again, which decides when to launch the action, depending on which of these following states is set:
  - `Enter`: if the flapping has just started. That is, the current sample is in a flapping state, and the previous sample was not in a flapping state.
  - `During`: if the flapping is ongoing. That is, the current and previous flapping sample are both in a flapping state.
  - `Leave`: if the flapping has just stopped. That is, the current sample is not in a flapping state, and the previous sample was in a flapping state.

Metric Configuration Tab: Thresholds Options

The `Metric Configuration` tab of figure 11.22 also has a `Thresholds` button associated with a selected metric.

Thresholds are defined and their underlying concepts are discussed in section 11.2.3. The current section describes the configuration of thresholds.

In the basic example of section 11.1, `CPUUser` was configured so that if it crossed a threshold of 50%, it would run an action (the `killallyes` script). The threshold configuration was done using the `Thresholds` button of `cmgui`.

Clicking on the `Thresholds` button launches the `Thresholds` display window, which lists the thresholds set for that metric. Figure 11.24, which corresponds to the basic example of section 11.1, shows a `Thresholds` display window with a threshold named `killallyesthreshold` configured for the metric `CPUUser`.

The `Edit`, and `Remove` buttons in this display edit and remove a selected threshold from the list of thresholds, while the `Add` button adds a new threshold to the list.

The `Edit` and `Add` dialogs for a threshold prompt for the following values (figure 11.25):

- `Name`: the threshold's name.
- `Bound`: the metric value which demarcates the threshold.
- `Bound type`: If checked, the radio button for
  - `upper bound`: places the threshold zone above the bound;

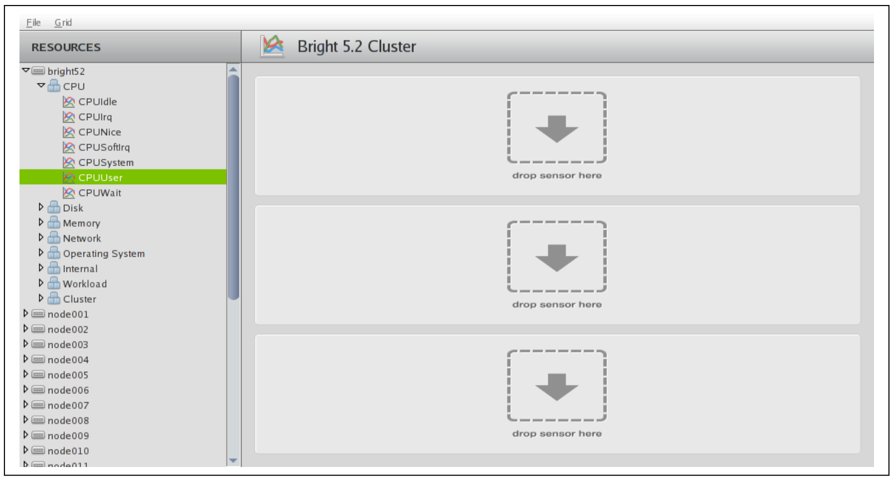
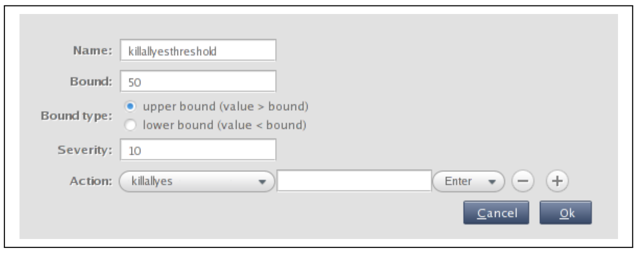


Figure 11.24: cmgui Monitoring: Thresholds Display



Name: killallyesthreshold

Bound: 50

Bound type: [✓ upper bound (value > bound)] lower bound (value < bound)

Severity: 10

Action: killallyes Enter

Figure 11.25: cmgui Metric Configuration: Thresholds Edit Dialog

  - lower bound: places the threshold zone below the bound.

- Severity: A value assigned to indicate the severity of the situation if the threshold is crossed. It is 10 by default. Severity is discussed in section 11.2.6.

- Action: The action field types decide how the action should be triggered and run. The field types are, from left to right:

  - script: a script selected from a drop-down list of available actions;

  - parameter: [optional] what parameter value to pass to the action;

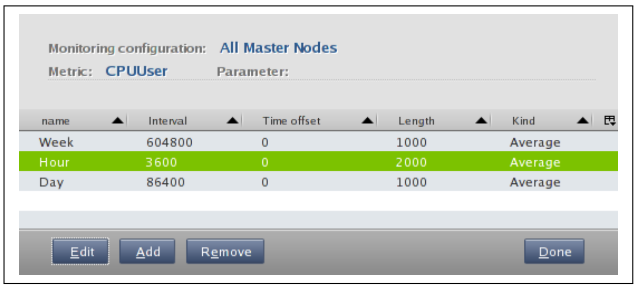
  - when: when the action is run. It is selected from a drop-down choice of Enter, During or Leave, where:

    * Enter runs the action if the sample has entered the zone;
    * Leave runs the action if the sample has left the zone;
    * During runs the action if the sample is in the zone, and the previous sample was also in the zone.

Metric Configuration Tab: Consolidators Options

The Metric Configuration tab of figure 11.22 also has a Consolidators button associated with the selected metric.

Consolidators decide how the data values are handled once the initial log length quantity for a metric is exceeded. Data points that have become old are gathered and, when enough have been gathered, they


Monitoring configuration: All Master Nodes
Metric: CPUUser Parameter:

name	Interval	Time offset	Length	Kind
Week	604800	0	1000	Average
Hour	3600	0	2000	Average
Day	86400	0	1000	Average

Figure 11.26: cmgui Metric Configuration: Consolidators Display

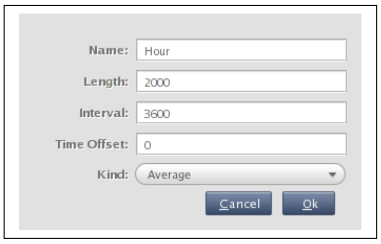

Name:	Hour
Length:	2000
Interval:	3600
Time Offset:	0
Kind:	Average

Figure 11.27: cmgui Metric Configuration: Consolidators Edit Dialog

are processed into consolidated data. Consolidated data values present fewer data values than the original raw data values over the same time duration. The aim of consolidation is to increase performance, save space, and keep the basic information still useful when viewing historical data.

The Consolidators button opens a window that displays a list of consolidators that have been defined for the selected metric (figure 11.26).

The Edit and Remove buttons in this display edit and remove a selected consolidator from the list of consolidators while the Add button in this display adds a new consolidator to the list of consolidators.

The Edit and Add dialogs for a consolidator prompt for the following values (figure 11.27):

- Name: the consolidator's name. By default Day, Hour, and Week are already set up, with appropriate values for their corresponding fields.
- Length: the number of intervals that are logged for this consolidator. Not to be confused with the metric log length.
- Interval: the time period (in seconds) associated with the consolidator. Not to be confused with the metric interval time period. For example, the default consolidator with the name Hour has a value of 3600.
- Time Offset: The time offset from the default consolidation time.

  To understand what this means, consider the Log length of the metric, which is the maximum number of raw data points that the metric stores. When this maximum is reached, the oldest data point is removed from the metric data when a new data point is added. Each removed data point is gathered and used for data consolidation purposes.

For a metric that adds a new data point every Sampling interval seconds, the time $\texttt{t}_{\texttt{raw gone}}$, which is how many seconds into the past the raw log data point is removed, is given by:

$$\texttt{t}_{\texttt{raw gone}} = (\texttt{Log length})_{\texttt{metric}} \times (\texttt{Sampling interval})_{\texttt{metric}}$$

This value is also the default consolidation time, because the consolidated data values are normally presented from $\texttt{t}_{\texttt{raw gone}}$ seconds ago, to further into the past. The default consolidation time occurs when the Time Offset has its default, zero value.

If however the Time Offset period is non-zero, then the consolidation time is offset, because the time into the past from which consolidation is presented to the user, $\texttt{t}_{\texttt{consolidation}}$, is then given by:

$$\texttt{t}_{\texttt{consolidation}} = \texttt{t}_{\texttt{raw gone}} + \texttt{Time Offset}$$

The monitoring visualization graphs then show consolidated data from $\texttt{t}_{\texttt{consolidation}}$ seconds into the past, to further into the past[1].

- Kind: the kind of consolidation done on the raw data samples. The output result for a processed set of raw data—the consolidated data point—is an average, a maximum or a minimum of the input raw data values. Kind can thus have the value Average, Maximum, or Minimum.

11.4.3 Health Check Configuration Tab

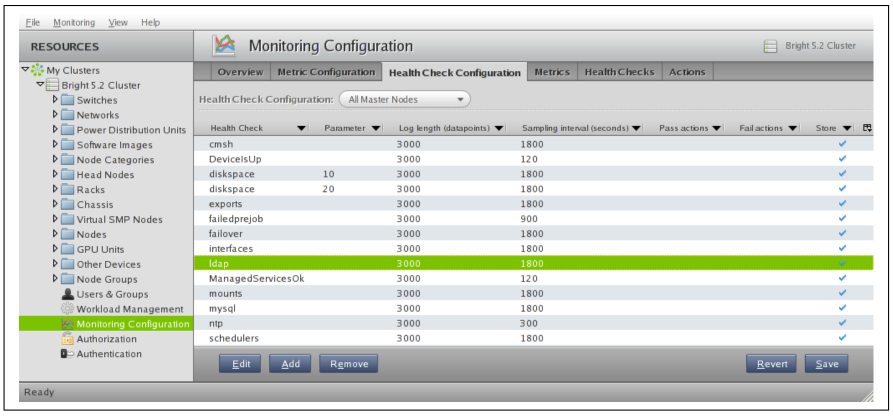
The Health Check Configuration tab behaves in a similar way to the Metric Configuration tab of section 11.4.2, with some differences arising due to working with health checks instead of metric values.

The Health Check Configuration tab allows device categories to be selected for the evaluating the states of health checks. Properties of health checks related to the evaluating these states can then be configured from this tab for the selected device category. These properties are the configuration of the state evaluation parameters themselves (for example, frequency and length of logs), but also the configuration of related properties such as severity levels based on the evaluated state, the actions to launch based on the evaluated state, or the action to launch if the evaluated state is flapping.

The Health Check Configuration tab is initially a blank tab until the device category is selected by using the Health Check Configuration selection box. The selection box selects a device category from a list of built-in categories and user-defined node categories (node categories are introduced in section 3.1.3). On selection, the health checks of the selected device category are listed in the Health Check Configuration tab. Properties of the health checks related to the evaluation of states are only available for configuration and manipulation after the health checks list is displayed. Handling health checks in this manner, via groups of devices, is slightly awkward for just a few machines, but for larger clusters it keeps administration scalable and thus manageable.

[1] For completeness: the time $\texttt{t}_{\texttt{consolidation gone}}$, which is how many seconds into the past the consolidated data goes and is viewable, is given by an analogous equation to that of the equation defining $\texttt{t}_{\texttt{raw gone}}$:

$$\texttt{t}_{\texttt{consolidation gone}} = (\texttt{Log length})_{\texttt{consolidation}} \times (\texttt{Sampling interval})_{\texttt{consolidation}}$$



Health Check Configuration: All Master Nodes

Health Check	Parameter	Log length (datapoints)	Sampling interval (seconds)	Pass actions	Fail actions	Store
cmsh		3000	1800			✓
DeviceIsUp		3000	120			✓
diskspace	10	3000	1800			✓
diskspace	20	3000	1800			✓
exports		3000	1800			✓
failedprejob		3000	900			✓
failover		3000	1800			✓
interfaces		3000	1800			✓
ldap		3000	1800			✓
ManagedServicesOk		3000	120			✓
mounts		3000	1800			✓
mysql		3000	1800			✓
ntp		3000	300			✓
schedulers		3000	1800			✓

Figure 11.28: `cmgui` Monitoring: Health Check Configuration Display After Category Selection

Figure 11.28 shows an example of the `Health Check Configuration` tab after `All master nodes` is chosen as the category. Examples of other categories that could be chosen to have health checks carried out on them are `All ethernet switches` and `All Power Distribution Units`.

With the screen displaying a list of health checks as in figure 11.28, the health checks in the `Health Check Configuration` tab can now be configured and manipulated. The buttons used to do this are: `Edit`, `Add`, `Remove`, `Revert` and `Save`.

These `Health Configuration` tab buttons behave just like the corresponding `Metric Configuration` tab buttons of section 11.4.2, that is:

The `Save` button saves as-yet-uncommitted changes made via the `Add` or `Edit` buttons.

The `Revert` button discards unsaved edits made via the `Edit` button. The reversion goes back to the last save.

The `Remove` button removes a selected health check from the health checks listed.

The remaining buttons, `Edit` and `Add`, open up options dialogs. These are now discussed.

Health Check Configuration Tab: Edit And Add Options

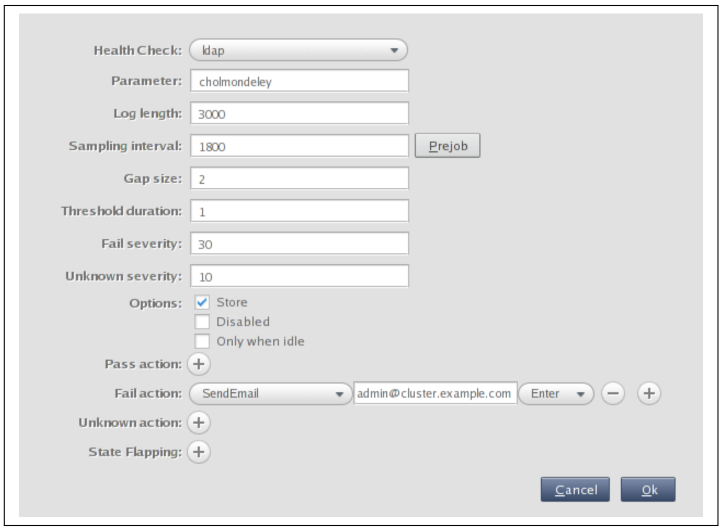
The `Health Check Configuration` tab of figure 11.28 has `Add` and `Edit` buttons. The `Add` button opens up a dialog to add a new health check to the list, and the `Edit` button opens up a dialog to edit a selected health check from the list. The dialogs are very similar to those of the `Add` and `Edit` options of `Metric Configuration` in section 11.4.2. The dialogs for the `Health Check Configuration` tab are as follows (figure 11.29):

- `Health Check`: The name of the health check.
- `Parameter`: The values that the health check script is designed to handle. For example:
  - the health check `ldap` checks if the ldap service is running. It tests the ability to look up a user on the LDAP server using

  `cmsupport` as the default user. If a value is specified for the parameter, it uses that value as the user instead;
  - the health check `portchecker` takes parameter values such as `192.168.0.1 22` to check the if host 192.168.0.1 has port 22 open.

- `Log length`: The maximum number of samples that are stored for the health check. 3000 by default.
- `Sampling interval`: The time between samples. 120s by default.
- `Prejob`: Clicking on this button sets the health check to run before a new job is run from the scheduler of the workload management system, instead of running at regular intervals.
- `Gap size`: The number of missing samples allowed before a null value is stored as a sample value. 2 by default.
- `Threshold duration`: Number of samples in the threshold zone before a health check state is decided to have changed. 1 by default.
- `Fail severity`: The severity value assigned to a FAIL response for a health check. 10 by default.
- `Unknown severity`: The severity value assigned to an UNKNOWN response for a health check. 10 by default.
- `Options` checkboxes:
  - `Store`: If ticked, the health check state data values are saved to the database. Note that health state changes and actions still take place, even if no values are stored.
  - `Disabled`: If ticked, the health state script does not run, and no health check state changes or actions associated with it occur. If `Store` is ticked, the value it stores while `Disabled` is ticked for this health check configuration is an `UNKNOWN` value
  - `Only when idle`: If ticked, the health check script is only run when the system is idling. This burdens a system less, and is useful if the health check is resource-hungry.
- `Pass action`, `Fail action`, `Unknown action`, `State Flapping`: These are all action launchers, which launch an action for a given health state (PASS, FAIL, UNKNOWN) or for a flapping state, depending on whether these states are true or false. Each action launcher is associated with three input boxes. The first selection box decides what action to launch if the state is true. The next box is a plain text-entry box that allows a parameter to be passed to the action. The third box is a selection box again, which decides when to launch the action, depending on which of the following conditions is met:
  - `Enter`: if the state has just started being true. That is, the current sample is in that state, and the previous sample was not in that state.

- During: if the state is true, and ongoing. That is, the current and previous state sample are both in the same state.
- Leave: if the state has just stopped being true. That is, the current sample is not in that state, and the previous sample was in that state.



Health Check:	ldap
Parameter:	cholmondeley
Log length:	3000
Sampling interval:	1800
Gap size:	2
Threshold duration:	1
Fail severity:	30
Unknown severity:	10
Options:	✓ Store ☐ Disabled ☐ Only when idle
Pass action:	+
Fail action:	SendEmail, admin@cluster.example.com, Enter
Unknown action:	+
State Flapping:	+

Figure 11.29: cmgui Monitoring: Health Check Configuration Edit Dialog

11.4.4 Metrics Tab

The Metrics tab displays the list of metrics that can be set in the cluster. Some of these metrics are built-ins, such as CPUUser in the basic example of section 11.1. Other metrics are standalone scripts. New custom metrics can also be built and added as standalone commands or scripts. A useful template for such a script is the testmetric script (Appendix H.1.1).

Metrics can be manipulated and configured.

The Save button saves as-yet-uncommitted changes made via the Add or Edit buttons.

The Revert button discards unsaved edits made via the Edit button. The reversion goes back to the last save.

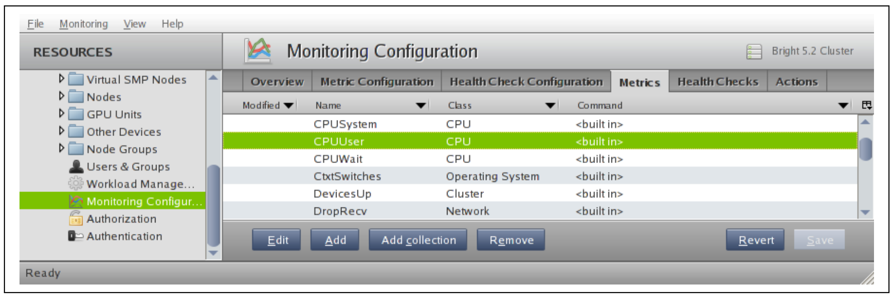
The Remove button removes a selected metric from the list.

The remaining buttons, Edit and Add, open up options dialogs. These are now discussed.

Metrics Tab: Edit And Add Options

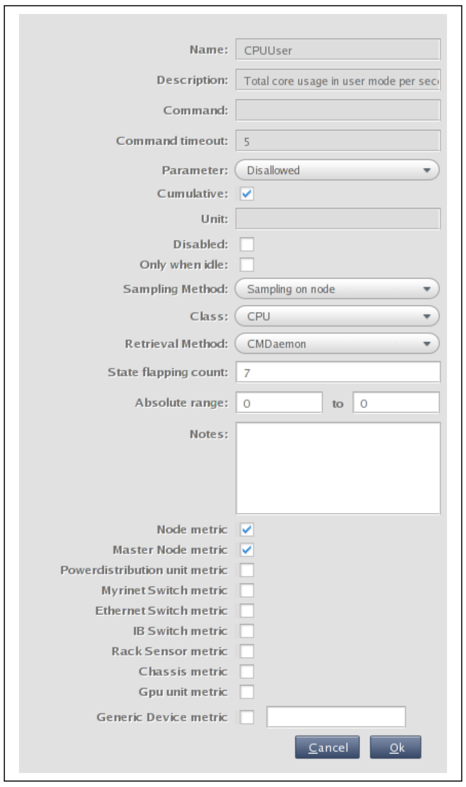
The Metrics tab of figure 11.30 has Add and Edit buttons. The Add button opens up a dialog to add a new metric to the list, and the Edit button opens up a dialog to edit a selected metric from the list. Both dialogs have the following options (figure 11.31):

- Name: the name of the metric.
- Description: the description of the metric.
- Command: the command that carries out the script, or the full path to the executable script.



Modified	Name	Class	Command
	CPUSystem	CPU	<built in>
	CPUUser	CPU	<built in>
	CPUWait	CPU	<built in>
	CtxtSwitches	Operating System	<built in>
	DevicesUp	Cluster	<built in>
	DropRecv	Network	<built in>

Figure 11.30: cmgui Monitoring: Metrics Tab



Name:	CPUUser
Description:	Total core usage in user mode per seco
Command:	
Command timeout:	5
Parameter:	Disallowed
Cumulative:	✓
Unit:	
Disabled:	☐
Only when idle:	☐
Sampling Method:	Sampling on node
Class:	CPU
Retrieval Method:	CMDaemon
State flapping count:	7
Absolute range:	0 to 0
Notes:	
Node metric	✓
Master Node metric	✓
Powerdistribution unit metric	☐
Myrinet Switch metric	☐
Ethernet Switch metric	☐
IB Switch metric	☐
Rack Sensor metric	☐
Chassis metric	☐
Gpu unit metric	☐
Generic Device metric	☐

Figure 11.31: cmgui Monitoring: Metrics Tab, Edit Dialog

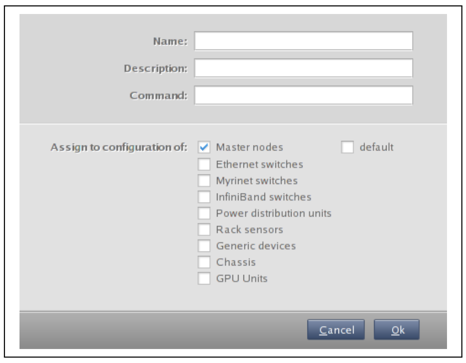
- `Command timeout`: After how many seconds the script should stop running, in case of no response.
- `Parameter`: an optional value that is passed to the script.
- `Cumulative`: whether the metric value is cumulative (for example, based on the bytes-received counter for an Ethernet interface, which accumulates over time), or non-cumulative (for example, based on the temperature, which does not accumulate over time). The value for a cumulative metric is the difference between the current measurement and the preceding measurement, divided by the time interval between the two measurements. The value of the `BytesRecv` metric (a cumulative metric) is thus in bytes/second and ideally indicates the number of bytes per second received through the interface at that time, although it is merely a simple extrapolation. Similarly, the value of the `Uptime` metric (a cumulative metric) is thus ideally a unitless 1 with this definition.
- `Unit`: the unit in which the measurement associated with the metric is done. For a cumulative metric, the unit is applied to the two measurements that occurred over the time that the metric was monitored. So the bytes-received measurement has units of bytes, in contrast to the `BytesRecv` metric, which has units of bytes/second. Similarly, the uptime measurement has units of seconds, in contrast to the `Uptime` metric, which is unitless.
- When to run:
  - `Disabled`: if ticked, the metric script does not run.
  - `Only when idle`: if ticked, the metric script only runs when the system is idling. This burdens the system less if the metric is resource-hungry.
- `Sampling Method`: the options are:
  - `Sampling on master`: The head node samples the metric on behalf of a device. For example: the head node may do this for a PDU, since a PDU does not have the capability to run the cluster management daemon at present, and so cannot itself pass on data values directly when `cmsh` or `cmgui` need them. Similarly, the IPMI health check—not classed as a metric, but classed as a health check, and using the same kind of dialog (section 11.4.5) as here—is also sampled via the head node, since IPMI diagnostics run on the node even when the node is down.
  - `Sampling on node`: The non-head node samples the metric itself. The administrator should ensure that the script is accessible from the non-head node.
- `Class`: An option selected from:
  - `Misc`
  - `CPU`
  - `GPU`

  - Disk
  - Memory
  - Network
  - Environmental
  - Operating System
  - Internal
  - Workload
  - Cluster
  - Prototype

  These options should not be confused with the device category that the metric can be configured for, which is a property of where the metrics can be applied. (The device category possibilities are listed in a bullet point a little further on).

- Retrieval Method:
  - cmdaemon: Metrics retrieved internally using CMDaemon (default).
  - snmp: Metrics retrieved internally using SNMP.
- State flapping count (default value 7): How many times the metric value must cross a threshold within the last 12 samples (a default setting, set in cmd.conf) before it is decided that it is in a flapping state.
- Absolute range: The range of values that the metric takes. A range of 0–0 implies no constraint is imposed.
- Notes: Notes can be made here.
- Which device category the metric is configured for, with choices out of:
  - Node metric
  - Master Node metric
  - Power Distribution Unit metric
  - Myrinet Switch metric
  - Ethernet Switch metric
  - IB Switch metric
  - Rack Sensor metric
  - Chassis metric
  - GPU Unit metric
  - Generic Device metric

  These options should not be confused with the class that the metric belongs to (the earlier Class bullet point), which is the property type of the metric.



Name:

Description:

Command:

Assign to configuration of: [✓] Master nodes [ ] default [ ] Ethernet switches [ ] Myrinet switches [ ] InfiniBand switches [ ] Power distribution units [ ] Rack sensors [ ] Generic devices [ ] Chassis [ ] GPU Units

Cancel Ok

Figure 11.32: `cmgui` Monitoring: Metrics Tab, `Add collection` Dialog

Metrics Tab: Add Collection Option

The `Add Collection` button opens a dialog which is used to create a *metric collection* (figure 11.32). A metric collection is a special metric script, with the following properties:

- It is able to return several metrics of different types when it is run, not just one metric of one type like a normal metric script does—hence the name, "metric collection".
- It autodetects if its associated metrics are able to run, and to what extent, and presents the metrics accordingly. For example, if the metric collection is run on a node which only has 3 CPUs running rather than a default of 4, it detects that and presents the results for just the 3 CPUs.

Further details on metric collections scripts are given in appendix I.

Because handling metric collections is just a special case of handling a metric, the `Add Collection` button dialog is merely a restricted version of the `Add` button dialog. Setting up a metric collection is therefore simplified by having most of the metric fields pre-filled and kept hidden. For example, the `Class` field for a metric collection would have the value `Prototype` in the `Add` button dialog, while this value is pre-filled and invisible in the `Add Collection` dialog. A metric collection can be created with the `Add` dialog, but it would be a little more laborious.

Whatever the method used to create the metric collection, it can always be edited with the `Edit` button, just like any other metric.

Viewing visualizations of a metric collection in `cmgui` is only possible through selection and viewing the separate graphs of its component metrics.

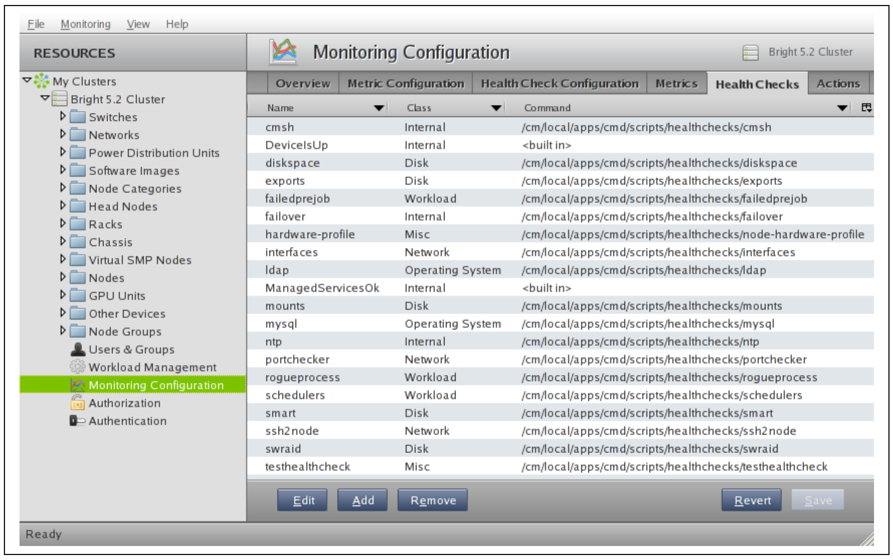
11.4.5 Health Checks Tab

The `Health Checks` tab lists available health checks (figure 11.33). These can be set to run from the system by configuring them from the `Health Check Configuration` tab of section 11.4.3.

What the listed health checks on a newly installed system do are described in appendix H.2.1.

The `remove`, `revert` and `save` buttons work for health checks just like they do for metrics in section 11.4.4

Also, the `edit` and `add` buttons start up dialogs to edit and add health checks. The dialog options for health checks are the same as for editing



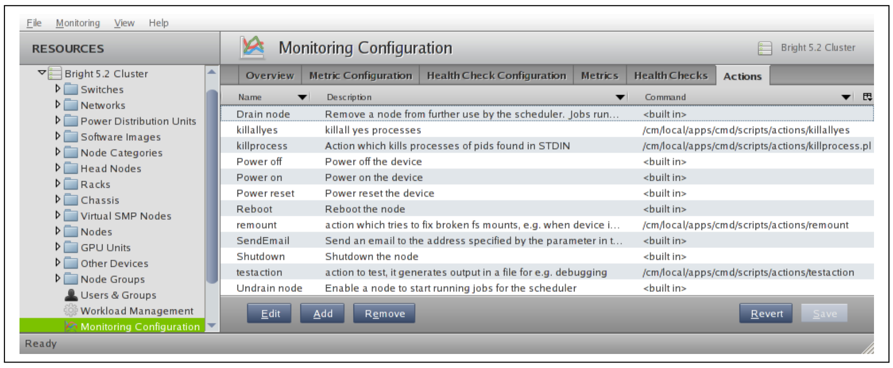
Name	Class	Command
cmsh	Internal	/cm/local/apps/cmd/scripts/healthchecks/cmsh
DeviceIsUp	Internal	<built in>
diskspace	Disk	/cm/local/apps/cmd/scripts/healthchecks/diskspace
exports	Disk	/cm/local/apps/cmd/scripts/healthchecks/exports
failedprejob	Workload	/cm/local/apps/cmd/scripts/healthchecks/failedprejob
failover	Internal	/cm/local/apps/cmd/scripts/healthchecks/failover
hardware-profile	Misc	/cm/local/apps/cmd/scripts/healthchecks/node-hardware-profile
interfaces	Network	/cm/local/apps/cmd/scripts/healthchecks/interfaces
ldap	Operating System	/cm/local/apps/cmd/scripts/healthchecks/ldap
ManagedServicesOk	Internal	<built in>
mounts	Disk	/cm/local/apps/cmd/scripts/healthchecks/mounts
mysql	Operating System	/cm/local/apps/cmd/scripts/healthchecks/mysql
ntp	Internal	/cm/local/apps/cmd/scripts/healthchecks/ntp
portchecker	Network	/cm/local/apps/cmd/scripts/healthchecks/portchecker
rogueprocess	Workload	/cm/local/apps/cmd/scripts/healthchecks/rogueprocess
schedulers	Workload	/cm/local/apps/cmd/scripts/healthchecks/schedulers
smart	Disk	/cm/local/apps/cmd/scripts/healthchecks/smart
ssh2node	Network	/cm/local/apps/cmd/scripts/healthchecks/ssh2node
swraid	Disk	/cm/local/apps/cmd/scripts/healthchecks/swraid
testhealthcheck	Misc	/cm/local/apps/cmd/scripts/healthchecks/testhealthcheck

Figure 11.33: cmgui Monitoring: Health Checks Tab

or adding metrics, with a few exceptions. The exceptions are for options that are inapplicable for health checks, and are elaborated on in appendix H.2.2.

11.4.6 Actions Tab

The Actions tab lists available actions (figure 11.34) that can be set to run on the system from metrics thresholds configuration, as explained in section 11.4.2, and as was done in the basic example of section 11.1. Actions can also be set to run from health check configuration action launcher options as described in section 11.4.3.



Name	Description	Command
Drain node	Remove a node from further use by the scheduler. Jobs run...	<built in>
killallyes	killall yes processes	/cm/local/apps/cmd/scripts/actions/killallyes
killprocess	Action which kills processes of pids found in STDIN	/cm/local/apps/cmd/scripts/actions/killprocess.pl
Power off	Power off the device	<built in>
Power on	Power on the device	<built in>
Power reset	Power reset the device	<built in>
Reboot	Reboot the node	<built in>
remount	action which tries to fix broken fs mounts, e.g. when device i...	/cm/local/apps/cmd/scripts/actions/remount
SendEmail	Send an email to the address specified by the parameter in t...	<built in>
Shutdown	Shutdown the node	<built in>
testaction	action to test, it generates output in a file for e.g. debugging	/cm/local/apps/cmd/scripts/actions/testaction
Undrain node	Enable a node to start running jobs for the scheduler	<built in>

Figure 11.34: cmgui Monitoring: Actions Tab

What the listed actions on a newly installed system do are described in appendix H.3.1.

The remove, revert, and save buttons work as described for metrics in section 11.4.4.

The edit and add buttons start up dialogs to edit or add options to action parameters. Action parameters are described in appendix H.3.2.

11.5 Overview Of Monitoring Data For Devices

These views are set up under the `Overview` tab for various devices that are items under the resource tree in the cluster.

They are a miscellany of monitoring views based on the monitored data for a particular device. The views are laid out as part of an overview tab for that device, which can be a switch, cluster, node, GPU unit, and so on.

When first connecting to a cluster with `cmgui`, the `Overview` tab of the cluster is the default view. The `Overview` tab is also the default view first time a device is clicked on in a `cmgui` session.

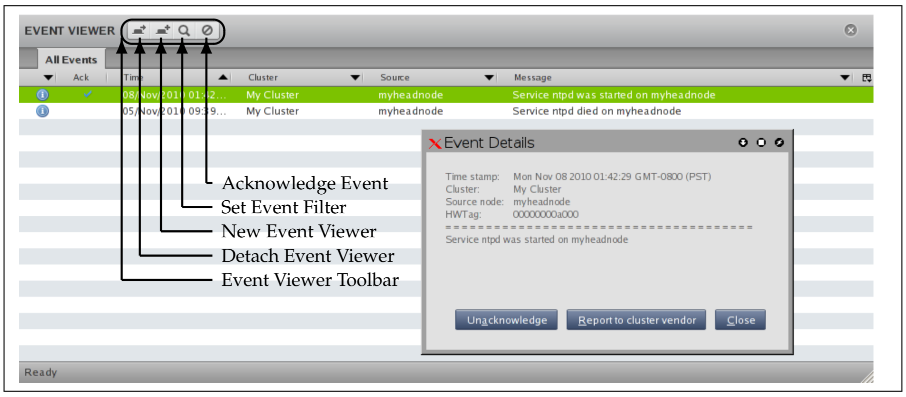
Of the devices, the cluster(s), head node(s) and regular nodes have a relatively extensive `Overview` tab, with a pre-selected mix of information from monitored data. For example, in figure 11.4, a head node is shown with an `Overview` tab presenting memory used, CPU usage, disk usage, network statistics, running processes, and health status. Some of these values are presented with colors and histograms to make the information easier to see.

11.6 Event Viewer

This is a view of events on the cluster(s). It is accessible from the `View` menu of the main window of `cmgui`. By default, there is no log file, but a log file can be activated with the `EventLogger` directive (Appendix C).

The events can be handled and viewed in several ways.

11.6.1 Viewing The Events In `cmgui`



Ack	Time	Cluster	Source	Message
✓	08/Nov/2010 01:42...	My Cluster	myheadnode	Service ntpd was started on myheadnode
	05/Nov/2010 09:39...	My Cluster	myheadnode	Service ntpd died on myheadnode

```
Time stamp:   Mon Nov 08 2010 01:42:29 GMT-0800 (PST)
Cluster:      My Cluster
Source node:  myheadnode
HWTag:        00000000a000
=====================================
Service ntpd was started on myheadnode
```

Figure 11.35: `cmgui` Monitoring: Event Viewer Pane

Double clicking on an event row starts up an `Event Details` dialog (figure 11.35), with buttons to:

- `Acknowledge` or `Unacknowledge` the event, as appropriate. Clicking on `Acknowledge` removes the event from the event view unless the `Show Acknowledged` checkbox has been checked. Any visible acknowledged events have their acknowledged status removed when the `Unacknowledge` button is clicked.
- `Report to cluster vendor`. The report option is used for sending an e-mail about the selected event to the cluster vendor in case troubleshooting and support is needed.

The event viewer toolbar (figure 11.35) offers icons to handle events:

- `detach event viewer`: Detaches the event viewer pane into its own window. Reattachment is done by clicking on the reattachment event viewer icon that becomes available in the detached window.
- `new event viewer filter dialog`: Loads or defines filters (figure 11.36). Filters can be customized according to acknowledgement status, time periods, cluster, nodes, severity, or message text. The filter settings can be saved for later reloading. If the dialog opened by this button is simply given an arbitrary name, and the `Ok` button clicked on to accept the default values, then a default event viewer tabbed pane is added to `cmgui`.

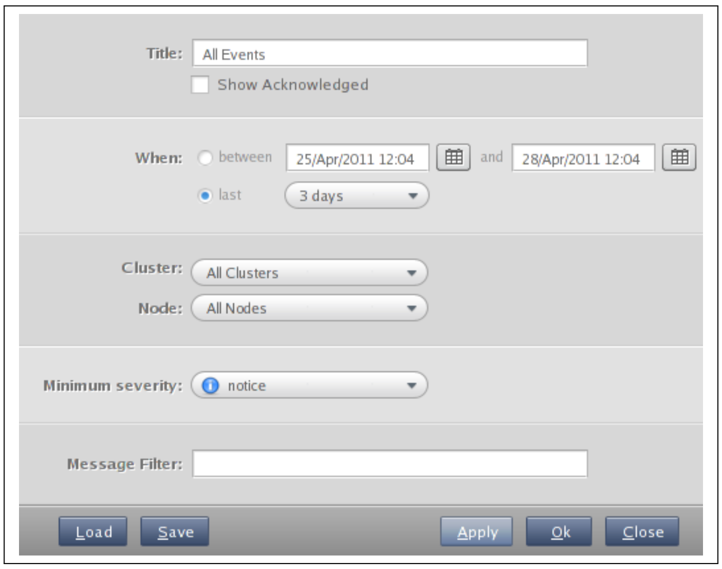

Title: All Events
Show Acknowledged [ ]
When: between 25/Apr/2011 12:04 and 28/Apr/2011 12:04
[✓ last] 3 days
Cluster: All Clusters
Node: All Nodes
Minimum severity: notice
Message Filter:

Figure 11.36: `cmgui` Monitoring: Event Viewer Filter Dialog

- `set event viewer filter dialog`: Adjusts an existing filter with a similar dialog to the `new event viewer filter dialog`.
- `acknowledge event`: Sets the status of one or more selected events in the display pane to an acknowledged state. To set all events in the pane to an acknowledged state, the button is shift-clicked. Acknowledged events are no longer seen, unless the filter setting for the `show acknowledged` checkbox is checked in the `set event filter` option.

11.6.2 Using The Event Bucket From The Shell For Events And For Tagging Device States

Event Bucket Default Behavior

The DirectMon *event bucket* accepts input piped to it, somewhat like the traditional Unix “bit bucket”, `/dev/null`. However, while the bit bucket simply accepts any input and discards it, the event bucket accepts a line of text and makes an event of it. Since the event bucket is essentially an event processing tool, the volumes that are processed by it are obviously less than that which `/dev/null` can handle.

By default, the location of the event bucket is at `/var/spool/cmd/eventbucket`, and a message can be written to the event pane like this:

Example

```
[root@ddnmon61 ~]# echo "Some text" > /var/spool/cmd/eventbucket
```

This adds an event with, by default, the `info` severity level, to the event pane.

Event Bucket Severity Levels

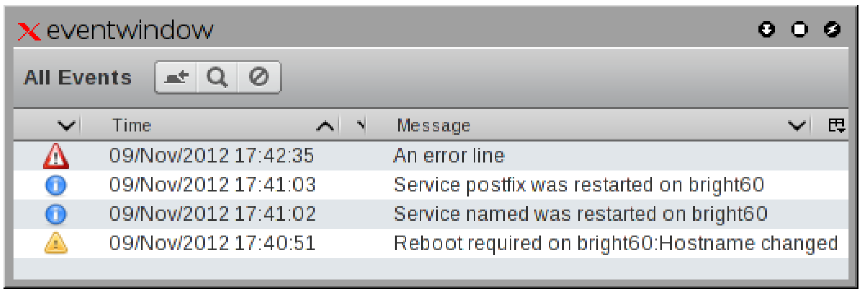
To write events at specific severity levels (section 11.2.6), and not just at the `info` level, the appropriate text can be prepended from the following to the text that is to be displayed:

```
EVENT_SEVERITY_DEBUG:
EVENT_SEVERITY_INFO:
EVENT_SEVERITY_NOTICE:
EVENT_SEVERITY_WARNING:
EVENT_SEVERITY_ERROR:
EVENT_SEVERITY_ALERT:
EVENT_SEVERITY_EMERGENCY:
```

Example

```
echo "EVENT_SEVERITY_ERROR:An error line" > /var/spool/cmd/eventbucket
```

The preceding example displays an output in the `cmgui` event viewer like in figure 11.37:



	Time	Message
⚠	09/Nov/2012 17:42:35	An error line
ℹ	09/Nov/2012 17:41:03	Service postfix was restarted on bright60
ℹ	09/Nov/2012 17:41:02	Service named was restarted on bright60
⚠	09/Nov/2012 17:40:51	Reboot required on bright60:Hostname changed

Figure 11.37: `cmgui` Monitoring: Event Bucket Message Example

Event Bucket Filter

Regex expressions can be used to conveniently filter out the user-defined messages that are about to go into the event bucket from the shell. The filters used are placed in the event bucket filter, located by default at `/cm/local/apps/cmd/etc/eventbucket.filter`.

Event Bucket CMDaemon Directives

The name and location of the event bucket file and the event bucket filter file can be set using the `EventBucket` and `EventBucketFilter` directives from the CMDaemon configuration file directives (Appendix C).

Adding A User-Defined Message To A Device State With The Event Bucket

While the event bucket is normally used to send a message to the event viewer, it can instead be used to add a message to the state of a device. The line passed to the echo command then has the message and device specified in the following format:

STATE.USERMESSAGE[.*device*]:[*message*].

The device can be anything with a status property, such as, for example, a node, a switch, or a chassis.

Example

```
echo "STATE.USERMESSAGE.node001:just right" > /var/spool/cmd/eventbucket
```

The state then shows as:

```
cmsh -c "device ; status node001"
node001 .................. (just right) [   UP   ]
```

If the device is not specified, then the current host of the shell that is executing the echo command is used. For example, running these commands from the head node, ddnmon61, as follows:

Example

```
echo "STATE.USERMESSAGE:too hot" > /var/spool/cmd/eventbucket
ssh node001 'echo "STATE.USERMESSAGE:too cold" > /var/spool/cmd/eventbu\
cket'
```

yields these states:

```
cmsh -c "device ; status ddnmon61"
ddnmon61 .................. (too hot) [   UP   ]
cmsh -c "device ; status node001"
node001 .................. (too cold) [   UP   ]
```

The added text can be cleared with echoing a blank message to that device. For example, for node001 that could be:

```
echo "STATE.USERMESSAGE.node001:" > /var/spool/cmd/eventbucket
```

11.7 The monitoring Modes Of cmsh

This section covers how to use cmsh to configure monitoring. The monitoring mode in cmsh is how metrics and health checks are configured from the command line, and corresponds to the configuration carried out by cmgui in section 11.4.

Visualization of data similar to how cmgui does it in section 11.3 can also be done from cmsh's command line, via its device mode. Graphs can be obtained from cmsh by piping values returned by device mode commands such as latestmetricdata (section 11.8.1) and dumpmetricdata (section 11.8.2) into graphing utilities. These techniques are not covered in this chapter.

Familiarity is assumed with handling of objects as described in the introduction to working with objects (section 3.5.3). When using cmsh's monitoring mode, the properties of these objects—the details of the monitoring settings—are the parameters and values which are accessed and manipulated from the monitoring mode hierarchy within cmsh.

The monitoring "mode" of cmsh gives access to 4 modes under it.

Example

```
[root@myheadnode ~]# cmsh
[myheadnode]% monitoring help | tail -5
============================== Monitoring ==============================
actions ...................... Enter threshold actions mode
healthchecks ................. Enter healthchecks mode
metrics ...................... Enter metrics mode
setup ........................ Enter monitoring configuration setup mode
```

These 4 modes are regarded as being the top level monitoring related modes:

- `monitoring actions`
- `monitoring healthchecks`
- `monitoring metrics`
- `monitoring setup`

The word `monitoring` is therefore merely a grouping label prefixed inseparably to these 4 modes. The syntax of the 4 bulleted commands above is thus consistent with that of the other top level `cmsh` modes.

The sections 11.7.1, 11.7.2, 11.7.3, and 11.7.4 give examples of how objects are handled under these 4 monitoring modes. To avoid repeating similar descriptions, section 11.7.1 is relatively detailed, and is often referred to by the other sections.

11.7.1 The `monitoring actions` Mode In `cmsh`

The `monitoring actions` mode of `cmsh` corresponds to the `cmgui` `actions` tab of section 11.4.6.

The `monitoring actions` mode handles actions objects in the way described in the introduction to working with objects (section 3.5.3). A typical reason to handle action objects—the properties associated with an action script or action built-in—might be to view the actions available, or to add a custom action for use by, for example, a metric or health check.

This section continues the `cmsh` session started above, giving examples of how the `monitoring actions` mode is used.

The `monitoring actions` Mode In `cmsh`: `list`, `show`, And `get`

The `list` command by default lists the names and command scripts available in `monitoring actions` mode:

Example

```
[myheadnode]% monitoring actions
[myheadnode->monitoring->actions]% list
Name (key)               Command
------------------------ ------------------------------------------------
Drain node               <built-in>
Power off                <built-in>
Power on                 <built-in>
Power reset              <built-in>
Reboot                   <built-in>
SendEmail                <built-in>
```

```
Shutdown                 <built-in>
Undrain node             <built-in>
killprocess              /cm/local/apps/cmd/scripts/actions/killprocess.+
remount                  /cm/local/apps/cmd/scripts/actions/remount
testaction               /cm/local/apps/cmd/scripts/actions/testaction
```

The above shows the actions available on a newly installed system. The details of what they do are covered in appendix H.3.1.

The show command of cmsh displays the parameters and values of a specified action:

Example

```
[myheadnode->monitoring->actions]% show poweroff
Parameter                      Value
------------------------------ ------------------------------------------------
Command                        <built-in>
Description                    Power off the device
Name                           Power off
Revision
Run on                         master
Timeout                        5
isCustom                       no
[myheadnode->monitoring->actions]%
```

The meanings of the parameters are covered in appendix H.3.2.

Tab-completion suggestions with the show command suggest arguments corresponding to names of action objects:

Example

```
[myheadnode->monitoring->actions]% show
```

A double-tap on the tab key to get tab-completions suggestions for show in the above displays the following:

Example

```
drainnode    killprocess   poweron     reboot    sendemail   testaction
killallyes   poweroff      powerreset  remount   shutdown    undrainnode
```

The Power off action name, for example, corresponds with the argument poweroff. By default, the arguments are the action names in lower case, with the spaces removed. However, they are space- and case-insensitive, so typing in show "Power off" with the quotes included to pass the space on is also valid.

The get command returns the value of an individual parameter of the action object:

Example

```
[myheadnode->monitoring->actions]% get poweroff runon
master
[myheadnode->monitoring->actions]%
```

The `monitoring actions` **Mode In** `cmsh`**:** `add`**,** `use`**,** `remove`**,** `commit`**,** `refresh`**,** `modified`**,** `set`**,** `clear`**, And** `validate`
In the basic example of section 11.1, in "Adding The Action To The Actions List", the name, description and command for an action were added via a dialog in the `Actions` tab of `cmgui`.

The equivalent is done in `cmsh` with `add` and `set` commands. The `add` command adds an object, makes it the current object, and sets its name at the same time; while the `set` command sets values.

If there is no `killallyes` action already, then the name is added in the `actions` mode with the `add` command as follows:

Example

```
[myheadnode->monitoring->actions]% add killallyes
[myheadnode->monitoring->actions*[killallyes*]]%
```

The converse to the `add` command is the `remove` command, which removes the action.

The `use` command is the usual way of "using" an object, where "using" means that the object being used is referred to by default by any command run. So if the `killallyes` object already exists, then `use killallyes` drops into the context of an already existing object (i.e. it "uses" the object).

The `set` command sets the value of each parameter displayed by a `show` command:

Example

```
[myheadnode->monitoring->actions*[killallyes*]]% set description "kill \
             all yes processes"
```

The `clear` command is the converse of `set`, and removes any value for a given parameter.

Example

```
[myheadnode->monitoring->actions*[killallyes*]]% clear command
```

The `validate` command checks if the object has all required values set to sensible values. The commands `refresh`, `modified` and `commit` work as expected from the introduction to working with objects (section 3.5.3). So, for example, `commit` only succeeds if the `killallyes` object passes validation.

Example

```
[myheadnode->monitoring->actions*[killallyes*]]% validate
Code  Field                     Message
----- ------------------------ ------------------------------------------
4     command                   command should not be empty
```

Here validation fails because the parameter `Command` has no value set for it yet. This is remedied with `set` acting on the parameter (some prompt text elided for display purposes):

Example

```
[...*]]% set command "/cm/local/apps/cmd/scripts/actions/killallyes"
[...*]]% commit
[...]]%
```

Validation then succeeds and the `commit` successfully saves the `killallyes` object.

Note that validation does not check if the script itself exists. It solely does a sanity check on the values of the parameters of the object, which is another issue. If the `killallyes` script does not yet exist in the location given by the parameter, it can be created as suggested in the basic example of section 11.1, in "Setting Up The Kill Action".

11.7.2 The `monitoring healthchecks` Mode in `cmsh`

The `monitoring healthchecks` mode of `cmsh` corresponds to the `cmgui Health Checks` tab of section 11.4.5.

The `monitoring healthchecks` mode handles health check objects in the way described in the introduction to working with objects (section 3.5.3). A typical reason to handle health check objects—the properties associated with an health check script or health check built-in—might be to view the health checks already available, or to add a health check for use by a device resource.

This section goes through a `cmsh` session giving some examples of how this mode is used and to illustrate what it looks like.

The `monitoring healthchecks` Mode in `cmsh`: `list`, `show`, And `get`
In `monitoring healthchecks` mode, the `list` command by default lists the names of the health check objects along with their command scripts:

Example

```
[ddnmon61->monitoring->healthchecks]% format name:18 command:55
[ddnmon61->monitoring->healthchecks]% list
name (key)         command
------------------ -------------------------------------------------------
DeviceIsUp         <built-in>
ManagedServicesOk  <built-in>
chrootprocess      /cm/local/apps/cmd/scripts/healthchecks/chrootprocess
cmsh               /cm/local/apps/cmd/scripts/healthchecks/cmsh
diskspace          /cm/local/apps/cmd/scripts/healthchecks/diskspace
...
```

The `format` command, introduced in section 3.5.3, is used here with the given column width values to avoid truncating the full path of the commands in the display.

The above example shows a truncated list of health checks that can be set for sampling on a newly installed system. The details of what these health checks do is covered in appendix H.2.1.

The `show` command of `cmsh` displays the parameters and values of a specified health check:

Example

```
[myheadnode->monitoring->healthchecks]% show deviceisup
Parameter                 Value
------------------------  ------------------------------------------------
Class of healthcheck      internal
Command                   <built-in>
Description               Returns PASS when device is up, closed or insta+
Disabled                  no
Extended environment      no
Name                      DeviceIsUp
Notes                     <0 bytes>
Only when idle            no
Parameter permissions     disallowed
Revision
Sampling method           samplingonmaster
State flapping count      7
Timeout                   5
Valid for                 node,master,pdu,ethernet,myrinet,ib,racksensor,+
[myheadnode->monitoring->healthchecks]%
```

The meanings of the parameters are covered in appendix H.2.2.

As detailed in section 11.7.1, tab-completion suggestions for the `show` command suggest arguments corresponding to names of objects that can be used in this mode. For `show` in `healthchecks` mode, tab-completion suggestions give the following as possible health check objects:

Example

```
[myheadnode->monitoring->healthchecks]% show
chrootprocess       failover            mysql               ssh2node
cmsh                hardware-profile    ntp                 swraid
deviceisup          interfaces          portchecker         testhealthcheck
diskspace           ldap                rogueprocess
exports             managedservicesok   schedulers
failedprejob        mounts              smart
[myheadnode->monitoring->healthchecks]% show
```

The `get` command returns the value of an individual parameter of a particular health check object:

Example

```
[myheadnode->monitoring->healthchecks]% get deviceisup description
Returns PASS when device is up, closed or installing
[myheadnode->monitoring->healthchecks]%
```

The `monitoring healthchecks` Mode In `cmsh`: `add`, `use`, `remove`, `commit`, `refresh`, `modified`, `set`, `clear`, And `validate`

The remaining commands in `monitoring healthchecks` mode: `add`, `use`, `remove`, `commit`, `refresh`, `modified`, `set`, `clear`, and `validate`; all work as outlined in the introduction to working with objects (section 3.5.3). More detailed usage examples of these commands within a `monitoring` mode are given in `Cmsh Monitoring Actions` (section 11.7.1).

In the basic example of section 11.1, a metric script was set up from `cmgui` to check if thresholds were exceeded, and if so, to launch an action.

A functionally equivalent task can be set up by creating and configuring a health check, because metrics and health checks are so similar in concept. This is done here to illustrate how cmsh can be used to do something similar to what was done with cmgui in the basic example. A start is made on the task by creating a health check object and setting its values using the monitoring healthchecks mode of cmsh. The task is completed in the section on the monitoring setup mode in section 11.7.4.

To start the task, cmsh's add command is used to create the new health check object:

Example

```
[root@myheadnode ~]# cmsh
[myheadnode]% monitoring healthchecks
[myheadnode->monitoring->healthchecks]% add cpucheck
[myheadnode->monitoring->healthchecks*[cpucheck*]]%
```

The set command sets the value of each parameter displayed by a show command (some prompt text elided for layout purposes):

Example

```
[...]% set command /cm/local/apps/cmd/scripts/healthchecks/cpucheck
[...]% set description "CPUuser under 50%?"
[...]% set parameterpermissions disallowed
[...]% set samplingmethod samplingonmaster
[...]% set validfor master
[...]]% commit
```

Since the cpucheck script does not yet exist in the location given by the parameter command, it needs to be created:

```
#!/bin/bash

## echo PASS if CPUUser < 50
## cpu is a %, ie: between 0 and 100

cpu=`mpstat 1 1 | tail -1 | awk '{print $3}'`
comparisonstring="$cpu"" < 50"

if (( $(bc <<< "$comparisonstring") )); then
  echo PASS
else
  echo FAIL
fi
```

The script should be placed in the location suggested by the object, /cm/local/apps/cmd/scripts/healthchecks/cpucheck, and made executable with a chmod 700.

The cpucheck object is handled further within the cmsh monitoring setup mode in section 11.7.4 to produce a fully configured health check.

11.7.3 The `monitoring metrics` Mode In `cmsh`

The `monitoring metrics` mode of `cmsh` corresponds to the `cmgui metrics` tab of section 11.4.4.

The `monitoring metrics` mode of `cmsh` handles metrics objects in the way described in the introduction to working with objects (section 3.5.3). A typical reason to handle metrics objects—the properties associated with a metrics script or metrics built-in—might be to view the configuration metrics already being used for sampling by a device category, or to add a metric for use by a device category.

This section goes through a `cmsh` session giving some examples of how this mode is used and to illustrate its behavior.

The `monitoring metrics` Mode In `cmsh`: `list`, `show`, And `get`

In metrics mode, the list command by default lists the names and command scripts available for setting for device categories:

Example

```
[root@myheadnode ~]# cmsh
[myheadnode]% monitoring metrics
[myheadnode->monitoring->metrics]% list
Name (key)                    Command
----------------------------- --------------------------------------------
AlertLevel                    <built-in>
AvgExpFactor                  <built-in>
AvgJobDuration                <built-in>
...
```

The above shows a truncated list of the metrics that may be used for sampling on a newly installed system. What these metrics do is described in appendix H.1.1.

The `show` command of `cmsh` displays the parameters and values of a specified metric:

Example

```
[myheadnode->monitoring->metrics]% show cpuuser
Parameter                Value
------------------------ ------------------------------------------------
Class of metric          cpu
Command                  <built-in>
Cumulative               yes
Description              Percent of node-wide core time spent in user mode
Disabled                 no
Extended environment     no
Maximum                  <range not set>
Measurement Unit         %
Minimum                  <range not set>
Name                     CPUUser
Notes                    <15 bytes>
Only when idle           no
Parameter permissions    disallowed
Retrieval method         cmdaemon
Revision
Sampling method          samplingonnode
```

```
State flapping count    7
Timeout                 5
Valid for               node,master
[myheadnode->monitoring->metrics]%
```

The meanings of the parameters above are explained in appendix H.1.2.

Tab-completion suggestions for the `show` command suggest arguments corresponding to names of objects (the names returned by the `list` command) that may be used in a monitoring mode. For `metrics` mode, `show`, followed by a double-tap on the tab key, displays a large number of possible metrics objects:

Example

```
[myheadnode->monitoring->metrics]% show
Display all 130 possibilities? (y or n)
alertlevel                 droprecv                   ipoutrequests
avgexpfactor               dropsent                   ipreasmoks
avgjobduration             errorsrecv                 ipreasmreqds
await_sda                  errorssent                 loadfifteen
...
```

The `get` command returns the value of an individual parameter of a particular metric object:

Example

```
[myheadnode->monitoring->metrics]% get CPUUser description
Percent of node-wide core time spent in user mode
```

The `monitoring metrics` Mode In `cmsh`: `add`, `use`, `remove`, `commit`, `refresh`, `modified`, `set`, `clear`, And `validate`

The remaining commands in `monitoring metrics` mode: `add`, `use`, `remove`, `commit`, `refresh`, `modified`, `set`, `clear`, and `validate`; all work as outlined in the introduction to working with objects (section 3.5.3). More detailed usage examples of these commands within a `monitoring` mode are given in `Cmsh Monitoring Actions` (section 11.7.1).

Adding a metric collections script to the framework is possible from this point in `cmsh` too. Details on how to do this are given in appendix I.

11.7.4 The `monitoring setup` Mode in `cmsh`

The `cmsh` `monitoring setup` mode corresponds to the `cmgui` `Metric Configuration` and `Health Check Configuration` tabs of sections 11.4.2 and 11.4.3.

The `monitoring setup` mode of `cmsh`, like the `Metric Configuration` and the `Health Check Configuration` tabs of `cmgui`, is used to select a device category. Properties of metrics or of health checks can then be configured for the selected device category. These properties are the configuration of the sampling parameters themselves (for example, frequency and length of logs), but also the configuration of related properties such as thresholds, consolidation, actions launched when a metric threshold is crossed, and actions launched when a metric or health state is flapping.

The `setup` mode only functions in the context of metrics or health checks, and therefore these contexts under the `setup` mode are called submodes. On a newly installed system, a `list` command from the `monitoring setup` prompt displays the following account of metrics and health checks that are in use by device categories:

Example

```
[root@myheadnode ~]# cmsh
[myheadnode]% monitoring setup
[myheadnode->monitoring->setup]% list
Category                Metric configuration    Health configuration
----------------------- ----------------------  --------------------
Chassis                 <2 in submode>          <1 in submode>
EthernetSwitch          <13 in submode>         <1 in submode>
GenericDevice           <2 in submode>          <1 in submode>
GpuUnit                 <3 in submode>          <0 in submode>
IBSwitch                <2 in submode>          <1 in submode>
HeadNode              <90 in submode>         <14 in submode>
MyrinetSwitch           <2 in submode>          <1 in submode>
PowerDistributionUnit   <5 in submode>          <1 in submode>
RackSensor              <2 in submode>          <1 in submode>
default                 <35 in submode>         <11 in submode>

[myheadnode->monitoring->setup]%
```

A device category must always be used when handling the properties of the metrics and health checks configurable under the submodes of `monitoring setup`. The syntax of a configuration submode, `metricconf` or `healthconf`, therefore requires the device category as a mandatory argument, and tab-completion suggestions become quite helpful at this point.

Examples are now given of how the metric configuration `metricconf` and health check configuration `healthconf` submodes are used:

The `monitoring setup` Mode in `cmsh`: `metricconf`
Continuing with the session above, the `metricconf` option can only be used with a device category specified. Tab-completion suggestions for `metricconf` suggest the following possible device categories:

Example

```
[myheadnode->monitoring->setup]% metricconf
chassis                 gpuunit                 powerdistributionunit
default                 ibswitch                racksensor
ethernetswitch          headnode
genericdevice           myrinetswitch
```

A category can be chosen with the `use` command, and `show` shows the properties of the category. With a category selected, the `metricconf` or `healthconf` submodes can then be invoked:

Example

```
[myheadnode->monitoring->setup]% use headnode
[myheadnode->monitoring->setup[HeadNode]]% show
```

```
Parameter                      Value
------------------------------ -------------------------------
Category                       HeadNode
Health configuration           <14 in submode>
Metric configuration           <90 in submode>
Normal pickup interval         180
Revision
Scrutiny pickup interval       60
[myheadnode->monitoring->setup[HeadNode]]% metricconf
[myheadnode->monitoring->setup[HeadNode]->metricconf]%
```

As an aside, data values are picked up by CMDaemon on the head node from the devices that run metrics or health checks at certain pickup intervals. The interval value used for the pickups is set with these category-level parameters:

- `Normal pickup interval`: used when all metrics are below threshold levels, and all health checks succeed.
- `Scrutiny pickup interval`: used when at least one of the following is true:
  - a metric is above a threshold level
  - a health check is in a failed state

The idea behind this is that the more frequent pickup lets the administrator get up-to-date values sooner during abnormal conditions. Increasing the value of these intervals reduces the amount of data values picked up over time for the category. To stop data values being picked up completely, a category, or other node grouping, can be set to a `CLOSED` state (section 6.5.4).

Dropping into a submode—in the example given, the `metricconf` submode—could also have been done directly in one command: `metricconf mastermode`. The synopsis of the command in the example is actually `[[[monitoring] setup] metricconf] headnode`, where the optional parts of the command are invoked depending upon the context indicated by the prompt. The example below clarifies this (some prompt text elided for display purposes):

Example

```
[...->monitoring->setup[HeadNode]->metricconf]% exit; exit; exit; exit
[...]% monitoring setup metricconf headnode
[...->monitoring->setup[HeadNode]->metricconf]% exit; exit; exit
[...->monitoring]% setup metricconf headnode
[...->monitoring->setup[HeadNode]->metricconf]% exit; exit
[...->monitoring->setup]% metricconf headnode
[...->monitoring->setup[HeadNode]->metricconf]% exit
[...->monitoring->setup[HeadNode]]% metricconf
[...->monitoring->setup[HeadNode]->metricconf]%
```

A list of metrics that have been set to do sampling for the device category `headnode` is obtained with `list`. Since there are many of these, only 10 lines are displayed in the list shown below by piping it through `head`:

Example

```
[myheadnode->monitoring->setup[HeadNode]->metricconf]% list | head
Metric                   Metric Param       Samplinginterval
------------------------ ------------------ ------------------
AlertLevel               max                0
AlertLevel               sum                0
AvgExpFactor                                120
AvgJobDuration           defq               60
BufferMemory                                120
BytesRecv                eth0               120
BytesRecv                eth1               120
BytesSent                eth0               120
BytesSent                eth1               120
CMDMemUsed                                  120
```

Besides `list`, an alternative way to get a list of metrics that are set to sample for headnode is to use the tab-completion suggestions to the `use` command.

The `use` command is normally used to drop into the configuration properties of the metric so that parameters of the metric object can be configured:

Example

```
[myheadnode->monitoring->setup[HeadNode]->metricconf]% use cpuuser
[myheadnode->monitoring->setup[HeadNode]->metricconf[CPUUser]]% show
Parameter                      Value
------------------------------ ------------------------------
Consolidators                  <3 in submode>
Disabled                       no
GapThreshold                   2
LogLength                      3000
Metric                         CPUUser
MetricParam
Only when idle                 no
Revision
Sampling Interval              120
Stateflapping Actions
Store                          yes
ThresholdDuration              1
Thresholds                     <1 in submode>
[myheadnode->monitoring->setup[HeadNode]->metricconf[CPUUser]]%
```

The `add` command adds a metric to be set for sampling for the device category. The list of all possible metrics that can be added to the device category can be seen with the command `monitoring metrics list`, or more conveniently, simply with tab-completion suggestions to the `add` command at the `[...metricconf]%` prompt in the above example.

The above example indicates that there are two submodes for each metric configuration: `Consolidators` and `Thresholds`. Running the `consolidators` or `thresholds` commands brings `cmsh` into the chosen submode.

Consolidation and threshold manipulation only make sense in the context of a metric configuration, so at the `metricconf` prompt in the example above (before `use cpuuser` is executed), the commands `thresholds cpuuser` or `consolidators cpuuser` can be executed as more direct ways of getting to the chosen submode.

The `thresholds` submode If, continuing on from the above example, the thresholds submode is entered, then the `list` command lists the existing thresholds. If the basic example of section 11.1 has already been carried out on the system, then a threshold called `killallyesthreshold` is already there with an assigned action `killallyes`. The properties of each threshold can be shown (some prompt text elided for layout purposes):

Example

```
[...metricconf]% thresholds
[...metricconf[CPUUser]]% thresholds
[...metricconf[CPUUser]]->thresholds]% list
Name (key)               Bound            Severity
------------------------ ---------------- --------------------
killallyesthreshold      50               10
[...metricconf[CPUUser]]->thresholds]% show killallyesthreshold
Parameter                      Value
------------------------------ ------------------------------
Actions                        enter: killallyes()
Bound                          50
Name                           killallyesthreshold
Revision
Severity                       10
UpperBound                     yes
```

The meanings of the parameters are explained in the GUI equivalent of the above example in section 11.4.2 in the section labeled "Metric Configuration: Thresholds Options". The object manipulation commands introduced in section 3.5.3 work as expected at this `cmsh` prompt level: `add` and `remove` add and remove a threshold; `set`, `get`, and `clear` set and get values for the parameters of each threshold; `refresh` and `commit` revert and commit changes; `use` "uses" the specified threshold, making it the default for commands; `validate` applied to the threshold checks if the threshold object has sensible values; and `append` and `removefrom` append an action to, and remove an action from, a specified threshold.

The `append` and `removefrom` commands correspond to the ⊕ and ⊖ widgets of `cmgui` in figure 11.25 and work with parameters that can have multiple values. For example, a `sendemail` action with a parameter `root` can be appended to the `Actions` parameter, which already has the `killallyes` action as a value. This sends an e-mail to the root mail account. A `get` command can be run to see the values for the threshold actions:

Example

```
[...->thresholds*]% append killallyesthreshold actions sendemail root
[...->thresholds*]% get killallyesthreshold actions
enter: killallyes()
enter: SendEmail(root)
```

When using the `sendemail()` command, the e-mail is sent to the parameter value. If no parameter is specified, then the e-mail is sent to root by default, unless a value for `administratore-mail` has been set for the cluster as described in the text accompanying figure 4.2.

The `actions` command takes flags that define when the action is taken on crossing the threshold. These are:

- `-e|--enter`: Run on entering the zone. This is also the implied default when the flag is omitted.
- `-l|--leave`: Run on leaving the threshold zone.
- `-d|--during`: Run during the time the value is within the threshold zone.

In the example, the "Actions" parameter now has the value of the built-in action name, `sendemail`, as well as the value of the action script name, `killallyes`. This means that both actions run when the threshold condition is met.

The `consolidators` submode If, continuing on with the preceding example, the `consolidators` submode is entered, then the `list` command lists the consolidators running on the system. On a newly installed system there are three consolidators by default for each metric set for a device category. Each consolidator has an appropriately assigned time `Interval`, in seconds. The `show` command shows the parameters and values of a specific consolidator:

Example

```
[...metricconf[CPUUser]->thresholds*]% exit
[...metricconf[CPUUser]]% consolidators
[...metricconf[CPUUser]->consolidators]% list
Name (key)               Length     Interval
------------------------ ---------- ----------
Day                      1000       86400
Hour                     2000       3600
Week                     1000       604800
[...metricconf[CPUUser]->consolidators]% show day
Parameter                      Value
------------------------------ --------------------------------
Interval                       86400
Kind                           AVERAGE
Length                         1000
Name                           Day
Offset                         0
Revision
```

The meanings of the parameters are explained in the GUI equivalent of the above example in section 11.4.2 in the section labeled "Metric Configuration: Consolidators Options".

The object manipulation commands introduced in section 3.5.3 work as expected at this `cmsh` prompt level: `add` and `remove` add and remove a consolidator; `set`, `get`, and `clear` set and get values for the parameters of each consolidator; `refresh` and `commit` revert and commit changes; `use` "uses" the specified consolidator, making it the default for commands; and `validate` applied to the consolidator checks if the consolidator object has sensible values.

The `monitoring setup` Mode in `cmsh`: `healthconf`

The `healthconf` submode is the alternative to the `metricconf` submode under the main `monitoring setup` mode. Like the `metricconf` option, `healthconf` too can only be used with a device category specified.

If the session above is continued, and the device category `headnode` is kept unchanged, then the `healthconf` submode can be invoked with:

```
[...metricconf[CPUUser]->consolidators]% exit; exit; exit
[myheadnode->monitoring->setup[HeadNode]]% healthconf
[...healthconf]%
```

Alternatively, the `healthconf` submode with the `headnode` device category could also have been reached from `cmsh`'s top level prompt by executing `monitoring setup healthconf headnode`.

The health checks set to do sampling in the device category `headnode` are listed:

Example

```
[myheadnode->monitoring->setup[HeadNode]->healthconf]% list
HealthCheck               HealthCheck Param  Check Interval
------------------------- ------------------ ------------------
DeviceIsUp                                   120
ManagedServicesOk                            120
chrootprocess                                900
cmsh                                         1800
diskspace                 2% 10% 20%         1800
exports                                      1800
failedprejob                                 900
failover                                     1800
interfaces                                   1800
ldap                                         1800
mounts                                       1800
mysql                                        1800
ntp                                          300
schedulers                                   1800
smart                                        1800
```

The `use` command would normally be used to drop into the health check object. However `use` can also be an alternative to the `list` command, since tab-completion suggestions to the `use` command get a list of currently configured health checks for the `headnode` too.

The `add` command adds a health check into the device category. The list of all possible health checks that can be added to the category can be seen with the command `monitoring healthchecks list`, or more conveniently, simply with tab-completion suggestions to the `add` command.

At the end of section 11.7.2 a script called `cpucheck` was built. This script was part of a task to use health checks instead of metric threshold actions to set up the functional equivalent of the behavior of the basic example of section 11.1. In this section the task is continued and completed, and on the way how to use the health checks configuration object methods to do this is shown.

First, the script is added, and as usual when using `add`, the prompt drops into the level of the added object. The `show` command acting on the object displays the following default values for its parameters (some prompt text elided for display purposes):

Example

```
[...[HeadNode]->healthconf]% add cpucheck
[...*[HeadNode*]->healthconf*[cpucheck*]]% show
Parameter                      Value
------------------------------ ------------------------------
Check Interval                 120
Disabled                       no
Fail Actions
Fail severity                  10
GapThreshold                   2
HealthCheck                    cpucheck
HealthCheckParam
LogLength                      3000
Only when idle                 no
Pass Actions
Revision
Stateflapping Actions
Store                          yes
ThresholdDuration              1
Unknown Actions
Unknown severity               10
[...*[HeadNode*]->healthconf*[cpucheck*]]%
```

The details of what these parameters mean is covered in section 11.4.3 where the edit and add dialog options for a health check state shown in figure 11.29 are explained.

The object manipulation commands introduced in section 3.5.3 work as expected at the `healthconf` prompt level in the example above: `add` and `remove` add and remove a health check; `set`, `get`, and `clear` set and get values for the parameters of each health check; `refresh` and `commit` revert and commit changes; `use` "uses" the specified health check, making it the default for commands; and `validate` applied to the health check checks if the health check object has sensible values; and `append` and `removefrom` append an action to, and remove an action from, a specified health check action parameter.

The `append` and `removefrom` commands correspond to the ⊕ and ⊖ widgets of `cmgui` in figure 11.29 and work with parameters that can have multiple values:

The action `killallyes` was set up to be carried out with the metric `CPUUser` in the basic example of section 11.1. The action can also be carried out with a `FAIL` response for the `cpucheck` health check by using `append` command:

Example

```
[...healthconf*[cpucheck*]]% append failactions killallyes
[...healthconf*[cpucheck*]]%
```

Sending an e-mail to root can be done by appending further:

Example

```
[...healthconf*[cpucheck*]]% append failactions sendemail root
[...healthconf*[cpucheck*]]% get failactions
enter: SendEmail(root)
enter: killallyes()
[...healthconf*[cpucheck*]]%
```

11.8 Obtaining Monitoring Data Values

The monitoring data values that are logged by devices can be used to generate graphs using the methods in section 11.3. However, sometimes an administrator would like to have the data values that generate the graphs instead, perhaps to import them into a spreadsheet for further direct manipulation, or to pipe them into a utility such as `gnuplot`.

The values can be obtained from within device mode in various ways.

11.8.1 The Latest Data Values—The `latest*data` Commands

Within `device` mode, the following commands display the latest metrics and health checks being monitored, along with their latest data:
For metrics:

- `metrics`: The `metrics` command in `device` mode lists the metrics that are currently configured to be monitored for a specified device. These correspond to the metrics shown in `cmgui` in the "`Metric Configuration`" tab for a specific device.

- `latestmetricdata`: The `latestmetricdata` command for a particular device displays the most recent value that has been obtained by the monitoring system for each item in the list of active metrics.

Similarly for health checks,

- `healthchecks`: The `healthchecks` command in `device` mode lists the health checks that are currently configured to be monitored for a device. These correspond to the health checks shown in `cmgui` in the "`Health Check Configuration`" tab for a device.

- `latesthealthdata`: The `latesthealthdata` command for a particular device displays the most recent value that has been obtained by the monitoring system for each item in the list of active health checks. For displaying a health check response on demand, the `check` command (page 456) can be used.

Using The `metrics` And `healthchecks` Commands

When using the `metrics` or `healthchecks` command, the device must be specified (some output elided):

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device
[ddnmon61->devices]% metrics node001
LoadOne
LoadFive
LoadFifteen
PageFaults
MajorPageFaults
Uptime
MemoryUsed
...
```

Using The `latestmetricdata` And `latesthealthdata` Commands

When using the `latestmetricdata` or `latesthealthdata` commands, the device must be specified (some output elided):

Example

```
[ddnmon61->device]% use node001
[ddnmon61->device[node001]]% latestmetricdata
Metric                   Value             Age (sec.) Info Message
------------------------ ----------------- ---------- ----------------
AlertLevel:max           30                76         FAIL schedulers
AlertLevel:sum           30                76         FAIL schedulers
BytesRecv:BOOTIF         690.733           76
BytesSent:BOOTIF         449.1             76
CPUIdle                  99.0333           76
CPUIrq                   0                 76
CPUNice                  0                 76
...
```

11.8.2 Data Values Over Time—The `dump*` Commands

Within `device` mode, the following commands display monitoring data values over a specified period:

- `dumpmetricdata`: The `dumpmetricdata` command for a particular device displays the values of metrics obtained over a specific period by the monitoring system for a particular metric.
- `dumphealthdata`: The `dumphealthdata` command for a particular device displays the values of health checks obtained over a specific period by the monitoring system for a particular health check.
- `dumpstatistics`: The `dumpstatistics` command for a particular device displays statistics obtained over a specific period by the monitoring system for a particular metric item in the list of checks.

Using The `dumpmetricdata` And `dumphealthdata` Commands

A concise overview of the `dumpmetricdata` or `dumphealthdata` commands can be displayed by, for example, typing in "`help dumpmetricdata`" in the `device` mode of `cmsh`.

The usage of the `dumpmetricdata` and `dumphealthdata` commands is:

```
dumpmetricdata [OPTIONS] <start-time> <end-time> <metric> [device]
```

The mandatory arguments for the times, the metric being dumped, and the device or devices being sampled, have values that are specified as follows:

- The metric item `<metric>` for which the data values are being gathered must always be given. Metrics currently in use can conveniently be listed by running the `metrics` command (section 11.8.1).
- If `[device]` is not specified when running the `dumpmetricdata` or `dumphealthdata` command, then it must either be set by specifying the object from the `device` mode of `cmsh` (for example, with "`use node001`"), or by specifying it in the options. The options allow specifying, amongst others, a list, a group, or a category of devices.
- The time pair `<start-time>` or `<end-time>` can conveniently be specified as follows:
  - `now`: That is, the time at which the `dumpmetricdata` or `dumphealthdata` command is run
  - for only one item in the time pair, its time can be set relative to the other (the fixed time item). The non-fixed time item (the relative time item) is specified as follows:
    - A `<start-time>` <*number*> value is prefixed with a "-"
    - an `<end-time>` <*number*> value is prefixed with "+"
    - The <*number*> values have suffix values indicating the units of time, as seconds (s), minutes (m), hours (h), or days (d).

    This is summarized in the following table:

Unit	`<start-time>`	`<end-time>`
seconds:	`-<`*number*`>s`	`+<`*number*`>s`
minutes:	`-<`*number*`>m`	`+<`*number*`>m`
hours:	`-<`*number*`>h`	`+<`*number*`>h`
days:	`-<`*number*`>d`	`+<`*number*`>d`

  - YY/MM/DD
  - HH:MM
  - HH:MM:SS
  - Unix epoch time (seconds since 00:00:00 1 January 1970)

An example of how the preceding mandatory arguments to the `dumpmetricdata` command might be used is:

Example

```
[ddnmon61->device]% dumpmetricdata  -10m now cpuidle node001
# From Thu Oct 20 15:32:41 2011 to Thu Oct 20 15:42:41 2011
Time                       Value
-------------------------- -------
Thu Oct 20 15:32:41 2011   96.3481
Thu Oct 20 15:34:00 2011   95.3167
Thu Oct 20 15:36:00 2011   93.4167
Thu Oct 20 15:38:00 2011   93.7417
Thu Oct 20 15:40:00 2011   92.2417
Thu Oct 20 15:42:00 2011   93.5083
```

The options applied to the samples are specified as follows:

Option	Argument(s)	Description
`-n, --nodes`	*<list>*	for list of nodes
`-g, --groups`	*<list>*	for list of groups
`-c, --categories`	*<list>*	for list of categories
`-r, --racks`	*<list>*	for list of racks
`-h, --chassis`	*<list>*	for list of chassis
`-s, --state`	*<states>*	for nodes in state
`-i, --intervals`	*<number>*	number of samples to show
`-k, --kind`	`average, min, max`	show the average (default), minimum, or maximum of set of stored values, out of the list of device samples
`-m, --sum`		sum over specified devices
`-u, --unix`		use a Unix timestamp instead of using the default date format
`-d, --delimiter`	"*<string>*"	set the delimiter to a character
`-v, --verbose`		show the rest of the line on a new line instead of cutting it off

The `-s|--state` option selects only for nodes in that state. States are explained in section 6.5.

The `-i|--intervals` option interpolates the data that is to be displayed to a specified number *<number>* of interpolated samples over the given time range. Using "`-i 0`" outputs only the non-interpolated stored samples—the raw data—and is the default.

Example

```
[ddnmon61->device]% dumpmetricdata -i 0 -38m now loadone node001
# From Thu Oct 20 17:22:12 2011 to Thu Oct 20 17:59:12 2011
Time                       Value
-------------------------- -----
Thu Oct 20 17:22:12 2011   0
Thu Oct 20 17:48:00 2011   0
Thu Oct 20 17:50:00 2011   0.03
Thu Oct 20 17:52:00 2011   0.02
Thu Oct 20 17:54:00 2011   0.08
Thu Oct 20 17:56:00 2011   0.07
Thu Oct 20 17:58:00 2011   0.69
```

```
[ddnmon61->device]% dumpmetricdata -n node001..node002 -5m now cpuidle
# From Fri Oct 21 16:52:41 2011 to Fri Oct 21 16:57:41 2011
Time                         Value
-------------------------- -------
node001
Fri Oct 21 16:52:41 2011   99.174
Fri Oct 21 16:54:00 2011   99.3167
Fri Oct 21 16:56:00 2011   99.1333
node002
Fri Oct 21 16:52:41 2011   98.1518
Fri Oct 21 16:54:00 2011   99.3917
Fri Oct 21 16:56:00 2011   99.1417
[ddnmon61->device]%
```

When a sample measurement is carried out, if the sample has the same value as the two preceding it in the records, then the "middle" sample is discarded from storage for performance reasons.

Thus, when viewing the sequence of output of non-interpolated samples, identical values do not exceed two entries one after the other.

The `-m|--sum` option sums a specified metric for specified nodes, for a set of specified times. For 2 nodes, over a period 2 hours ago, with values interpolated over 3 time intervals, the option can be used as follows:

Example

```
[ddnmon61->device]% dumpmetricdata -2h now -i 3 loadone -n node00[1-2]\
 --sum
Time                         Value
-------------------------- --------
Thu Sep 19 14:54:19 2013   1.330446
Thu Sep 19 15:34:19 2013   1.944914
Thu Sep 19 16:14:19 2013   2.529622
```

Each entry in the values column in the preceding table is the sum of `loadone` displayed by `node001`, and by `node002`, at that time, as can be seen from the following corresponding table:

Example

```
[ddnmon61->device]% dumpmetricdata -2h now -i 3 loadone -n node00[1-2]
Device       Time                     Value
------------ ------------------------ --------
node001      Thu Sep 19 14:54:19 2013 0.715523
node001      Thu Sep 19 15:34:19 2013 1.03349
node001      Thu Sep 19 16:14:19 2013 1.264811
node002      Thu Sep 19 14:54:19 2013 0.614923
node002      Thu Sep 19 15:34:19 2013 0.911424
node002      Thu Sep 19 16:14:19 2013 1.264811
```

Each `loadone` value shown by a node at a time shown in the preceding table, is in turn an average interpolated value, based on actual data values sampled for that node around that time.

Using The `dumpstatistics` Commands

The usage of the `dumpstatistics` command is:

```
dumpstatistics [OPTIONS] <start-time> <end-time> <metric> [device]
```

and it follows the pattern of earlier (page 453) for the `dumpmetricdata` and `dumphealthdata` commands.

There are two significant differences:

- The `-p` option sets percentile boundaries. By default, the statistics are divided into percentiles of 25%.
- The `-i` option is set by default to a value of 20. A setting of `i 0` (which displays non-interpolated values when used with `dumpmetricdata` and `dumphealthdata`) is not a valid value for `dumpstatistics`.

Example

```
[ddnmon61->device]% dumpstatistics -i 5 -1h -p 100 now loadone -n node00\
1..node003
Start                      End                        100%
-------------------------- -------------------------- ------
Fri Oct 21 17:04:30 2011   Fri Oct 21 17:16:30 2011   0.11
Fri Oct 21 17:16:30 2011   Fri Oct 21 17:28:30 2011   0.08
Fri Oct 21 17:28:30 2011   Fri Oct 21 17:40:30 2011   0.09
Fri Oct 21 17:40:30 2011   Fri Oct 21 17:52:30 2011   0.27
Fri Oct 21 17:52:30 2011   Fri Oct 21 18:04:30 2011   0.22
[ddnmon61->device]% dumpstatistics -i 5 -u -1h now loadone -n node001..n\
ode003
Start      End        0%     25%    50%    75%    100%
---------- ---------- ------ ------ ------ ------ ------
1319444362 1319445082 0      0      0.03   0.0475 0.07
1319445082 1319445802 0      0      0      0.0025 0.04
1319445802 1319446522 0      0      0      0.01   0.06
1319446522 1319447242 0      0      0      0.01   0.08
1319447242 1319447962 0      0      0      0.015  0.05
```

11.8.3 The `check` Command For On-Demand Health Checks

The `latesthealthdata` command (page 11.8.1) displays the results from the latest health checks that have been run by the cluster. However, an administrator can also run a health check on demand, by using the `check` command. With it, a particular health check that is to be run can be specified for a range of devices:

Example

```
[ddnmon61->device]% check -n node001..node002 ib
Device       Health Check       Value            Age (sec.)Info Message
----------------------------------------------------------------------
node001      ib                 no data          0         Node down
node002      ib                 PASS             0
```

All the health checks can be run as follows:

Example

```
check -n node001..node002 *
[ddnmon61->device]% check -n node001..node002 *
Device       Health Check            Value       Age (sec.)Info Message
---------------------------------------------------------------------
```

```
node001     ManagedServicesOk      no data    1          Node down
node001     mounts                 no data    1          Node down
node001     rogueprocess           no data    1          Node down
node001     smart                  no data    1          Node down
node001     interfaces             no data    1          Node down
node001     dmesg                  no data    1          Node down
node001     ib                     no data    1          Node down
node001     diskspace:2% 10% 20%   no data    1          Node down
node001     ntp                    no data    1          Node down
node001     schedulers             no data    1          Node down
node002     ManagedServicesOk      PASS       1
node002     mounts                 PASS       1
node002     rogueprocess           PASS       1
node002     smart                  PASS       1
node002     interfaces             PASS       1
node002     dmesg                  PASS       1
node002     ib                     PASS       1
node002     diskspace:2% 10% 20%   PASS       1
node002     ntp                    PASS       1
node002     schedulers             PASS       1
```

11.9 The User Portal

11.9.1 Accessing The User Portal

The user portal is located by default on the head node. For a head node `ddnmon61`, it is accessible to users for a login via a browser at the URL `http://ddnmon61`. The state of the cluster can then be viewed by the users via a read-only interface.

The first time a browser is used to login to the portal, a warning about the site certificate being untrusted appears.

The certificate is a self-signed certificate (the X509v3 certificate of section 4.1), generated and signed by DDN, and the attributes of the cluster owner are part of the certificate. However, DDN is not a recognized Certificate Authority (CA) like the CAs that are recognized by a browser, which is why the warning appears.

Portals Isolated From The Outside

For a portal that is not accessible from the outside world, such as the internet, this warning is not an issue, and the user can simply accept the "untrusted" certificate.

Indeed, for such portals, the administrator may wish to do away with user authentication for the portal as well. A way to bypass authentication in such a case is to change the file `/var/www/html/userportal/header.php` so that one line is added as indicated:

```
...
Session()->set("username", "cmsupport"); // THIS LINE IS ADDED
if(!Session()->get("username")) {
$current = Input()->getCurrentPage();
$current = urlencode($current);
redirect("login.php?redirect=$current");
}
...
```

For standard logins, the type of authentication protocol used (for example, Unix, LDAP, Kerberos) is defined using PAM in /etc/pam.d/php.

Portals Accessible From The Outside

For a portal that is accessible via the internet, some administrators may regard it as more secure to ask users to trust the self-signed certificate rather than external certificate authorities.

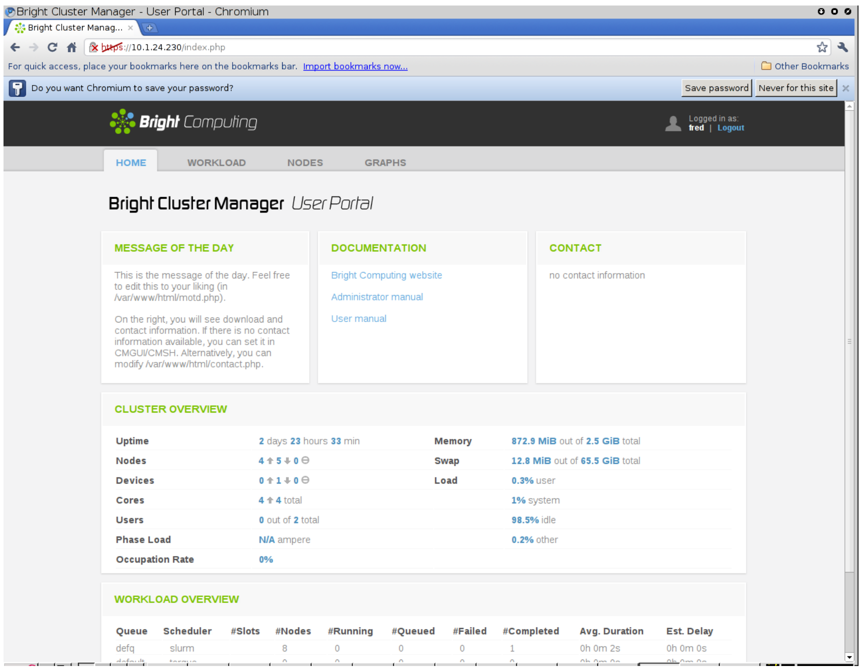
Alternatively the administrator can replace the self-signed certificate with one obtained by a standard CA, if that is preferred.

11.9.2 Disabling The User Portal

Only the user portal uses the apache2 web server running on the head node on the default port 80. If the user portal is not needed, the web server can be disabled or removed from the head node.

11.9.3 User Portal Home Page

The default user portal home page allows a quick glance to convey the most important cluster-related information for users (figure 11.38):



For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

Do you want Chromium to save your password?

Bright Cluster Manager *User Portal*

MESSAGE OF THE DAY

This is the message of the day. Feel free to edit this to your liking (in /var/www/html/motd.php).

On the right, you will see download and contact information. If there is no contact information available, you can set it in CMGUI/CMSH. Alternatively, you can modify /var/www/html/contact.php.

DOCUMENTATION

Bright Computing website
Administrator manual
User manual

CONTACT

no contact information

CLUSTER OVERVIEW

Uptime	2 days 23 hours 33 min	Memory	872.9 MiB out of 2.5 GiB total
Nodes	4 ↑ 5 ↓ 0 ⊖	Swap	12.8 MiB out of 65.5 GiB total
Devices	0 ↑ 1 ↓ 0 ⊖	Load	0.3% user
Cores	4 ↑ 4 total		1% system
Users	0 out of 2 total		98.5% idle
Phase Load	N/A ampere		0.2% other
Occupation Rate	0%		

WORKLOAD OVERVIEW

Queue	Scheduler	#Slots	#Nodes	#Running	#Queued	#Failed	#Completed	Avg. Duration	Est. Delay
defq	slurm		8	0	0	0	1	0h 0m 2s	0h 0m 0s

Figure 11.38: User Portal: Default Home Page

The following items are displayed on the home page:

- a Message Of The Day. This can be edited in /var/www/html/userportal/motd.php
- links to the documentation for the cluster
- contact information. This can be edited in /var/www/html/userportal/contact.php
- an overview of the cluster state, carried out by displaying the values of items in the cluster. These are a subset of the items seen in the

`Overview` tab of the cluster in `cmgui` in figure 3.4

- a workload overview. This is a table displaying a summary of queues and their associated jobs

The user portal from the point of view of a user is described further in the *User Manual*.

12 Day-to-day Administration

This chapter discusses several tasks that may come up in day-to-day administration of a cluster running DirectMon.

12.1 Parallel Shell

The cluster management tools include the parallel shell execution command, `pexec`. The `pexec` command can be run from within the OS shell (`bash` by default), or from within CMDaemon (i.e., the `cmsh` or `cmgui` front ends). The name `pexec` is the same in both these cases, but the syntax differs slightly.

A one-time execution of `pexec` from within `bash` or from within CMDaemon can run one or more `bash` commands on a group of nodes.

The CMDaemon `pexec` commands run in parallel.

The OS shell `pexec` commands, however, run on the nodes sequentially by default, waiting for the output from one node before moving on to the next. If the OS shell `pexec` is run with the background execution option (`-b`), then the `bash` commands are executed in parallel. Running in parallel is not done by default, because it could be risky for some commands, such as power-cycling nodes with a `reboot`, which may put unacceptable surge demands on the power supplies.

For example: Within `cmsh` or `cmgui`, the execution of a `power reset` command from `device` mode to power cycle a properly-configured group of nodes is safe, due to safeguards in CMDaemon to prevent nodes powering up too soon after each other (section 5.2). However, running the power-cycle command with `pexec` from the OS shell or within CMDaemon's `cmsh/cmgui` could, due to a similar lack of safeguard, cause a surge in the power demand that the infrastructure may not be able to cope with. For this reason, some care should be taken when using `pexec` in any form, and `pexec` from `bash` is generally regarded as a legacy application.

The only time that running `pexec` from `bash` is currently required instead of running `pexec` from within `cmsh/cmgui`, is when stopping and restarting CMDaemon on a large number of regular nodes (section 3.6.1). This is because a command to stop CMDaemon on a regular node, that itself relies on being run from a running CMDaemon on a regular node, can obviously give unexpected results.

12.1.1 `pexec` In The OS Shell

In the OS shell, running `pexec` without any arguments or options displays the following help text:

```
------------------------------------------------------------------------
Cluster commands v1.3

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

Options:
  \*                      use wildcard for file selection
  --                      end of parallel commandline options
  -v                      verbose
  -a                      do not broadcast to see which nodes are alive
  -b                      background execution (not supported by pcopy)
  -t=time(s)              timeout in seconds, default is 10 seconds, 0
                          disables timeout
  -d=directory            change the working directory
  -g=nodegroup            group selection
  -n=node                 single node selection
  -n=node001,node010      node list selection (node001 and node010)
  -n=node001..node010     node range selection (node001 through node010)
  -c                      prevent colored output
  -e                      only execute on nodes running the installer
  -m                      execute on all nodes, including nodes running
                          the installer
  -u=url                  alternative master CMDaemon URL

Commands:
  pexec <command>         execute <command>

  pcopy <f1..fn> <dir>    copy files and/or directories to <dir>. Add '/'
                          to the end of a directory in order to
                          transfer only the contents of a directory

  pkilluser               as root: user-only command, it should not be
                                   used by root
                          as user: kill processes owned by user

------------------------/////////////////////////---------------------
```

Some of the less obvious options are explained here:

- `\*`: Not actually an option, but a reminder to escape the asterisk from the OS shell if using wildcards. The following are equivalent:

  Example

  ```
  pexec ls "*"
  pexec ls \*
  ```

- `--`: After the list of parallel commands is run, `--` is used as a marker to indicate the `pexec` command has ended, and that anything that continues on the same line is no longer part of the `pexec` just ended. Thus, in the following command:

```
pexec ls -- ; ls
```

it means that the first `ls` is executed in parallel as part of `pexec`, while the second `ls` is run only from the current OS session.

- `-a`: By default a broadcast ping is used to check what nodes are alive. The `-a` option tries to run the command on all nodes (head and regular) without running the check.
- `-u=url`: This can be useful in the odd case of CMDaemon not running on the default ports. Appendix C includes a description of the port directives which allow non-default ports to be used.

12.1.2 `pexec` In `cmsh`

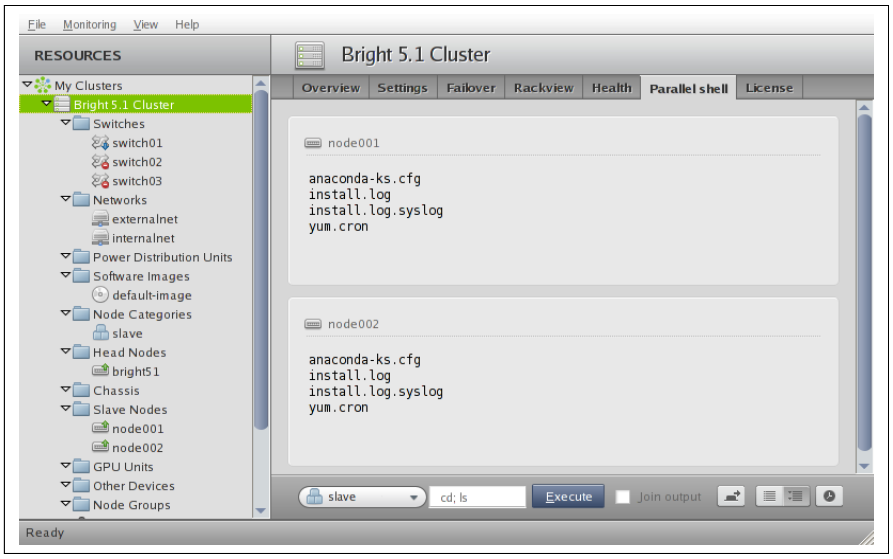
In `cmsh`, the `pexec` command is run from `device` mode:

Example

```
[ddnmon61->device]% pexec -n node001,node002 "cd ; ls"

[node001] :
anaconda-ks.cfg
install.log
install.log.syslog


[node002] :
anaconda-ks.cfg
install.log
install.log.syslog
```

12.1.3 `pexec` In `cmgui`

In `cmgui`, it is executed from the `Parallel Shell` tab after selecting the cluster from the resource tree (figure 12.1):



```
node001

anaconda-ks.cfg
install.log
install.log.syslog
yum.cron

node002

anaconda-ks.cfg
install.log
install.log.syslog
yum.cron
```

Figure 12.1: Executing Parallel Shell Commands

For large numbers of nodes, rendering the node subpanes (little boxes) in the `Parallel shell` tabbed pane of figure 12.1 can take a long time. To improve the `cmgui` experience, ticking the checkbox for `Only user parallel shell single view (enable for large node counts)` in the `File` menu under `settings` speeds up the rendering significantly at the cost of removing the borders of the subpanes, while ticking the `Hide down` button in the `Parallel shell` tabbed pane hides further messages about nodes that are in the `DOWN` state (section 3.1.1). If the `Join output` checkbox is ticked, then output that is the same is grouped into the same subpane.

Running parallel shell commands from `cmsh` instead is faster in most cases due to less graphics rendering overhead.

12.1.4 Using The `-j|-join` Option Of `pexec`

The output of the `pexec` command by default can come out in a sequence depending on node response time. To make it more useful for an administrator, order can be imposed on the output. Checking consistency across nodes is then trivial. For example, to see if all nodes have the same mounts on a cluster with 10 nodes, of which `node002` is down:

Example

```
[ddnmon61->device]% pexec -j -c default "mount|sort"
Nodes down:          node002
[node002]
Node down

[node001,node003..node010]
/dev/hda1 on / type ext3 (rw,noatime,nodiratime)
/dev/hda2 on /var type ext3 (rw,noatime,nodiratime)
/dev/hda3 on /tmp type ext3 (rw,nosuid,nodev,noatime,nodiratime)
/dev/hda6 on /local type ext3 (rw,noatime,nodiratime)
master:/cm/shared on /cm/shared type nfs
(rw,rsize=32768,wsize=32768,hard,intr,addr=10.141.255.254)
master:/home on /home type nfs
(rw,rsize=32768,wsize=32768,hard,intr,addr=10.141.255.254)
none on /dev/pts type devpts (rw,gid=5,mode=620)
none on /dev/shm type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
none on /proc type proc (rw,nosuid)
none on /sys type sysfs (rw)
```

Here, the `-j` option joins up identical fields (like the standard unix text utility, `join`). Additional order is imposed by sorting the output of each mount command within `bash` before the `-j` option operates from `cmsh`. The `-c` option executes the command on the `default` category of nodes.

12.1.5 Other Parallel Commands

Besides `pexec`, CMDaemon has several other parallel commands:

pcopy: parallel copy
Synopsis:

`pcopy [OPTIONS]` *<file/directory>* `[`*<file/directory>*`...]` *<destination directory>*

pkill: parallel kill
Synopsis:
`pkill [OPTIONS]` *<tracker>* `[`*<tracker>* `... ]`

plist: List the parallel commands that are currently running, with their tracker ID
Synopsis:
`plist`

pping: ping nodes in parallel
Synopsis:
`pping [OPTIONS]`

pread: read a small text file in parallel
Synopsis:
`pread [OPTIONS]` *<file>*

psave: save a small text file in parallel
Synopsis:
`psave [OPTIONS]` *<file>* *<destination>*

pwait: wait for parallel commands that are running to complete
Synopsis:
`pwait [OPTIONS]` *<tracker>* `[`*<tracker>*`...]`

Details on these parallel commands, including examples, can be seen by executing the `help` command within the `device` mode of `cmsh` for a parallel command, *<pcommand>*, as follows:

```
[ddnmon61->device]%help <pcommand>
```

12.2 Getting Support With Cluster Manager Issues

DirectMon is constantly undergoing development. While the result is a robust and well-designed cluster manager, it is possible that the administrator may run into a feature that requires technical help from DDN or DDN's resellers. This section describes how to get support for such issues.

12.2.1 Support Via E-mail

Unless your reseller offers direct support, the primary means of support is to contact `support@ddn.com`. This opens up a trouble ticket. It is a good idea to try to use a clear subject header, since that is used as part of a reference tag as the ticket progresses. Also helpful is a good description of the issue.

The document "How To Ask Questions The Smart Way" at `http://www.catb.org/esr/faqs/smart-questions.html` is quite readable, and explains how to submit a good technical query on online forums. While the document is not entirely relevant to the support provided by DDN, it is true that many of the principles covered there are also useful when sending a query to DDN when trying to resolve an issue. A read through the document may therefore be helpful before submitting a query to DDN.

If the issue appears to be a bug, then a bug report should be submitted. It is helpful to include as many details as possible to ensure the development team is able to reproduce the apparent bug. The policy at DDN is to welcome such reports, to provide feedback to the reporter, and to resolve the problem.

If there is apparently no response from DDN over e-mail, checking the spam folder is advised. Mail providers have been known to flag mails as spam, even in mid-thread.

As a supplement to e-mail support, DDN also provides the `cm-diagnose` and the `request-remote-assistance` utilities to help resolve issues.

12.2.2 Reporting Cluster Manager Diagnostics With `cm-diagnose`

The diagnostic utility `cm-diagnose` is run from the head node and gathers data on the system that may be of use in diagnosing issues. To view its options, capabilities, and defaults, it can be run as "`cm-diagnose --help`". For particular issues it may be helpful to change some of the default values to gather data in a more targeted way.

When run without any options, it runs interactively, and allows the administrator to send the resultant diagnostics file to DDN directly. The output of a `cm-diagnose` session looks something like the following (the output has been made less verbose for easy viewing):

Example

```
[root@ddnmon61 ~]# cm-diagnose
To be able to contact you about the issue, please provide
your e-mail address (or support ticket number, if applicable):
franknfurter@example.com

To diagnose any issue, it would help if you could provide a description
about the issue at hand.
Do you want to enter such description? [Y/n]

End input with ctrl-d
I tried X, Y, and Z on the S2464 motherboard. When that didn't work, I
tried A, B, and C, but the florbish is grommicking.
Thank you.

If issues are suspected to be in the cmdaemon process, a gdb trace of
that process is useful.
In general such a trace is only needed if DDN Support asks for this.
Do you want to create a gdb trace of the running CMDaemon? [y/N]

Proceed to collect information? [Y/n]

Processing master
    Processing commands
    Processing file contents
    Processing large files and log files
    Collecting process information for CMDaemon
    Executing CMSH commands
    Finished executing CMSH commands

Processing default-image
    Processing commands
    Processing file contents

Creating log file: /root/ddnmon61_609.tar.gz

Cleaning up

Automatically submit diagnostics to
http://support.ddn.com/cm-diagnose/ ? [Y/n] y

Uploaded file: ddnmon61_609.tar.gz
Remove log file (/root/ddnmon61_609.tar.gz)?  [y/N] y
[root@ddnmon61 ~]
```

12.2.3 Requesting Remote Support With `request-remote-assistance`

Some problems, if handled over e-mail support, may require a lengthy and time-consuming e-mail exchange between the cluster administrator and DDN before resolution. The `request-remote-assistance` utility speeds up resolution by avoiding this exchange, and is generally recommended to obtain the quickest resolution for such problems.

The `request-remote-assistance` utility allows a DDN engineer to securely tunnel into the cluster without a change in firewall or `ssh` settings of the cluster.

With `request-remote-assistance`:

- It must be allowed to access the `www` and `ssh` ports of DDN's internet servers.
- For some problems, the engineer may wish to power cycle a node. In that case, indicating what node the engineer can power cycle should be added to the option for entering additional information.
- Administrators familiar with `screen` may wish to run it within a screen session and detach it.

It is run as follows:

Example

```
[root@ddnmon61 ~]# request-remote-assistance

This tool helps securely set up a temporary ssh tunnel to
sandbox.ddn.com.

Allow a DDN engineer ssh access to the cluster? [Y/n]

Enter additional information for DDN (eg: related
ticket number, problem description)? [Y/n]
```

```
End input with ctrl-d
Ticket 1337 - the florbish is grommicking

Thank you.

Added temporary DDN public key.
```

After the administrator has responded to the `Enter additional information...` entry, and has typed in the `ctrl-d`, the utility tries to establish the connection. The screen clears, and the secure tunnel opens up, displaying the following notice:

```
REMOTE ASSISTANCE REQUEST
########################################################
A connection has been opened to DDN Support.
Closing this window will terminate the remote assistance
session.
--------------------------------------------------------

Hostname: ddnmon61.NOFQDN
Connected on port: 7000

ctrl-c to terminate this session
```

DDN support automatically receives an e-mail alert that an engineer can now securely tunnel into the cluster. The session activity is not explicitly visible to the administrator. When the engineer has ended the session, the administrator may remove the secure tunnel with a `ctrl-c`, and the display then shows:

```
Tunnel to sandbox.ddn.com terminated.
Removed temporary DDN public key.
[root@ddnmon61 ~]#
```

The DDN engineer is then no longer able to access the cluster.

12.3 Backups

12.3.1 Cluster Installation Backup

DirectMon does not include facilities to create backups of a cluster installation. When setting up a backup mechanism, it is recommended that the full file-system of the head node (i.e. including all software images) is backed up. Unless the node hard drives are used to store important data, it is not necessary to back up nodes.

If no backup infrastructure is already in place at the cluster site, the following open source (GPL) software packages may be used to maintain regular backups:

- **Bacula**: Bacula is a mature network based backup program that can be used to backup to a remote storage location. If desired, it is also possible to use Bacula on nodes to back up relevant data that is stored on the local hard drives. More information is available at `http://www.bacula.org`

- **rsnapshot**: rsnapshot allows periodic incremental file system snapshots to be written to a local or remote file system. Despite its simplicity, it can be a very effective tool to maintain frequent backups of a system. More information is available at `http://www.rsnapshot.org`.

12.3.2 Local Database Backups And Restoration

The CMDaemon database is stored in the MySQL `cmdaemon` database, and contains most of the stored settings of the cluster.

Other databases stored separately in MySQL are:

- The CMDaemon monitoring database `cmdaemon_mon`
- The Slurm account database `slurm_acct_db`

The administrator is expected to run a regular backup mechanism for the cluster to allow restores of all files from a recent snapshot. As an additional, separate, convenience:

- For the CMDaemon database, the entire database is also backed up nightly on the cluster filesystem itself ("local rotating backup") for the last 7 days.
- For the monitoring database, the monitoring database raw data records are not backed up locally, since these can get very large, but the rest of the database is backed up for the last 7 days too.
- The Slurm account database is not backed up[1].

Database Corruption Messages And Repairs

A corrupted MySQL database is commonly caused by an improper shutdown of the node. To deal with this, when starting up, MySQL checks itself for corrupted tables, and tries to repair any such by itself. Detected corruption causes an event notice to be sent to `cmgui` or `cmsh`.

When there is database corruption, info messages in the `/var/log/cmdaemon` log may mention:

- "`Unexpected eof found`" in association with a table in the database,
- "`can't find file`" when referring to an entire missing table,
- locked tables,
- error numbers from table handlers,
- "`Error while executing`" a command.

An example of an event message on a head node `icarus`:

Example

[1]Actually, more precisely, for failover configurations, there is an automated synchronization of all databases between the active and passive head nodes that allows failover to work. So in that sense the Slurm account database is "backed up".

```
[icarus]%
Fri Jan 18 10:33:15 2012 [notice] icarus: Error when reading data from m\
onitoring database. No monitoring data will be saved. (details: Incorrec\
t key file for table './cmdaemon_mon/MonData.MYI'; try to repair it)
[icarus]%
```

The associated messages logged in `/var/log/cmdaemon` may show something like:

Example

```
Jan 18 10:31:05 icarus CMDaemon: Info: Reconnect command processed.
Jan 18 10:31:05 icarus CMDaemon: Info: MysqlBuffer: starting main mysql \
buffer thread
Jan 18 10:31:05 icarus CMDaemon: Info: MysqlBuffer: starting mirroring t\
hread
Jan 18 10:31:05 icarus CMDaemon: Info: Preloading mysql key cache using \
query: LOAD INDEX INTO CACHE MonMetaData
Jan 18 10:31:05 icarus CMDaemon: Info: MysqlBuffer: starting MysqlBuffer\
 thread
Jan 18 10:31:05 icarus CMDaemon: Info: Database: Mirroring required to r\
remote master
Jan 18 10:31:05 icarus CMDaemon: Info: Database: generalQuery returned: \
cmdaemon_mon.MonMetaData preload_keys status OK
Jan 18 10:31:05 icarus CMDaemon: Info: Preloading mysql key cache using \
query: LOAD INDEX INTO CACHE MonData INDEX (MonDataIndex) IGNORE LEAVES
Jan 18 10:31:05 icarus CMDaemon: Info: Database: generalQuery returned: \
cmdaemon_mon.MonData preload_keys Error Unexpected eof found when readin\
g file '/var/lib/mysql/cmdaemon_mon/MonData.MYI' (Errcode: 0) cmdaemon_m\
on.MonData preload_keys status OK
```

Looking through the preceding messages, the conclusion is that the monitoring database has a corrupted MonData table. Being in MyISAM format (.MYI) means the `myisamchk` repair tool can be run on the table, for example as:

```
[root@icarus ~]# service cmd stop
[root@icarus ~]# myisamchk --recover /var/lib/mysql/cmdaemon_mon/MonData\
.MYI
[root@icarus ~]# service cmd start
```

If basic repair fails, more extreme repair options—`man myisamchk(1)` suggests what—can then be tried out.

Another example: If CMDaemon is unable to start up due to a corrupted database, then messages in the `/var/log/cmdaemon` file might show something like:

Example

```
Oct 11 15:48:19 solaris CMDaemon: Info: Initialize cmdaemon database
Oct 11 15:48:19 solaris CMDaemon: Info: Attempt to set provisioningNetwo\
rk (280374976710700) not an element of networks
Oct 11 15:48:19 solaris CMDaemon: Fatal: Database corruption! Load Maste\
rNode with key: 280374976782569
Oct 11 15:48:20 solaris CMDaemon: Info: Sending reconnect command to all\
 nodes which were up before master went down ...
Oct 11 15:48:26 solaris CMDaemon: Info: Reconnect command processed.
```

Here it is CMDaemon's "`Database corruption`" message that the administrator should be aware of, and which suggests database repairs are required for the CMDaemon database. The severity of the corruption, in this case not even allowing CMDaemon to start up, may mean a restoration from backup is needed. How to restore from backup is explained in the next section.

Restoring From The Local Backup

If the MySQL database repair tools of the previous section do not fix the problem, then, for a failover configuration, the `dbreclone` option (section 15.4.2) should normally provide a CMDaemon database that is current. The option also clones the Slurm database. It does not clone the monitoring database.

If the head node is not part of a failover configuration, then a restoration from local backup can be done. The local backup directory is `/var/spool/cmd/backup`, with contents that look like (some text elided):

Example

```
[root@solaris ~]# cd /var/spool/cmd/backup/
[root@solaris backup]# ls -l
total 280
...
-rw------- 1 root root  1593 Oct 10 04:02 backup-monitor-Mon.sql.gz
-rw------- 1 root root  1593 Oct  9 04:02 backup-monitor-Sun.sql.gz
...
-rw------- 1 root root 33804 Oct 10 04:02 backup-Mon.sql.gz
-rw------- 1 root root 33805 Oct  9 04:02 backup-Sun.sql.gz
-rw------- 1 root root 33805 Oct 11 04:02 backup-Tue.sql.gz
...
```

The CMDaemon database snapshots are stored as `backup-<`*`day of week`*`>.sql.gz` while the monitoring database snapshots are stored as `backup-monitor-<`*`day of week`*`>.sql.gz`. In the example, the latest backup available in the listing for CMDaemon turns out to be `backup-Tue.sql.gz`

The latest backup can then be ungzipped and piped into the MySQL database for the user `cmdaemon`. The password, <*password*>, can be retrieved from `/cm/local/apps/cmd/etc/cmd.conf`, where it is configured in the `DBPass` directive (Appendix C).

Example

```
gunzip backup-Tue.sql.gz
service cmd stop #(just to make sure)
mysql -ucmdaemon -p<password> cmdaemon < backup-Tue.sql
```

Running "`service cmd start`" should have CMDaemon running again, this time with a restored database from the time the snapshot was taken. That means, that any changes that were done to the cluster manager after the time the snapshot was taken are no longer implemented.

A similar procedure for monitoring database backups can be used to restore the table structure and info messages of the data records from local backup, which means that messages and data record schema changes after the snapshot time are gone. Restoring the monitoring database from the backup only restores the schema and info messages of the data records. It does not restore the monitoring data records themselves. To restore records, the administrator would need to retrieve a snapshot from the regular backup that the administrator should be running for the cluster.

12.4 BIOS Configuration And Updates

DirectMon includes a number of tools that can be used to configure and update the BIOS of nodes. All tools are located in the `/cm/shared/apps/cmbios/nodebios` directory on the head node. The remainder of this section assumes that this directory is the current working directory.

Due to the nature of BIOS updates, it is highly recommended that these tools are used with great care. Incorrect use may render nodes unusable.

Updating a BIOS of a node requires booting it from the network using a specially prepared DOS image. From the `autoexec.bat` file, one or multiple automated BIOS operations can be performed.

12.4.1 BIOS Configuration

In order to configure the BIOS on a group of nodes, an administrator needs to manually configure the BIOS on a reference node using the conventional method of entering BIOS Setup mode at system boot time. After the BIOS has been configured, the machine needs to be booted as a node. The administrator may subsequently use the `cmospull` utility on the node to create a snapshot of the reference node's NVRAM contents.

Example

```
ssh node001 /cm/shared/apps/cmbios/nodebios/cmospull > node001.nvram
```

After the NVRAM settings of the reference node have been saved to a file, the settings need to be copied to the generic DOS image so that they can be written to the NVRAM of the other nodes.

The generic DOS image is located in `/cm/shared/apps/cmbios/nodebios/win98boot.img`. It is generally a good idea to copy the generic image and make changes to the copy only.

Example

```
cp -a win98boot.img flash.img
```

To modify the image, it is first mounted:

```
mount -o loop flash.img /mnt
```

When the DOS image has been mounted, the utility that writes out the NVRAM data needs to be combined with the NVRAM data into a single DOS executable. This is done by appending the NVRAM data to the `cmosprog.bin` file. The result is a DOS `.COM` executable.

Example

```
cat cmosprog.bin node001.nvram > cmosprog.com
```

The generated .COM is then copied to the image and should be started from the autoexec.bat file. Note that DOS text files require a carriage return at the end of every line.

Example

```
cp cmosprog.com /mnt
/bin/echo -e "A:\\\cmosprog.com\r" >> /mnt/autoexec.bat
```

After making the necessary changes to the DOS image, it is unmounted:

```
umount /mnt
```

After preparing the DOS image, it is booted as described in section 12.4.3.

12.4.2 Updating BIOS

Upgrading the BIOS to a new version involves using the DOS tools that were supplied with the BIOS. Similar to the instructions above, the flash tool and the BIOS image must be copied to the DOS image. The file autoexec.bat should be altered to invoke the flash utility with the correct parameters. In case of doubt, it can be useful to boot the DOS image and invoke the BIOS flash tool manually. Once the correct parameters have been determined, they can be added to the autoexec.bat.

After a BIOS upgrade, the contents of the NVRAM may no longer represent a valid BIOS configuration because different BIOS versions may store a configuration in different formats. It is therefore recommended to also write updated NVRAM settings immediately after flashing a BIOS image (section 12.4.1).

The next section describes how to boot the DOS image.

12.4.3 Booting DOS Image

To boot the DOS image over the network, it first needs to be copied to software image's /boot directory, and must be world-readable.

Example

```
cp flash.img /cm/images/default-image/boot/bios/flash.img
chmod 644 /cm/images/default-image/boot/bios/flash.img
```

An entry is added to the PXE boot menu to allow the DOS image to be selected. This can easily be achieved by modifying the contents of /cm/images/default-image/boot/bios/menu.conf, which is by default included automatically in the PXE menu. By default, one entry Example is included in the PXE menu, which is however invisible as a result of the MENU HIDE option. Removing the MENU HIDE line will make the BIOS flash option selectable. Optionally the LABEL and MENU LABEL may be set to an appropriate description.

The option MENU DEFAULT may be added to make the BIOS flash image the default boot option. This is convenient when flashing the BIOS of many nodes.

Example

```
LABEL FLASHBIOS
  KERNEL memdisk
  APPEND initrd=bios/flash.img
  MENU LABEL ^Flash BIOS
#  MENU HIDE
  MENU DEFAULT
```

The `bios/menu.conf` file may contain multiple entries corresponding to several DOS images to allow for flashing of multiple BIOS versions or configurations.

12.5 Hardware Match Check

Often a large number of identical nodes may be added to a cluster. In such a case it is a good practice to check that the hardware matches what is expected. This can be done easily as follows:

1. The new nodes, say node129 to node255, are committed to a newly created category `newnodes` as follows (output truncated):

   ```
   [root@ddnmon61 ~]# cmsh -c "category add newnodes; commit"
   [root@ddnmon61 ~]# for i in 129..255
   > do
   > cmsh -c "device; set node00$i category newnodes; commit"
   > done
   Successfully committed 1 Devices
   Successfully committed 1 Devices
   ```

2. The hardware profile of one of the new nodes, say node129, is saved into the category `newnodes`. This is done using the `node-hardware-profile` health check (Appendix H.2.1) as follows:

   ```
   [root@ddnmon61 ~]# /cm/local/apps/cmd/scripts/healthchecks/node-har\
   dware-profile -n node129 -s newnodes
   ```

   The profile is intended to be the reference hardware against which all the other nodes should match.

3. The frequency with which the health check should run in normal automated periodic use is set as follows (some prompt text elided):

   ```
   [root@ddnmon61 ~]# cmsh
   [ddnmon61]% monitoring setup healthconf newnodes
   [...->healthconf]% add hardware-profile
   [...->healthconf*[hardware-profile*]]% set checkinterval 600; commit
   ```

4. The `cmdaemon` then automatically alerts the administrator if one of the nodes does not match the hardware of that category during the first automated check. In the unlikely case that the reference node is itself faulty, then that will also be obvious because all—or almost all, if more nodes are faulty—of the other nodes in that category will then be reported "faulty" during the first check.

12.6 Serial Over LAN Console Access

Direct console access to nodes is not always possible. Other possibilities to access the node are:

1. **SSH access via an ssh client.** This requires that an `ssh` server runs on the node and that it is accessible via the network.
2. **Remote shell via CMDaemon.** This is possible if CMDaemon is running on the node and accessible via `cmgui` or `cmsh`.
   - For `cmgui`, for a node in the node settings resource, in the `Tasks` tabbed pane, clicking the `Root Shell` button launches an interactive `bash` session connected to the node via CMDaemon.
   - For `cmsh`, in `device` mode, running the command `rshell node001` launches an interactive bash session connected to node001 via CMDaemon.
3. **Connecting via a serial over LAN console.** If a serial console is configured then a serial over LAN (SOL) console can be accessed from `cmgui` (`Remote Console`) or from `cmsh` (`rconsole`).

Item 3 in the preceding list, SOL access, is very useful for administrators, and is covered next more thoroughly with:

- some background notes on serial over LAN console access (section 12.6.1)
- the configuration of SOL with `cmgui` (section 12.6.2)
- the configuration of SOL with `cmsh` (section 12.6.3)
- the `conman` SOL logger and viewer (section 12.6.4)

12.6.1 Background Notes On Serial Console And SOL

Serial ports are data ports that can usually be enabled or disabled for nodes in the BIOS.

If the serial port of a node is enabled, it can be configured in the node kernel to redirect a console to the port. The serial port can thus provide what is called serial console access. That is, the console can be viewed using a terminal software such as minicom (in Linux) or Hyperterminal (in Windows) on another machine to communicate with the node via the serial port, using a null-modem serial cable. This has traditionally been used by system administrators when remote access is otherwise disabled, for example if ssh access is not possible, or if the TCP/IP network parameters are not set up right.

While traditional serial port console access as just described can be useful, it is inconvenient, because of having to set arcane serial connection parameters, use the relatively slow serial port and use a special serial cable. Serial Over LAN (SOL) is a more recent development of serial port console access, which uses well-known TCP/IP networking over a faster Ethernet port, and uses a standard Ethernet cable. SOL is thus generally more convenient than traditional serial port console access. The serial port DE-9 or DB-25 connector and its associated chip may or may not physically still exist on servers that support SOL, but a serial port chip is nonetheless usually implied to exist in the BIOS, and can be "enabled" or "disabled" in the BIOS, thus enabling or disabling SOL.

SOL is a feature of the BMC (Baseboard Management Controller) for IPMI 2.0 and iLO. It is enabled by configuring the BMC BIOS. When enabled, data that is going to the BMC serial port is sent to the BMC LAN port. SOL clients can then process the LAN data to display the console. As far as the node kernel is concerned, the serial port is still just behaving like a serial port, so no change needs to be made in kernel configuration in doing whatever is traditionally done to configure serial connectivity. However, the console is now accessible to the administrator using the SOL client on the LAN.

SOL thus allows SOL clients on the LAN to access the Linux serial console if

1. SOL is enabled and configured in the BMC BIOS
2. the serial console is enabled and configured in the node kernel
3. the serial port is enabled and configured in the node BIOS

The BMC BIOS, node kernel, and node BIOS therefore all need to be configured to implement SOL console access.

Background Notes: BMC BIOS Configuration

The BMC BIOS SOL values are usually enabled and configured as a submenu or pop-up menu of the node BIOS. These settings must be manually made to match the values in DirectMon, or vice versa.

During a factory reset of the node, it is likely that a SOL configuration in the cluster manager will no longer match the configuration on the node BIOS after the node boots. This is because the cluster manager cannot configure these. This is in contrast to the IP address and user authentication settings of the BMC (section 4.8), which the cluster manager is able to configure on reboot.

Background Notes: Node Kernel Configuration

Sections 12.6.2 and 12.6.3 explain how SOL access configuration is set up for the node kernel using `cmgui` or `cmsh`. SOL access configuration on the node kernel is serial access configuration on the node kernel as far as the system administrator is concerned; the only difference is that the word "serial" is replaced by "SOL" in DirectMon's `cmgui` and `cmsh` front ends to give a cluster perspective on the configuration.

Background Notes: Node BIOS Configuration

Since BIOS implementations vary, and serial port access is linked with SOL access in various ways by the BIOS designers, it is not possible to give short and precise details on how to enable and configure them. The following rules-of-thumb, if followed carefully, should allow most BMCs to be configured for SOL access with DirectMon:

- Serial access, or remote access via serial ports, should be enabled in the BIOS, if such a setting exists.
- The node BIOS serial port settings should match the node configuration SOL settings (sections 12.6.2, 12.6.3). That means, items such

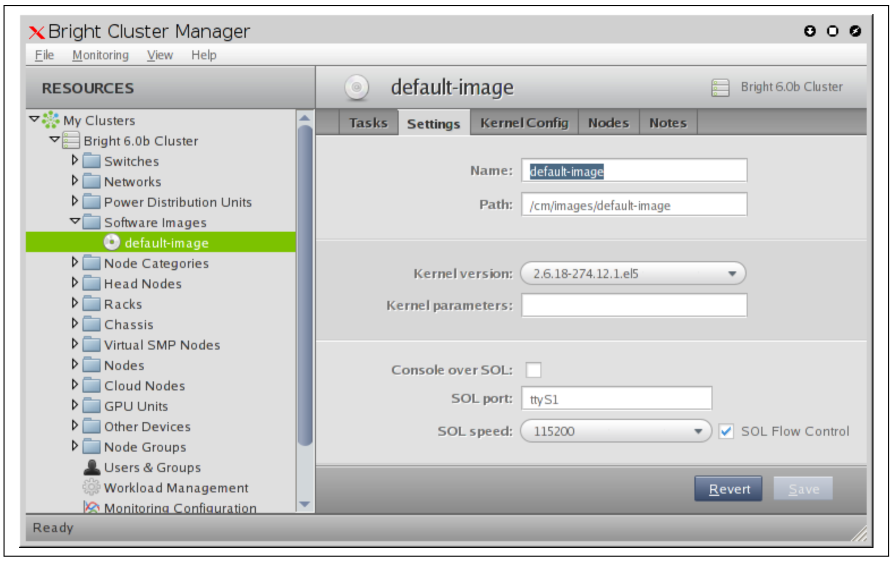
as "`SOL speed`", "`SOL Flow Control`", and "`SOL port`" in the node configuration must match the equivalent in the node BIOS. Reasonable values are

  - SOL speed: 115200bps. Higher speeds are sometimes possible, but are more likely to have problems.
  - SOL flow control: On. It is however unlikely to cause problems if flow control is off in both.
  - SOL port: COM1 (in the BIOS serial port configuration), corresponding to ttyS0 (in the node kernel serial port configuration). Alternatively, COM2, corresponding to ttyS1. Sometimes, the BIOS configuration display indicates SOL options with options such as: "`COM1 as SOL`", in which case such an option should be selected for SOL connectivity.
  - Terminal type: VT100 or ANSI.

- If there is an option for BIOS console redirection after BIOS POST, it should be disabled.
- If there is an option for BIOS console redirection before or during BIOS POST, it should be enabled.
- The administrator should be aware that the BMC LAN traffic, which includes SOL traffic, can typically run over a dedicated NIC or over a shared NIC. The choice of dedicated or shared is toggled, either in the BIOS, or via a physical toggle, or both. If BMC LAN traffic is configured to run on the shared NIC, then just connecting a SOL client with an Ethernet cable to the dedicated BMC NIC port shows no console.
- The node BIOS values should manually be made to match the values in DirectMon, or vice versa.

12.6.2 SOL Console Configuration And Access With `cmgui`

In `cmgui`, within the "`Software Images`" resource, if the "`Console over SOL`" checkbox (figure 12.2) is checked for the software image that the node is using, then the kernel option to make the Linux serial console accessible is used after the node is rebooted.

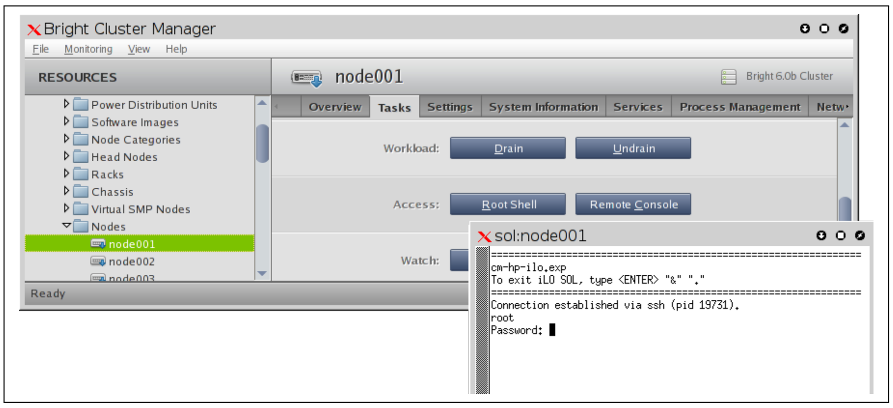


Name: default-image
Path: /cm/images/default-image
Kernel version: 2.6.18-274.12.1.el5
Kernel parameters:
Console over SOL: ☐
SOL port: ttyS1
SOL speed: 115200 ☑ SOL Flow Control

Figure 12.2: Configuring The SOL Console For A Node With `cmgui`

This means that if the serial port and SOL are enabled for the node hardware, then after the node reboots the Linux serial console is accessible over the LAN via an SOL client. Some of the settings for SOL can be set from the same screen.

With SOL correctly configured, an SOL client to access the Linux serial console can be launched using the "`Remote Console`" button for the node (figure 12.3).



```
=====================================================================
cm-hp-ilo.exp
To exit iLO SOL, type <ENTER> "&" "."
=====================================================================
Connection established via ssh (pid 19731).
root
Password: █
```

Figure 12.3: Accessing The SOL Console For A Node With `cmgui`

12.6.3 SOL Console Configuration And Access With `cmsh`

In `cmsh`, the serial console kernel option for a software image can be enabled within the `softwareimage` mode of `cmsh`. For the default image of `default-image`, this can be done as follows:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% softwareimage use default-image

[ddnmon61->softwareimage[default-image]]% set enablesol yes
[ddnmon61->softwareimage*[default-image*]]% commit
```

The SOL settings for a particular image can be seen with the `show` command:

```
[ddnmon61->softwareimage[default-image]]% show | grep SOL
Parameter                      Value
------------------------------ --------------
Enable SOL                     yes
SOL Flow Control               yes
SOL Port                       ttyS1
SOL Speed                      115200
```

Values can be adjusted if needed with the `set` command.

On rebooting the node, the new values are used.

To access a node via an SOL client, the node can be specified from within the `device` mode of `cmsh`, and the `rconsole` command run on the head node:

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% device use node001
[ddnmon61->device[node001]]% rconsole
```

12.6.4 The `conman` Serial Console Logger And Viewer

In DirectMon, the console viewer and logger service `conman` is used to connect to an SOL console and log the console output.

If "`Console over SOL`" in `cmgui` or `enablesol` in `cmsh` are enabled for the software image, then the `conman` configuration is written out and the `conman` service is started.

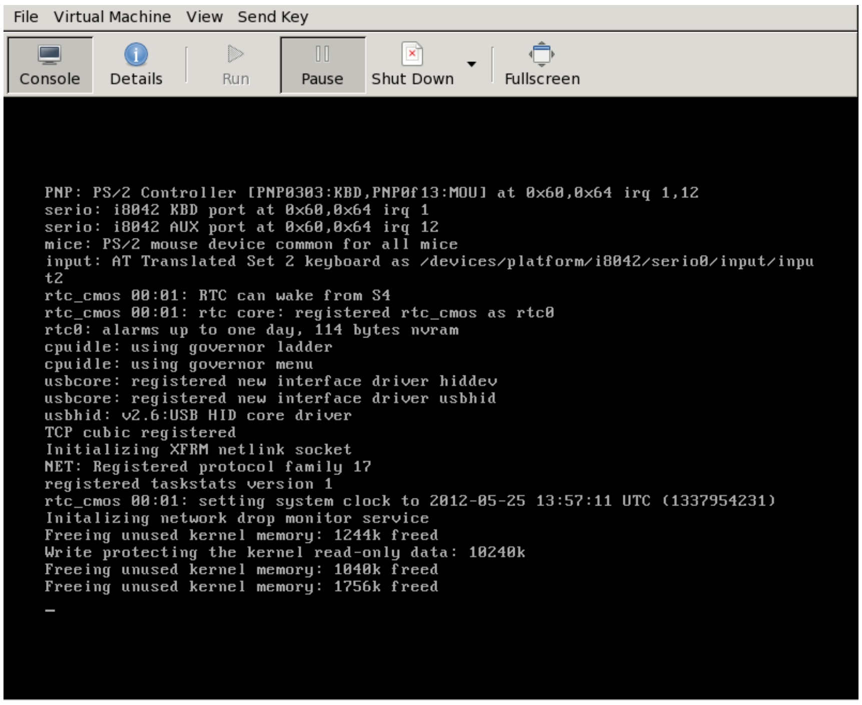
Logging The Serial Console

The data seen at the serial console is then logged via SOL to the head node after reboot. For each node that has logging enabled, a log file is kept on the head node. For example, for `node001` the log file would be at `/var/log/conman/node001.log`. To view the logged console output without destroying terminal settings, using `less` with the `-R` option is recommended, as in: `less -R /var/log/conman/node001.log`.

Using The Serial Console Interactively

Viewing quirk during boot: In contrast to the logs, the console viewer shows the initial booting stages of the node as it happens. There is however a quirk the system administrator should be aware of:

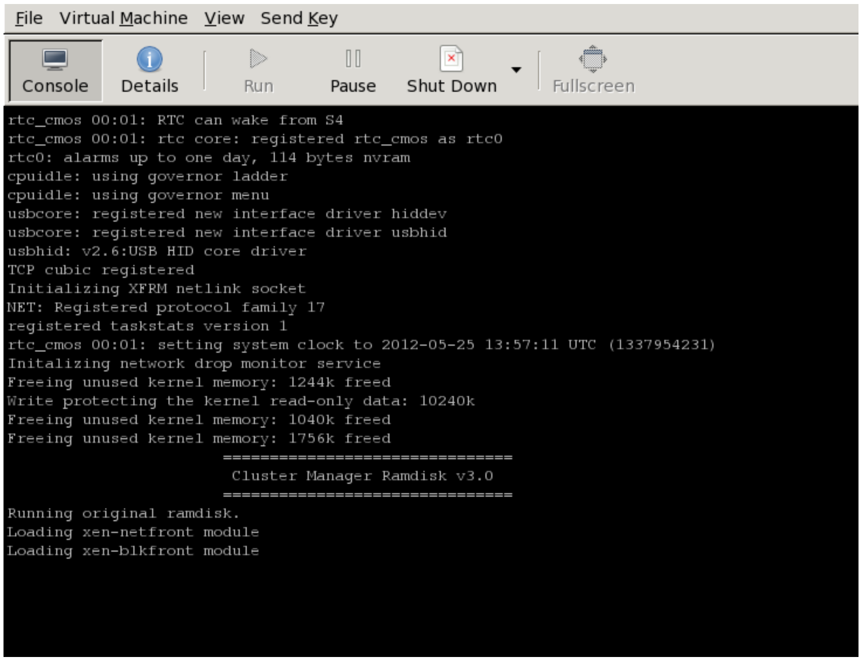
Normally the display on the physical console is a copy of the remote console. However, during boot, after the remote console has started up and been displaying the physical console for a while, the physical console display freezes. For the Linux 2.6 kernel series, the freeze occurs just before the ramdisk is run, and means that the display of the output of the launching `init.d` services is not seen on the physical console (figure 12.4).



```
PNP: PS/2 Controller [PNP0303:KBD,PNP0f13:MOU] at 0x60,0x64 irq 1,12
serio: i8042 KBD port at 0x60,0x64 irq 1
serio: i8042 AUX port at 0x60,0x64 irq 12
mice: PS/2 mouse device common for all mice
input: AT Translated Set 2 keyboard as /devices/platform/i8042/serio0/input/inpu
t2
rtc_cmos 00:01: RTC can wake from S4
rtc_cmos 00:01: rtc core: registered rtc_cmos as rtc0
rtc0: alarms up to one day, 114 bytes nvram
cpuidle: using governor ladder
cpuidle: using governor menu
usbcore: registered new interface driver hiddev
usbcore: registered new interface driver usbhid
usbhid: v2.6:USB HID core driver
TCP cubic registered
Initializing XFRM netlink socket
NET: Registered protocol family 17
registered taskstats version 1
rtc_cmos 00:01: setting system clock to 2012-05-25 13:57:11 UTC (1337954231)
Initalizing network drop monitor service
Freeing unused kernel memory: 1244k freed
Write protecting the kernel read-only data: 10240k
Freeing unused kernel memory: 1040k freed
Freeing unused kernel memory: 1756k freed
_
```

Figure 12.4: Physical Console Freeze During SOL Access

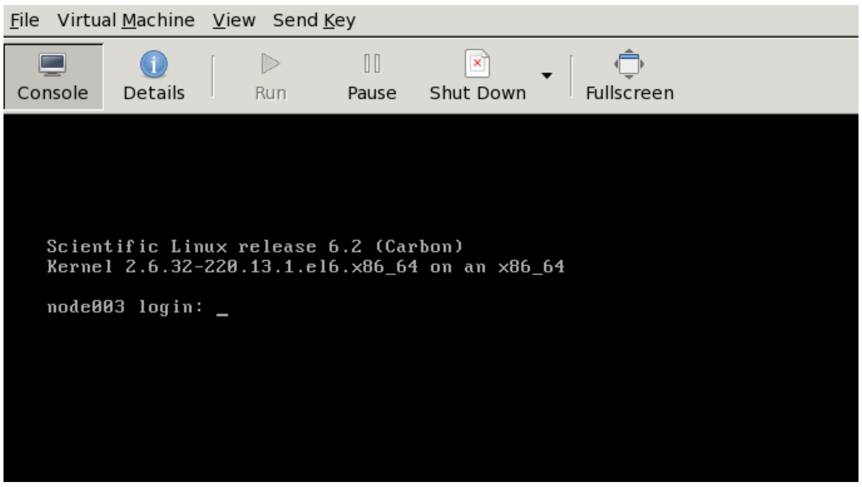
The freeze is only a freeze of the display, and should not be mistaken for a system freeze. It occurs because the kernel is configured during that stage to send to only one console, and that console is the remote console. The remote console continues to display its progress (figure 12.5) during the freeze of the physical console display.



```
rtc_cmos 00:01: RTC can wake from S4
rtc_cmos 00:01: rtc core: registered rtc_cmos as rtc0
rtc0: alarms up to one day, 114 bytes nvram
cpuidle: using governor ladder
cpuidle: using governor menu
usbcore: registered new interface driver hiddev
usbcore: registered new interface driver usbhid
usbhid: v2.6:USB HID core driver
TCP cubic registered
Initializing XFRM netlink socket
NET: Registered protocol family 17
registered taskstats version 1
rtc_cmos 00:01: setting system clock to 2012-05-25 13:57:11 UTC (1337954231)
Initalizing network drop monitor service
Freeing unused kernel memory: 1244k freed
Write protecting the kernel read-only data: 10240k
Freeing unused kernel memory: 1040k freed
Freeing unused kernel memory: 1756k freed
                             ===============================
                              Cluster Manager Ramdisk v3.0
                             ===============================
Running original ramdisk.
Loading xen-netfront module
Loading xen-blkfront module
```

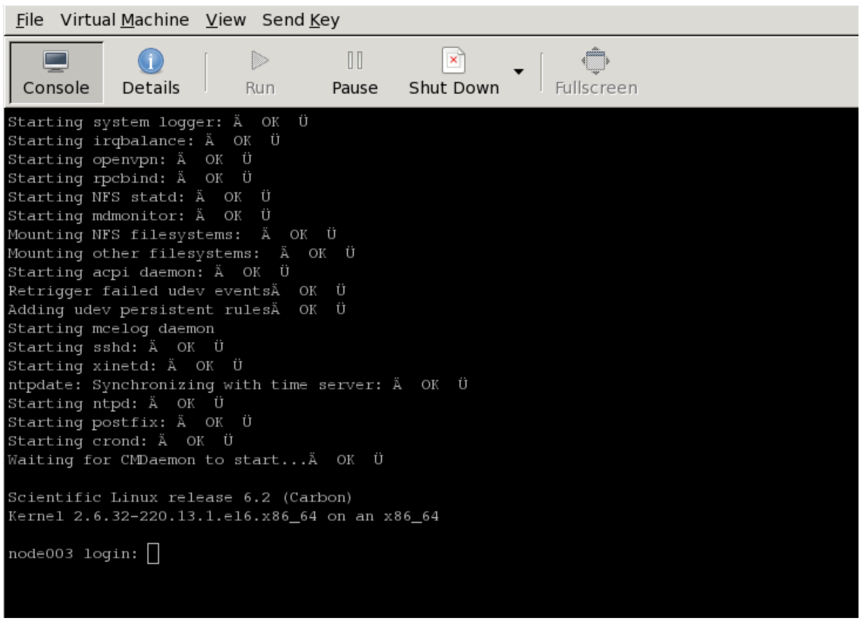
Figure 12.5: Remote Console Continues During SOL Access During Physical Console Freeze

Finally, just before login is displayed, the physical console once more (figure 12.6) starts to display what is on the remote console (figure 12.7).



```
Scientific Linux release 6.2 (Carbon)
Kernel 2.6.32-220.13.1.el6.x86_64 on an x86_64

node003 login: _
```

Figure 12.6: Physical Console Resumes After Freeze During SOL Access



```
Starting system logger: Ä  OK  Ü
Starting irqbalance: Ä  OK  Ü
Starting openvpn: Ä  OK  Ü
Starting rpcbind: Ä  OK  Ü
Starting NFS statd: Ä  OK  Ü
Starting mdmonitor: Ä  OK  Ü
Mounting NFS filesystems:  Ä  OK  Ü
Mounting other filesystems:  Ä  OK  Ü
Starting acpi daemon: Ä  OK  Ü
Retrigger failed udev eventsÄ  OK  Ü
Adding udev persistent rulesÄ  OK  Ü
Starting mcelog daemon
Starting sshd: Ä  OK  Ü
Starting xinetd: Ä  OK  Ü
ntpdate: Synchronizing with time server: Ä  OK  Ü
Starting ntpd: Ä  OK  Ü
Starting postfix: Ä  OK  Ü
Starting crond: Ä  OK  Ü
Waiting for CMDaemon to start...Ä  OK  Ü

Scientific Linux release 6.2 (Carbon)
Kernel 2.6.32-220.13.1.el6.x86_64 on an x86_64

node003 login: 
```

Figure 12.7: Remote Console End Display After Boot

The physical console thus misses displaying several parts of the boot progress.

Exit sequence: The `conman` console viewer session can be exited with the sequence `&.` (the last entry in the sequence being a period). Strictly speaking, the `&.` sequence must actually be preceded by an *<ENTER>*.

The console buffer issue when accessing the remote console: A feature of SOL console clients is that the administrator is not presented with any text prompt from the node that is being accessed. This is useful in some cases, and can be a problem in others.

An example of the issue is the case where the administrator has already logged into the console and typed in a command in the console shell, but has no intention of pressing the <*ENTER*> key until some other tasks are first carried out. If the connection breaks at this point, then the command typed in is held in the console shell command buffer, but is not displayed when a remote serial connection is re-established to the console—the previously entered text is invisible to the client making the connection. A subsequent <*ENTER*> would then attempt to execute the command. This is why an <*ENTER*> is not sent as the last key sequence during automated SOL access, and it is left to the administrator to enter the appropriate key strokes.

To avoid commands in the console shell buffer inadvertently being run when taking over the console remotely, the administrator can start the session with a <*CTRL*>`-u` to clear out text in the shell before pressing <*ENTER*>.

13 Third Party Software

In this chapter, several third party software packages included in the DirectMon repository are described briefly. For all packages, references to the complete documentation are provided.

13.1 Modules Environment

Package name: `env-modules` for head node

The *modules environment* (`http://modules.sourceforge.net/`) allows a user of a cluster to modify the shell environment for a particular application or even a particular version of an application. Typically, a module file defines additions to environment variables such as `PATH`, `LD_LIBRARY_PATH`, and `MANPATH`.

Cluster users use the `module` command to load or remove modules from their environment. The `module(1)` man page has more details about the command, and aspects of the modules environment that are relevant for administrators are discussed in section 3.2.

The modules environment from a user's perspective is covered in the DirectMon *User Manual*.

13.2 Shorewall

Package name: `shorewall`

13.2.1 The Shorewall Service Paradigm

DirectMon uses Shoreline Firewall (more commonly known as "Shorewall") package to provide firewall and gateway functionality on the head node of a cluster.

Shorewall is a flexible and powerful high-level interface for the netfilter packet filtering framework inside the 2.4 and 2.6 Linux kernels. Behind the scenes, Shorewall uses the standard `iptables` command to configure netfilter in the kernel. All aspects of firewall and gateway configuration are handled through the configuration files located in `/etc/shorewall`.

After modifying Shorewall configuration files, Shorewall must be restarted to have the new configuration take effect. From the shell prompt, this can be carried out with:

```
service shorewall restart
```

In DirectMon^{TM}, Shorewall is managed by CMDaemon, in order to handle the automation of cloud node access. Restarting Shorewall can thus also be carried out within the services submode (section 4.12) by default:

```
[ddnmon61->device[ddnmon61]->services[shorewall]]% restart
restart Successfully restarted service shorewall on: ddnmon61
```

System administrators who need a deeper understanding of how Shorewall is implemented should be aware that Shorewall does not really run as a daemon process. The command to restart the service therefore does not stop and start a `shorewall` daemon. Instead it carries out the configuration of netfilter through implementing the iptables configuration settings, and then exits. It exits without leaving some kind of `shorewall` process up and running.

13.2.2 Shorewall Zones, Policies, And Rules

In the default setup, Shorewall provides gateway functionality to the internal cluster network on the first network interface (`eth0`). This network is known as the `nat` zone to Shorewall. The external network (i.e. the connection to the outside world) is assumed to be on the second network interface (`eth1`). This network is known as the `net` zone in Shorewall. Letting DirectMon take care of the network interfaces settings is recommended for all interfaces on the head node (section 4.3). The `interfaces` file is generated by the cluster management daemon, and any extra instructions that cannot be added via `cmgui` or `cmsh` can be added outside of the file section clearly demarcated as being maintained by CMDaemon.

Shorewall is configured by default (through `/etc/shorewall/policy`) to deny all incoming traffic from the `net` zone, except for the traffic that has been explicitly allowed in `/etc/shorewall/rules`. Providing (a subset of) the outside world with access to a service running on a cluster, can be accomplished by creating appropriate rules in `/etc/shorewall/rules`. By default, the cluster responds to ICMP ping packets. Also, during cluster installation, the following ports are open by default, but can be set to be blocked by the administrator (figure 2.25):

- SSH
- HTTP
- HTTPS
- port 8081, which allows access to the cluster management daemon.

13.2.3 Clear And Stop Behavior In `service` Options, `bash` Shell Command, And `cmsh` Shell

To remove all rules, for example for testing purposes, the `clear` option should be used from the Unix shell. This then allows all network traffic through:

```
shorewall clear
```

Administrators should be aware that in Red Hat distribution variants the `service shorewall stop` command corresponds to the unix shell `shorewall stop` command, and not to the unix shell `shorewall clear` command. The `stop` option for the service and shell blocks network traffic but allows a pre-defined minimal safe set of connections, and is not the same as completely removing Shorewall from consideration. This situation is indicated in the following table:

Correspondence Of Stop And Clear Options In Shorewall In Red Hat Derivatives

iptables rules	**Service**	**Unix Shell**	`cmsh` **shell**
keep a safe set:	service shorewall stop	shorewall stop	*no equivalent*
clear all rules:	*no equivalent*	shorewall clear	stop shorewall

13.2.4 Further Shorewall Quirks

Behavior That May Trip Debian Users

The situation differs from Debian-like distributions where "`service shorewall stop`" corresponds to "`shorewall clear`" and removes Shorewall from consideration.

Standard Distribution Firewall Should Be Disabled

Administrators should also be aware that distributions such as Red Hat and SUSE run their own set of high-level iptable setup scripts if the standard distribution firewall is enabled. Since DirectMon requires Shorewall, then, to avoid conflict, the standard distribution firewall must stay disabled. Shorewall can be configured to set up whatever iptable rules are installed by the standard distribution script instead.

Shorewall Stopped Outside Of DirectMon Considered Harmful

System administrators wishing to stop Shorewall should note that DirectMon by default has the `autostart` setting (section 4.12) set to `on`. With such a value, CMDaemon attempts to restart a stopped Shorewall if the service has been stopped from outside of `cmsh` or `cmgui`.

Stopping Shorewall outside of `cmsh` or `cmgui` is considered harmful, because it can trigger a failover. This is because stopping Shorewall blocks the failover prevention monitoring tests. These tests are the status ping and backup ping (both based on SYN packet connections), and the CMDaemon status (based on SOAP calls) (section 15.4.2). In most cases, with default settings, Shorewall is not restarted in time, even when `autostart` is on, so that a failover then takes place.

A failover procedure is quite a sensible option when Shorewall is stopped from outside of `cmsh` or `cmgui`, because besides the failover monitoring tests failing, other failures also make the head node pretty useless. The blocking of ports means that, amongst others, workload managers and NFS shares are also unable to connect. Ideally, therefore, Shorewall should not be stopped outside `cmsh` or `cmgui` in the first place.

Full documentation on the specifics of Shorewall is available at `http://www.shorewall.net`.

13.3 Compilers

DDN provides convenient RPM packages for several compilers that are popular in the HPC community. All of those may be installed through `yum` or `zypper` (section 10.2) but (with the exception of GCC) require an installed license file to be used.

13.3.1 GCC

Package name: `gcc-recent`

The GCC suite that the distribution provides is also present by default.

13.3.2 Intel Compiler Suite

Package names:

Intel Compiler Suite Versions

2011	2013
`intel-compiler-common`	`intel-compiler-common-2013`
`intel-compiler-common-32`	`intel-compiler-common-2013-32`
`intel-cc`	`intel-cc-2013`
`intel-cc-32`	`intel-cc-2013-32`
`intel-fc`	`intel-fc-2013`
`intel-fc-32`	`intel-fc-2013-32`
`intel-idb`	`intel-idb-2013`
`intel-idb-32`	`intel-idb-2013-32`
`intel-ipp`	`intel-ipp-2013`
`intel-ipp-32`	`intel-ipp-2013-32`
`intel-itac`	`intel-itac-2013`
`intel-mkl`	`intel-mkl-2013`
`intel-mkl-32`	`intel-mkl-2013-32`
`intel-openmp-2013`	`intel-openmp-2013`
`intel-openmp-32`	`intel-openmp-2013-32`
`intel-sourcechecker`	`intel-sourcechecker-2013`
`intel-sourcechecker-32`	`intel-sourcechecker-2013-32`
`intel-tbb`	`intel-tbb-2013`

The Intel compiler packages are provided as a suite, for example by Intel® C++ Composer XE. The 2013 version of the suite can be installed concurrently with the 2011 version. However, the administrator should be aware when doing this that if a short module (section 3.2) name is used, such as `intel/compiler`, then the compiler version with the highest character value folowing the short module name is the one that will be used.

Typically the suite includes the Intel Fortran (indicated by `fc`) and Intel C++ compilers (part of the C compiler package, indicated by `cc`). Along with the 64-bit (i.e. EM64T) version of both compilers, the 32-bit version may optionally be installed. The 32-bit packages have package names ending in "-32"

Both the 32-bit and 64-bit versions can be invoked through the same set of commands. The modules environment (section 3.2) provided when installing the packages can be loaded accordingly, to select one of the two versions. For the C++ and Fortran compilers the 64-bit and 32-bit modules are called `intel/compiler/64` and `intel/compiler/32` respectively.

Version 2013 and higher of the suite can also be used to compile a native application on Intel Xeon Phi coprocessors. The compiler for this is installed by default in `/opt/intel/composer_xe_`<*version*>. The *DirectMon User Manual* has more on compiling for the Intel Xeon Phi.

The Intel compilers include a debugger, which can also be accessed by loading the `intel/compiler/64` or `intel/compiler/32` module. The following commands can be used to run the Intel compilers and debugger:

- `icc`: Intel C/C++ compiler
- `ifort`: Intel Fortran 90/95 compiler
- `idb`: Intel Debugger

Optional packages are:

- `intel-ipp`: Integrated Performance Primitives
- `intel-mkl`: Math Kernel Library
- `intel-itac`: Trace Analyzer And Collector
- `intel-sourcechecker`: Source Checker
- `intel-tbb`: Threading Building Blocks

A short summary of a package can be shown using, for example: "`yum info intel-fc`".

The compiler packages require a license, obtainable from Intel, and placed in `/cm/shared/licenses/intel`.

Full documentation for the Intel compilers is available at `http://software.intel.com/en-us/intel-compilers/`.

In the following example the license file is copied into the appropriate location, the standard 64-bit version of the C/C++ and Fortran compilers are installed along with their 32-bit versions, a modules environment (section 3.2) is loaded for use by the root user, and the modules environment is added for regular root user use with "`module initadd`":

Example

```
[root@ddnmon61~]# cp <license file> /cm/shared/licenses/intel/
[root@ddnmon61~]# yum install intel-cc intel-cc-32 intel-fc intel-fc-32
 (installation text output skipped)
[root@ddnmon61~]# module load intel/compiler/64
[root@ddnmon61~]# module initadd intel/compiler/64
```

How to load modules for use and regular use by non-root users is explained in section 3.2.3.

13.3.3 PGI High-Performance Compilers

Package name: `pgi`

The PGI compiler package contains the PGI C++ and Fortran 77/90/95 compilers.

- `pgcc`: PGI C compiler
- `pgCC`: PGI C++ compiler
- `pgf77`: PGI Fortran 77 compiler
- `pgf90`: PGI Fortran 90 compiler
- `pgf95`: PGI Fortran 95 compiler
- `pgdbg`: PGI debugger

Full documentation for the PGI High-Performance Compilers is available at `http://www.pgroup.com/resources/docs.htm`.

13.3.4 AMD Open64 Compiler Suite

Package name: `open64`

The Open64 Compiler Suite contains optimizing C++ and Fortran compilers.

- `opencc`: Open64 C compiler
- `openCC`: Open64 C++ compiler
- `openf90`: Open64 Fortran 90 compiler
- `openf95`: Open64 Fortran 95 compiler

Full documentation for the AMD Open64 Compiler Suite is available at: `http://www.amd.com`.

13.3.5 FLEXlm License Daemon

Package name: `flexlm`

For the Intel and PGI compilers a FLEXlm license must be present in the `/cm/shared/licenses` tree.

For workstation licenses, i.e. a license which is only valid on the head node, the presence of the license file is typically sufficient.

However, for floating licenses, i.e. a license which may be used on several machines, possibly simultaneously, the FLEXlm license manager, `lmgrd`, must be running.

The `lmgrd` service serves licenses to any system that is able to connect to it through the network. With the default firewall configuration, this means that licenses may be checked out from any machine on the internal cluster network. Licenses may be installed by adding them to `/cm/shared/licenses/lmgrd/license.dat`. Normally any FLEXlm license starts with the following line:

```
SERVER hostname MAC port
```

Only the first FLEXlm license that is listed in the `license.dat` file used by `lmgrd` may contain a `SERVER` line. All subsequent licenses listed in `license.dat` should have the `SERVER` line removed. This means in practice that all except for the first licenses listed in `license.dat` start with a line:

```
DAEMON name /full/path/to/vendor-daemon
```

The `DAEMON` line must refer to the vendor daemon for a specific application. For PGI the vendor daemon (called `pgroupd`) is included in the `pgi` package. For Intel the vendor daemon (called `INTEL`) must be installed from the `flexlm-intel`.

Installing the `flexlm` package adds a system account `lmgrd` to the password file. The account is not assigned a password, so it cannot be used for logins. The account is used to run the lmgrd process. The `lmgrd` service is not configured to start up automatically after a system boot, but can be configured to do so with:

```
chkconfig lmgrd on
```

The `lmgrd` service is started manually with:

```
/etc/init.d/lmgrd start
```

The `lmgrd` service logs its transactions and any errors to `/var/log/lmgrd.log`.

More details on FLEXlm and the `lmgrd` service are available at `http://www.rovicorp.com`.

13.4 Intel Cluster Checker

Package name: `intel-cluster-checker`

Intel Cluster Checker is a tool that checks the health of the cluster and verifies its compliance against the requirements defined by the Intel Cluster Ready Specification. This section lists the steps that must be taken to certify a cluster as Intel Cluster Ready.

For additional instructions on using Intel Cluster Checker and its test modules for a particular version <*version*>, the tool documentation located in the cluster at `/opt/intel/clck/<version>/doc/` can be referred to. The URL `http://software.intel.com/en-us/cluster-ready/` has more information on the Intel Cluster Ready (ICR) program.

13.4.1 Package Installation

Package Installation: Other Required Packages

The Intel Cluster Checker tool is provided by the `intel-cluster-checker` package. To meet all the Intel Cluster Ready specification requirements the following software packages also need to be installed on the head and regular nodes:

- `intel-cluster-runtime`
- `cm-config-intelcompliance-master`
- `cm-config-intelcompliance-slave`

Package Installation: Where The Packages Go

The `intel-cluster-checker` and `intel-cluster-runtime` packages are installed only on the head node, although libraries are available to the regular nodes through the shared file system. Packages `cm-config-intelcompliance-master` and `cm-config-intelcompliance-slave` are installed on the head node and software images respectively.

Package Installation: Installing The Packages With A Package Manager

The packages are normally already installed by default on a standard DirectMon cluster. If they are not installed then the packages can be installed using `yum` (or `zypper` if using SLES 11).

Example

```
[root@mycluster ~]# yum install intel-cluster-runtime intel-cluster-che\
cker cm-config-intelcompliance-master
[root@mycluster ~]# chroot /cm/images/default-image
[root@mycluster /]# yum install cm-config-intelcompliance-slave
```

The packages guarantee through package dependencies that all Intel Cluster Ready package requirements are satisfied. If the package manager reports that any additional packages need to be installed, simply agreeing to install them is enough to satisfy the requirements. To ensure compatibility throughout the cluster for packages released by Intel, such as the Intel compilers (section 13.3.2), it is usually necessary to keep `cm-config-intelcompliance-slave` on the regular nodes updated to the same release version as the corresponding packages running on the head node.

Package Installation: Updating The Nodes

After installing the necessary packages the nodes need to be updated. This can be done with an `updateprovisioners` command (if there are node provisioners in the cluster) followed by an `imageupdate` command.

13.4.2 Preparing Configuration And Node List Files

The configuration and package list files are located in the `/etc/intel/clck` directory:

- `config-ib.xml`
- `config-nonib.xml`
- `packagelist.head`
- `packagelist.node`

The input files, containing a list of the nodes to be checked, are created in the `/home/cmsupport/intel-cluster-ready` directory:

- `nodelist`
- `nodelist.ib`

These files are used during the cluster checker execution. During the cluster checker preparation, the `nodelist` and `nodelist.ib` files must be generated. During the first boot of a head node the package list files `packagelist.head` and `packagelist.node` are generated.

Configuration Files

The `config-nonib.xml` and `config-ib.xml` files are default configuration files that have been included as part of the `cm-config-intelcompliance-master` package. Both configuration files may need small modifications based on the cluster for which certification is required.

The configuration file can be copied to the user's home directory and edited as needed. The adjusted configuration file name needs to be provided to the Intel cluster checker command as an argument. Otherwise, the tool uses `/etc/intel/clck/config.xml` by default.

For the certification run, two configuration files are available:

- `config-nonib.xml`
- `config-ib.xml`

During the first boot of the head node, the `/etc/intel/clck/config.xml` link is created.

- If no configuration file is provided when running the cluster check, then `/etc/intel/clck/config.xml` is used as the configuration file
- If the cluster has no InfiniBand interconnect, then `/etc/intel/clck/config.xml` links to the `config-nonib.xml` file
- If the cluster uses an InfiniBand interconnect, then `/etc/intel/clck/config.xml` links to the `config-ib.xml` file

The existence of a link and where it points to can be checked as follows:

```
[root@mycluster ~]# ls -l /etc/intel/clck/config.xml
```

The file or link `/etc/intel/clck/config.xml` can be changed if needed.

Although it is not required for an ICR certification, several performance thresholds can be defined which require tuning based on the hardware that is included in the cluster.

When in doubt, it can be useful to configure threshold values which are certainly too high in performance for the cluster to meet. For example, too high a throughput for disk I/O bandwidth, or too low a time in the case of latency. After running the cluster checker, a (failed) value for the concerned performance parameters will be given, and the performance thresholds can then be adjusted to more realistic numbers based on the results obtained from the run.

Intel Cluster Checker can also be run with the `---autoconfigure` option for automatic configuration, in which case a basic configuration is written to an existing configuration file before the execution starts.

Node Lists

The `nodelist` and `nodelist.ib` files list the nodes which are considered by the Intel Cluster Checker. In the normal case `nodelist` is used. When an InfiniBand interconnect is used in the cluster, the `nodelist.ib` file can be used to run the cluster check entirely over InfiniBand. When the cluster changes, the node lists files must be regenerated with the `clck-prepare` command.

Updating the Node Lists

The `clck-prepare` command is used to generate or update the node lists files. The `cmsupport` account is used to generate the files, since the `cmsupport` account is used to perform the cluster check run. For clusters without InfiniBand interconnect, the `nodelist.ib` file is not generated.

Example

```
[root@mycluster ~]# su - cmsupport
[cmsupport@mycluster ~]$ clck-prepare
Created non InfiniBand node list file /home/cmsupport/intel-cluster-rea\
dy/nodelist

Created InfiniBand node list file /home/cmsupport/intel-cluster-ready/n\
odelist.ib
```

Package Lists

The package list files `packagelist.head` and `packagelist.node` contain lists of all packages installed on the head node and on the regular nodes. These lists are used to ensure that the same software is available on all nodes. The package lists are created on the first boot, and do not change unless explicitly regenerated.

Regenerating Package Lists

The package list files are generated during the first boot of the head node. They can be regenerated, if needed.

An old version of the head node package list can be backed up, and a current one generated, by running the following on the head node:

```
[root@mycluster ~]# cp -p /etc/intel/clck/packagelist.head /etc/intel/c\
lck/packagelist.head.old
[root@mycluster ~]# rpm -qa | sort > /etc/intel/clck/packagelist.head
```

Similarly, an old version of the regular node package list can be backed up, and a current one generated, for `node001` for example, by running the following on the head node:

```
[root@mycluster ~]# cp -p /etc/intel/clck/packagelist.node /etc/intel/c\
lck/packagelist.node.old
[root@mycluster ~]# ssh node001 rpm -qa | sort > /etc/intel/clck/packag\
elist.node
```

13.4.3 Running Intel Cluster Checker

The `cmsupport` account, which is part of a default installation, is used to perform the cluster check run.

The following commands start the cluster checker:

```
[root@mycluster ~]# su - cmsupport
[cmsupport@mycluster ~]$ module initadd intel-cluster-runtime
[cmsupport@mycluster ~]$ module load intel-cluster-runtime
[cmsupport@mycluster ~]$ cluster-check --certification
```

The last line could instead be:

```
[cmsupport@mycluster ~]$ cluster-check --certification ~/custom_con\
fig.xml
```

if a configuration file `config-ib.xml` from the default location has been copied over to the `cmsupport` account directory, and then modified for use by the cluster checker.

Handling Test Failures

The cluster checker produces several output files, with `.xml`, `.out`, `.debug` suffixes, which include time stamps in the filenames. If tests fail, the output files can be consulted for details. The output files can be found in the `~/intel-cluster-ready/logs` directory.

When debugging and re-running tests, the option

```
--include_only <test>
```

can be passed to `cluster-check` to execute only the test named "<*test*>" (and the tests on which it depends).

In a heterogeneous cluster the cluster check run fails as a result of hardware differences. To resolve the failures, it is necessary to create multiple groups of homogeneous hardware. For more information, the Intel Cluster Checker documentation can be consulted.

13.4.4 Applying For The Certificate

When the cluster check run has reported that the "`Check has Succeeded`", a certificate may be requested for the cluster. Requesting a certificate involves creating a "`Bill of Materials`", which includes software as well as hardware. This is then submitted together with the output files from Intel Cluster Checker runs and the packages lists to `cluster@intel.com`. The Intel Cluster Ready site contains interactive submissions forms that make the application process as easy as possible. For more details, `http://software.intel.com/en-us/cluster-ready/` can be visited.

13.5 CUDA For GPUs

The optional CUDA packages should be installed in order to take advantage of the computational capabilities of NVIDIA GPUs.

13.5.1 Installing CUDA

Where And What CUDA Packages Are Installed

A number of CUDA 4.2, 5.0, 5.5, and 6.0 packages exist in the YUM repository. The CUDA 5.5 and generic driver packages are:

Package	Type	Description
`cuda55-toolkit`	shared	CUDA math libraries and utilities
`cuda55-sdk`	shared	CUDA software development kit
`cuda55-profiler`	shared	CUDA profiler
`cuda55-tdk`*	shared	CUDA Tesla deployment kit tools
`cuda-driver`	local	CUDA driver
`cuda-xorg`*	local	CUDA X.org driver and libraries

* optional

For CUDA 4.2 and 5.0 package names, `42` and `50` are used instead of `55`. The local packages for CUDA can be used for all versions. For CUDA 6.0, profiler components have been moved into the CUDA toolkit package, and the CUDA profiler package no longer exists.

X is used in the following sections to indicate all the possible CUDA version numbers. The installation procedure for CUDA 4.2, CUDA 5.0, CUDA 5.5, or CUDA 6.0, described next, is thus indicated as the installation procedure for CUDA*X*.

The packages of type "shared" in the preceding table should be installed on the head nodes of a cluster using CUDA-compatible GPUs. The packages of type "local" should be installed to all nodes that access the GPUs. In most cases this means that the `cuda-driver` package should be installed in a software image (section 3.1.2).

If a head node also accesses GPUs, the `cuda-driver` package should be installed on it, too.

For packages of type shared, multiple versions can be installed on the head node at one time, out of the choice of CUDA*X*. The particular CUDA version that is run on the node can be selected via a modules environment command:

Example

```
module add shared cuda55/toolkit
```

CUDA Package Dependencies

The CUDA packages have additional dependencies that may require access to repositories besides the main repository in order to resolve them automatically. For example, for Red Hat, a subscription (Appendix M.1.2) is needed to the `rhel-x86_64-server-supplementary-5` channel for RHEL5, and to the `rhel-x86_64-server-optional-6` channel for RHEL6.

In particular, the `freeglut`, `freeglut-devel`, and `xorg-x11-util-macros` packages are required. The installation ISO/DVD that Red Hat provides contains packages from the main repository, and does not contain these packages. These packages are provided with a DirectMon installation ISO/DVD for Red Hat. Updates must however come from a subscription to the Red Hat supplementary/optional channels. Packages that are needed for a working DDN cluster, and which are provided by the DirectMon installation ISO/DVD, but which are not provided in the Red Hat installation DVD, are discussed in general in section 10.6.2, where the problems that such package dependencies can cause when creating a software image for a cluster with `cm-create-image` are discussed.

As a separate issue, one of the dependencies of the `cuda-driver` package is the `freeglut-devel` package, so it should be installed on a node that accesses a GPU. If the CUDA SDK source is to be compiled on the head node (with the head node not accessing a GPU, and with the `cuda-driver` package not installed) then the `freeglut`, `freeglut-devel`, and `libXi-devel` packages should be installed on the head node.

The `cuda-driver` package is used to compile the kernel driver, which manages the GPU. Therefore, when installing `cuda-driver` with `yum`, several other X11-related packages are installed too, due to package dependencies.

The `cuda`*X*`-sdk` can be used to compile libraries and tools that are not part of the CUDA toolkit, but used by CUDA software developers, such as the `deviceQuery` binary (section 13.5.3).

The `cuda-xorg` and `cuda`*X*`-tdk` packages are optional. The `cuda-xorg` package contains the driver and libraries for an X server. The `cuda`*X*`-tdk` contains files for the CUDA profiling tools interface, CUDA debugging API and the nVidia Management Library.

Example

For example, on a cluster where (some of) the nodes access GPUs, but the head node does not access a GPU, the following commands can be issued on the head node to install the CUDA 5.5 packages through YUM:

```
yum install cuda55-toolkit cuda55-sdk cuda55-profiler
yum --installroot=/cm/images/default-image install cuda-driver
```

Compiling And Loading CUDA Drivers On The Fly

The `cuda-driver` package provides an `init` script which is executed at boot-time to load the CUDA driver. Because the CUDA driver depends on the running kernel, the script compiles the CUDA driver on the fly, and subsequently loads the module into the running kernel.

The `cuda-driver` package can also be loaded on the fly by calling the `init` script.

Loading the CUDA driver causes a number of diagnostic kernel messages to be logged:

Example

```
[root@mycluster ~]# /etc/init.d/cuda-driver start
Compiling nvidia driver.. installing.. loading..           [  OK  ]
[root@mycluster ~]# dmesg
...
NVRM: loading NVIDIA UNIX x86_64 Kernel Module  319.23  Thu May 16 19:\
36:02 PDT 2013
```

If there is a failure in compiling the CUDA module, it is usually indicated by a message saying “`Could not make module`”, “`NVRM: API mismatch:`”, or “`Cannot determine kernel version`”. Such a failure typically occurs because compilation is not possible due to missing the correct kernel development package from the distribution. Section 13.5.2 explains how to check for, and install, the appropriate missing package.

13.5.2 Installing Kernel Development Packages

This section can be skipped if there is no CUDA compilation problem.

Typically, a CUDA compilation problem (section 13.5.1) is due to a missing or mismatched `kernel` package and `kernel-devel package`.

To check the head node and software images for the installation status of the `kernel-devel` package, the DirectMon utility `kerneldevel-check` is used (section 10.3.5).

Alternatively, if a standard kernel is in use by the image, then simply upgrading CUDA, the standard kernel, and `kernel-devel`, to their latest versions may be a good tactic to fix a CUDA compilation problem, because the `kernel` and `kernel-devel` package versions become synchronized during such an upgrade.

13.5.3 Verifying CUDA

An extensive method to verify that CUDA is working is to run the `verify_cudaX.sh` script, located in the CUDA SDK directory.

This script first copies the CUDA SDK source to a local directory under `/tmp` or `/local`. It then builds CUDA test binaries and runs them. It is possible to select which of the CUDA test binaries are run. These binaries clutter up the disk and are not intended for use as regular tools, so the administrator is urged to remove them after they are built and run.

A help text showing available script options is displayed when "`verify_cudaX.sh -h`" is run.

Example

```
[root@cuda-test ~]# module load shared cuda55/toolkit

[root@cuda-test ~]# cd $CUDA_SDK
[root@cuda-test 5.5.22]# ./verify_cuda55.sh
Copy cuda55 sdk files to "/local/cuda55" directory.

make clean

make (may take a while)


Run all tests? (y/N)? y

Executing: /local/cuda55/bin/linux/release/alignedTypes

[/local/cuda55/bin/linux/release/alignedTypes] - Starting...
GPU Device 0: "Tesla K20c" with compute capability 3.5

[Tesla K20c] has 13 MP(s) x 192 (Cores/MP) = 2496 (Cores)
> Compute scaling value = 1.00
> Memory Size = 49999872
Allocating memory...
Generating host input data array...
Uploading input data to GPU memory...
Testing misaligned types...
uint8...
Avg. time: 1.735281 ms / Copy throughput: 26.834849 GB/s.
        TEST OK
```

```
...
...
All cuda55 just compiled test programs can be found in the
 "/local/cuda55/bin/linux/release/" directory
They can be executed from the "/local/cuda55" directory.

The "/local/cuda55" directory may take up a lot of diskspace.
Use "rm -rf /local/cuda55" to remove the data.
```

Another method to verify that CUDA is working, is to build and use the `deviceQuery` command on a node accessing one or more GPUs. The `deviceQuery` command lists all CUDA-capable GPUs that a device can access, along with several of their properties.

Example

```
[root@cuda-test ~]# module load shared cuda55/toolkit
[root@cuda-test ~]# module load openmpi/gcc/64
[root@cuda-test ~]# cd $CUDA_SDK
[root@cuda-test 5.5.22]# make clean
...
[root@cuda-test 5.5.22]# make
...
[root@cuda-test 5.5.22]# bin/x86_64/linux/release/deviceQuery
bin/x86_64/linux/release/deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 3 CUDA Capable device(s)

Device 0: "Tesla K20c"
  CUDA Driver Version / Runtime Version          5.5 / 5.5
  CUDA Capability Major/Minor version number:    3.5
  Total amount of global memory:                 4800 MBytes
  (13) Multiprocessors, (192) CUDA Cores/MP:     2496 CUDA Cores
  GPU Clock rate:                                614 MHz (0.61 GHz)
  Memory Clock rate:                             2600 Mhz
  Memory Bus Width:                              320-bit
  L2 Cache Size:                                 1310720 bytes
...
...
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 5.5,
CUDA Runtime Version = 5.5, NumDevs = 3, Device0 = Tesla K20c,
Device1 = Tesla K20c, Device2 = Tesla K20c
Result = PASS
```

The CUDA user manual has further information on how to run compute jobs using CUDA.

13.5.4 Verifying OpenCL

CUDA also contains an OpenCL compatible interface. To verify that the OpenCL is working, the `verify_opencl.sh` script can be run.

Example

```
[root@cuda-test ~]# module load shared cuda55/toolkit
[root@cuda-test ~]# cd $CUDA_SDK
[root@cuda-test 5.0.35]# ./verify_opencl.sh
Copy opencl files to "/local/opencl" directory.

make clean

make (may take a while)


Run all tests? (y/N)? y

Executing: /local/opencl/OpenCL/bin/linux/release/oclBandwidthTest

[oclBandwidthTest] starting...

/local/opencl/OpenCL/bin/linux/release/oclBandwidthTest Starting...

Running on...

Tesla K20c

Quick Mode

Host to Device Bandwidth, 1 Device(s), Paged memory, direct access
   Transfer Size (Bytes)        Bandwidth(MB/s)
   33554432                     2528.0
...
...
All opencl just compiled test programs can be found in the
 "/local/opencl/OpenCL/bin/linux/release/" directory
They can be executed from the "/local/opencl/OpenCL" directory.

The "/local/opencl" directory may take up a lot of diskspace.
Use "rm -rf /local/opencl" to remove the data.
```

13.5.5 Configuring The X server

The X server can be configured to use a CUDA GPU. To support the X server, the `cuda-driver`, and `cuda-xorg` packages need to be installed.

The following file pathname lines need to be added to the `Files` section of the X configuration file:

```
    ModulePath "/usr/lib64/xorg/modules/extensions/nvidia"
    ModulePath "/usr/lib64/xorg/modules/extensions"
    ModulePath "/usr/lib64/xorg/modules"
```

The following dynamic module loading lines need to be added to the `Module` section of the X configuration:

```
    Load        "glx"
```

The following graphics device description lines need to be replaced in the `Device` section of the X configuration:

```
    Driver      "nvidia"
```

The default configuration file for X.org is `/etc/X11/xorg.conf`.

Example

```
Section "ServerLayout"
    Identifier     "Default Layout"
    Screen      0  "Screen0" 0 0
    InputDevice    "Keyboard0" "CoreKeyboard"
EndSection

Section "Files"
    ModulePath     "/usr/lib64/xorg/modules/extensions/nvidia"
    ModulePath     "/usr/lib64/xorg/modules/extensions"
    ModulePath     "/usr/lib64/xorg/modules"
EndSection

Section "Module"
    Load           "glx"
EndSection

Section "InputDevice"
    Identifier     "Keyboard0"
    Driver         "kbd"
    Option         "XkbModel" "pc105"
    Option         "XkbLayout" "us"
EndSection

Section "Device"
    Identifier     "Videocard0"
    Driver         "nvidia"
    BusID          "PCI:14:0:0"
EndSection

Section "Screen"
    Identifier     "Screen0"
    Device         "Videocard0"
    DefaultDepth   24
    SubSection     "Display"
        Viewport   0 0
        Depth      24
    EndSubSection
EndSection
```

13.6 OFED Software Stack

13.6.1 Choosing A Distribution Version Or DirectMon Version, Ensuring The Kernel Matches, And Logging The Installation

By default, the Linux distribution OFED packages are matched to the distribution kernel version and installed on the cluster. This is the safest option in most cases, and also allows NFS over RDMA.

DirectMon also packages the OFED software developed by Mellanox and QLogic. The DirectMon packages can be more recent than the distribution packages and in that case can provide support for more recent hardware, as well as more features. The DirectMon OFED packages can be selected and installed during the initial cluster installation (figure 2.11), replacing the default distribution OFED stack. The stack can also be installed later on, after the cluster is set up.

If there is no OFED kernel modules package available for the kernel in use, then the DDN OFED install script tries to build the package from the source package provided by the vendors Mellanox or QLogic. However, very recent kernels may not yet be supported by the source package. If a build fails for such a kernel, then the OFED software stack will fail to install, and nothing is changed on the head node or the software image. OFED hardware manufacturers resolve build problems with their software shortly after they become aware of them, but in the meantime a supported kernel must be used.

When updating kernels on the head or the regular nodes, the updated DirectMon OFED software stack must be reinstalled (if there are packages already available for the kernel) or rebuilt (if there are no such packages provided).

If the DirectMon OFED software stack is installed during the cluster installation procedure itself (section 2.3.7), then some basic information is logged to `/var/log/cmfirstboot.log`, which is the general first boot log.

If the DirectMon OFED software stack is not installed during the cluster installation procedure itself, then it can be installed later when the cluster is up and running. A successful installation of the DirectMon OFED software stack (sections 13.6.2 and 13.6.2) onto a running cluster includes the running of an installation script after the DirectMon OFED package installation. The vendor and version number installed can then be found in `/etc/cm-ofed`. Further installation details can be found in `/var/log/cm-ofed.log`.

13.6.2 Mellanox and QLogic OFED Stack Installation Using The DDN Repository

Package names: `mlnx-ofed`, `mlnx-ofed2`, `qlgc-ofed`

The Mellanox or QLogic OFED stacks are installed and configured by DirectMon in an identical way as far as the administrator is concerned. In this section (section 13.6.2):

<vendor-ofedVersion>

is used to indicate where the administrator must carry out a substitution. Depending on the vendor and version used, the substitution is one of the following:

- `mlnx-ofed`, for the Mellanox version 1.5 stack. This stack is currently supported on DirectMon™ for:
  - Red Hat and derivatives versions 5.5 to 5.10, 6.0 to 6.3
  - SLES11SP1, SLES11SP2
- `mlnx-ofed2`, for the Mellanox version 2 stack. This stack is currently supported on DirectMon™ for:
  - Red Hat and derivatives versions 6.2 to 6.4
  - SLES11SP2
- `qlgc-ofed`, for the QLogic stack.

Thus, for example, a `yum install` command indicated by:

`yum install` *<vendor-ofedVersion>*

means that the installation of the DDN Mellanox package is executed with one of these corresponding `yum install` commands:

```
   yum install mlnx-ofed,
   yum install mlnx-ofed2,
```

or

```
   yum install qlgc-ofed
```

Installing The OFED Stack Provided By The DDN Repository Vendor Package

Running the command:

   `yum install` <*vendor-ofedVersion*>

unpacks several packages and scripts, but does not install and configure the driver due to the fundamental nature of the changes it would carry out. The script:

   <*vendor-ofedVersion*>`-install.sh`

can then be used to install and configure the driver on the nodes as follows:

- On the head node, the default distribution OFED software stack can be replaced with the vendor OFED software stack made available from the DDN repository, by using the script's head option, `-h`:

  ```
  [root@ddnmon61~]# /cm/local/apps/<vendor-ofedVersion>/current/\
  bin/<vendor-ofedVersion>-install.sh -h
  ```

  A reboot is recommended after the script completes the install.

- For a software image, for example `default-image`, used by the regular nodes, the default distribution OFED software stack can be replaced with the vendor OFED software stack made available from the DDN repository, by using the script's software image option, `-s`:

  ```
  [root@ddnmon61~]# /cm/local/apps/<vendor-ofedVersion>/current/\
  bin/<vendor-ofedVersion>-install.sh -s default-image
  ```

  A reboot updates the software image on the regular node.

If the distribution kernel is updated on any of these head or regular nodes after the vendor OFED stack has been installed, then the vendor OFED kernel modules made available from the DDN repository must be recompiled and reinstalled. This can be done by running the installation scripts again. This replaces the kernel modules, along with all the other OFED packages again.

Upgrading Kernels When The OFED Stack Has Been Provided By The DDN Repository Vendor Package

If a vendor OFED stack is installed and configured via the script:

   <*vendor-ofedVersion*>`-install.sh`

then kernel and kernel development updates are prevented from taking place by the configuration. The reason for setting such a block is that the OFED stack is customized for the kernel, and updating the kernel without updating the stack along with it, could lead to unexpected behavior. Updating the kernel, kernel development, and OFED stack in such a configuration therefore requires that the administrator intervene to manually override the block.

The following procedure overrides the block, then updates and installs the kernel packages and OFED stack:

1. Overriding the block:

   - In Red Hat-based systems, The `/etc/yum.conf` file must be edited. In that file, in the line that starts with `exclude`, the `kernel` and `kernel-devel` packages need to be removed, so that they are no longer excluded from updates.
   - In SUSE, the `kernel-default` and `kernel-default-devel` packages must be unlocked. The command:

     ```
     zypper removelock kernel-default kernel-default-devel
     ```

     unlocks them so that they can take part in updates again.

2. Updating the kernel and kernel development packages:

   - `yum update`—or for SUSE `zypper up`—updates the packages on the head node.
   - To update the packages on the regular nodes the procedure outlined in section 10.3.3 is followed:
     - The packages on the regular node image (for example, `default-image`) are updated in Red Hat-based systems as follows:

       ```
       yum --installroot=/cm/images/default-image update
       ```

       or in SUSE as follows:

       ```
       zypper --root=/cm/images/default-image up
       ```

     - The `kernelversion` setting (the exact value used varies) is updated as follows:

       Example

       ```
       cmsh -c "softwareimage use default-image; set kernel\
       version 2.6.32-220.17.1.el6.x86_64; commit"
       ```

       This ensures that the updated kernel is used after reboot.

3. A reboot of the regular and head nodes installs the new kernel.

4. Configuring and installing the OFED stack driver for the new kernel is done by running the script <*vendor-ofedVersion*>`-install.sh` as follows:

   - For a stack that is on the head node, the compilation should be done together with the `-h` option:

     ```
     [root@ddnmon61~]# /cm/local/apps/<vendor-ofedVersion>/current/\
     bin/<vendor-ofedVersion>-install.sh -h
     ```

   - For a software image used by the regular nodes, for example `default-image`, the compilation should be done together with the `-s` option:

     ```
     [root@ddnmon61~]# /cm/local/apps/<vendor-ofedVersion>/current/\
     bin/<vendor-ofedVersion>-install.sh -s default-image
     ```

13.7 Lustre

This section covers aspects of Lustre, a parallel distributed filesystem which can be used for clusters.

After a short architectural overview of Lustre, steps to set up a Lustre filesystem to work with DirectMon are described.

Further details on Lustre, including the Lustre manual, can be found at `http://wiki.whamcloud.com`.

13.7.1 Architecture

There are four components to a Lustre filesystem:

1. One management service (MGS)
2. One metadata target (MDT) on the metadata server (MDS)
3. Multiple object storage target (OSTs), on an object storage server (OSS)
4. Clients that access and use the data on the Lustre filesystem

The management services run on the metadata server, and hold information for all Lustre filesystems running in a cluster.

Metadata values, like filenames, directories, permissions, and file layout are stored on the metadata target. The file data values themselves are stored on the object storage targets.

Among the supported Lustre networking types are TCP/IP over Ethernet and InfiniBand.

13.7.2 Server Implementation

Lustre servers, MDS, and OSSs, run on a patched kernel. The patched kernel, kernel modules, and software can be installed with RPM packages. The Lustre server software can also be compiled from source, but the kernel needs to be patched and recreated. Lustre supports one kernel version per Lustre version.

To use Lustre with DirectMon, a Lustre server image and a Lustre client image are installed onto the head node so that they can provision the Lustre nodes.

Creating The Lustre Server Image

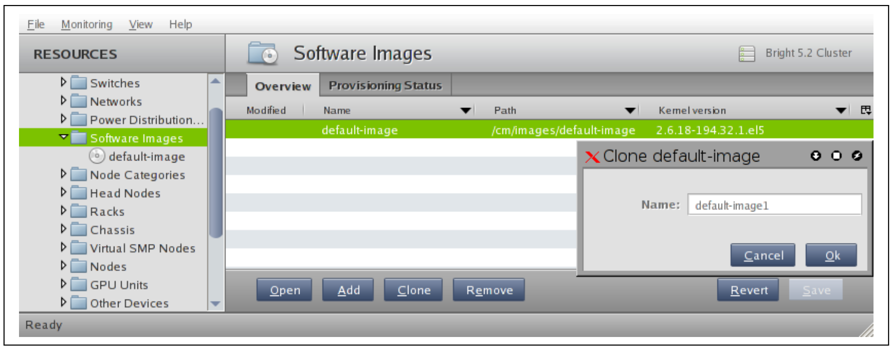
To create a Lustre server image, a clone is made of an existing software image, for example from `default-image`.

In `cmgui` this is done by selecting the `Software Images` resource to bring up the `Overview` tabbed pane display. Selecting the image to clone and then clicking on the `Clone` button prompts for a confirmation to build a clone image (figure 13.1):

Alternatively, `cmsh` on the head node can create a clone image:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% softwareimage
[mycluster->softwareimage]% clone default-image lustre-server-image
[mycluster->softwareimage*[lustre-server-image*]]% commit
```



Modified	Name	Path	Kernel version
	default-image	/cm/images/default-image	2.6.18-194.32.1.el5

Clone default-image

Name: default-image1

Figure 13.1: cmgui: Cloning An Image

The RPM Lustre packages can be downloaded from the Whamcloud website. It is best to first check which version of Lustre can be used for a particular distribution against the Lustre Support Matrix at http://wiki.whamcloud.com/display/PUB/Lustre+Support+Matrix.

After choosing a Lustre version from the Lustre Support Matrix, the appropriate distribution and platform can be chosen. For CentOS and Scientific Linux (SL), Red Hat packages can be used. Download links for Lustre releases, among others kernel, module, lustre and lustre-ldiskf packages, can be found at http://wiki.whamcloud.com/display/PUB/Lustre+Releases. Download links for Lustre tools, among others the e2fsprogs package, can be found at http://wiki.whamcloud.com/display/PUB/Lustre+Tools

The RPM packages to download are:

- kernel: Lustre-patched kernel (MDS/MGS/OSS only)
- lustre-modules: Lustre kernel modules (client and server for the Lustre-patched kernel)
- lustre: Lustre userland tools (client and server for the Lustre-patched kernel)
- lustre-ldiskfs: Backing filesystem kernel module (MDS/MGS/OSS only)
- e2fsprogs: Backing filesystem creation and repair tools (MDS/MGS/OSS only)
- e2fsprogs-libs: Backing filesystem creation and repair tools libraries (MDS/MGS/OSS and EL6 only)
- e2fsprogs-devel: Backing filesystem creation and repair tools development (MDS/MGS/OSS and EL6 only)

The Lustre e2fsprogs package may also depend upon the libss and libcom_err packages, which are also found at http://wiki.whamcloud.com/display/PUB/Lustre+Tools and must then be installed along with e2fsprogs.

In most cases on EL5 nodes, the e2fsprogs package from the distribution is already installed, so the package only has to be upgraded. On EL5 nodes it is possible that the Lustre e2fsprogs package conflicts with the `e4fsprogs` package from the distribution, in which case the `e4fsprogs` package has to be removed. If the Lustre kernel version has a lower version number than the already installed kernel, then the Lustre kernel needs to be installed with the `--force` option. Warning and error messages that may display about installing packages in a software image can be ignored.

Example

```
[root@mycluster ~]# mkdir /cm/images/lustre-server-image/root/lustre
[root@mycluster ~]# cp kernel-* lustre-* e2fsprogs-* /cm/images/lustre-\
server-image/root/lustre
[root@mycluster ~]# chroot /cm/images/lustre-server-image
[root@mycluster /]# cd /root/lustre
[root@mycluster lustre]# rpm -e e4fsprogs
[root@mycluster lustre]# rpm -Uvh e2fsprogs-1.41.90.wc3-0redhat.x86_64.\
rpm
[root@mycluster lustre]# rpm -ivh --force kernel-2.6.18-238.19.1.el5_lu\
stre.g65156ed.x86_64.rpm
[root@mycluster lustre]# rpm -ivh \
  lustre-ldiskfs-3.3.0-2.6.18_238.19.1.el5_lustre.g65156ed_g9d71fe8.x86_\
64.rpm \
  lustre-2.1.0-2.6.18_238.19.1.el5_lustre.g65156ed_g9d71fe8.x86_64.rpm \
  lustre-modules-2.1.0-2.6.18_238.19.1.el5_lustre.g65156ed_g9d71fe8.x86_\
64.rpm
[root@mycluster lustre]# rm kernel-* lustre-* e2fsprogs-*
```

In most cases on EL6 nodes, the `e2fsprogs` package from the distribution is already installed, so the package only has to be upgraded. If the Lustre kernel version has a lower version number than the already installed kernel, then the Lustre kernel needs to be installed with the `--force` option. Warning and error messages that may display about installing packages in a software image can be ignored.

Example

```
[root@mycluster ~]# mkdir /cm/images/lustre-server-image/root/lustre
[root@mycluster ~]# cp kernel-* lustre-* e2fsprogs-* /cm/images/lustre-\
server-image/root/lustre/
[root@mycluster ~]# chroot /cm/images/lustre-server-image
[root@mycluster /]# cd /root/lustre
[root@mycluster lustre]# rpm -Uvh \
  e2fsprogs-1.41.90.wc3-7.el6.x86_64.rpm \
  e2fsprogs-libs-1.41.90.wc3-7.el6.x86_64.rpm \
  e2fsprogs-devel-1.41.90.wc3-7.el6.x86_64.rpm
[root@mycluster lustre]# rpm -ivh --force kernel-2.6.32-131.6.1.el6_lus\
tre.g65156ed.x86_64.rpm
[root@mycluster lustre]# rpm -ivh \
  lustre-ldiskfs-3.3.0-2.6.32_131.6.1.el6_lustre.g65156ed.x86_64_g9d71f\
e8.x86_64.rpm \
  lustre-2.1.0-2.6.32_131.6.1.el6_lustre.g65156ed.x86_64_g9d71fe8.x86_6\
4.rpm \
  lustre-modules-2.1.0-2.6.32_131.6.1.el6_lustre.g65156ed.x86_64_g9d71f\
e8.x86_64.rpm
[root@mycluster lustre]# rm kernel-* lustre-* e2fsprogs-*
```

The kernel version is set to the Lustre kernel version for the Lustre server image:

Example

```
[root@mycluster ~]# cd /cm/images/lustre-server-image/boot
[root@mycluster boot]# ls -1 vmlinuz-*
vmlinuz-2.6.18-238.19.1.el5_lustre.g65156ed
vmlinuz-2.6.18-274.7.1.el5
[root@mycluster ~]# cmsh
[mycluster]% softwareimage
[mycluster->softwareimage]% use lustre-server-image
[mycluster->softwareimage[lustre-server-image]]% set kernelversion 2.6.\
18-238.19.1.el5_lustre.g65156ed
[mycluster->softwareimage*[lustre-server-image*]]% commit
```

Creating The Lustre Server Category

A node category is cloned. For example, `default` to `lustre-server`. The software image is set to the Lustre server image, the `installbootrecord` option is enabled, and the `roles` option is cleared:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% category
[mycluster->category]% clone default lustre-server
[mycluster->category*[lustre-server*]]% set softwareimage lustre-server\
-image
[mycluster->category*[lustre-server*]]% set installbootrecord yes
[mycluster->category*[lustre-server*]]% clear roles
[mycluster->category*[lustre-server*]]% commit
```

Creating Lustre Server Nodes

An MDS node is created with `cmsh`:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% device
[mycluster->[device]]% add physicalnode mds001 10.141.16.1
[mycluster->[device*[mds001*]]]% set category lustre-server
[mycluster->[device*[mds001*]]]% commit
```

One or multiple OSS node(s) are created with `cmsh`:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% device
[mycluster->[device]]% add physicalnode oss001 10.141.32.1
[mycluster->[device*[oss001*]]]% set category lustre-server
[mycluster->[device*[oss001*]]]% commit
```

After the first boot and initial installation, the MDS and OSS(s) are configured to boot from the local drive instead of the network, to preserve locally made changes. The BIOS of each server needs to be configured to boot from the local drive.

For nodes based on EL6 the Lustre initrd file needs to regenerated, after the first boot and initial installation. To regenerate the initrd image file, for the nodes in the `lustre-server` category:

```
[root@mycluster ~]# cmsh
[mycluster]% device
[mycluster->device]% pexec -c lustre-server "mv \
  /boot/initrd-2.6.32-131.6.1.el6_lustre.g65156ed.x86_64.orig \
  /boot/initrd-2.6.32-131.6.1.el6_lustre.g65156ed.x86_64.old"
[mycluster->device]% pexec -c lustre-server "mkinitrd \
  /boot/initrd-2.6.32-131.6.1.el6_lustre.g65156ed.x86_64.orig \
  2.6.32-131.6.1.el6_lustre.g65156ed.x86_64"
```

Warning and error messages that display about write errors or broken pipes can be ignored.

Creating The Lustre Metadata Target

On the metadata server a metadata target must be created. To create the metadata target, a raw block device without partitioning should be used. The device can also be an external storage device and/or a redundant storage device. Setting up a RAID mode capable of dealing with device failure is strongly recommended for block devices to be used as metadata targets, since Lustre itself does not support any redundancy on the filesystem level. The metadata server also acts as a management server.

To format a metadata target, `mkfs.lustre` is used. For example, the following formats `/dev/sdb`, setting the Lustre filesystem name to "`lustre00`":

Example

```
[root@mds001 ~]# mkfs.lustre --fsname lustre00 --mdt --mgs /dev/sdb
```

The filesystem is mounted and the entry added to `/etc/fstab`:

Example

```
[root@mds001 ~]# mkdir /mnt/mdt
[root@mds001 ~]# mount -t lustre /dev/sdb /mnt/mdt
[root@mds001 ~]# echo "/dev/sdb /mnt/mdt lustre rw,_netdev 0 0" >> /etc\
/fstab
```

Creating The Lustre Object Storage Target

On the object storage server one or multiple object storage target(s) can be created. To create the object storage target, a raw block device without partitioning should be used. The device can also be an external storage device and/or a redundant storage device. Setting up a RAID mode capable of dealing with device failure is strongly recommend for block devices to be used as object storage targets, since Lustre itself does not support any redundancy on the filesystem level.

To format an object storage target, `mkfs.lustre` is used. For example, the following command:

Example

```
[root@oss001 ~]# mkfs.lustre --fsname lustre00 --ost --index=0\
--mgsnode=10.141.16.1@tcp0 /dev/sdb
```

- sets the filesystem name to `lustre00`
- sets the OST index number to `0`

- sets the management node to `10.141.16.1`
- and sets the network type to TCP/IP

when formatting the Lustre filesystem on `/dev/sdb`.

Specifying the OST index at format time simplifies certain debugging and administrative tasks.

After formatting the filesystem, it can be mounted and an entry added to `/etc/fstab`:

Example

```
[root@oss001 ~]# mkdir /mnt/ost01
[root@oss001 ~]# mount -t lustre /dev/sdb /mnt/ost01
[root@oss001 ~]# echo "/dev/sdb /mnt/ost01 lustre rw,_netdev 0 0" >> /e\
tc/fstab
```

After mounting the OST(s) the Lustre clients can mount the Lustre filesystem.

13.7.3 Client Implementation

There are several ways to install a Lustre client.

If the client has a supported kernel version, the `lustre-client` RPM package and `lustre-client-modules` RPM package can be installed. The `lustre-client-modules` package installs the required kernel modules.

If the client does not have a supported kernel, a Lustre kernel, Lustre modules, and Lustre userland software can be installed with RPM packages.

The client kernel modules and client software can also be built from source.

Creating The Lustre Client Image: Method 1

This method describes how to create a Lustre client image with Lustre client RPM packages. It requires that the `lustre-client-module` package have the same kernel version as the kernel version used for the image.

To create a starting point image for the Lustre client image, a clone is made of the existing software image, for example from `default-image`.

The clone software image is created via `cmgui` (figure 13.1), or using `cmsh` on the head node:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% softwareimage
[mycluster->softwareimage]% clone default-image lustre-client-image
[mycluster->softwareimage*[lustre-client-image*]]% commit
```

The RPM Lustre client packages are downloaded from the Whamcloud website:

- `lustre-client`: Lustre client userland tools (client for unpatched vendor kernel)

- `lustre-client-modules`: Lustre client kernel modules (client for unpatched vendor kernel)

The same Lustre version which is used for the Lustre servers is used for the Lustre clients.

The kernel version of the `lustre-client-modules` package must also match that of the kernel used. It is 2.6.18-238.19.1.el5 in the following example:

Example

```
[root@mycluster ~]# ls lustre-client-modules-*
lustre-client-modules-2.1.0-2.6.18_238.19.1.el5_g9d71fe8.x86_64.rpm
[root@mycluster ~]# cmsh -c "softwareimage; use lustre-client-image; ge\
t kernelversion"
2.6.18-238.19.1.el5
```

The installation can then be carried out:

Example

```
[root@mycluster ~]# mkdir /cm/images/lustre-client-image/root/lustre
[root@mycluster ~]# cp lustre-client-*.rpm /cm/images/lustre-client-ima\
ge/root/lustre/
[root@mycluster ~]# chroot /cm/images/lustre-client-image
[root@mycluster /]# cd /root/lustre
[root@mycluster lustre]# rpm -ivh lustre-client-modules-2.0.0.1-2.6.18_\
164.11.1.el5_lustre.2.0.0.1.x86_64.rpm
[root@mycluster lustre]# rpm -ivh \
  lustre-client-modules-2.1.0-2.6.18_238.19.1.el5_g9d71fe8.x86_64.rpm \
  lustre-client-2.1.0-2.6.18_238.19.1.el5_g9d71fe8.x86_64.rpm
[root@mycluster lustre]# rm lustre-client-*.rpm
```

Creating The Lustre Client Image: Method 2

This method describes how to create a Lustre client image with a Lustre kernel package.

To create the image for the Lustre client image, a clone is made of the existing `lustre-server-image` software image.

A clone software image is created via `cmgui` (figure 13.1), or using `cmsh` on the head node.

Example

```
[root@mycluster ~]# cmsh
[mycluster]% softwareimage
[mycluster->softwareimage]% clone lustre-server-image lustre-client-image
[mycluster->softwareimage*[lustre-client-image*]]% commit
```

Creating the `lustre-server-image` software image is described in the Lustre Server Implementation section (section 13.7.2).

Creating The Lustre Client Image: Method 3

This method describes how to create a Lustre client image by building Lustre from source.

As a starting point image for a Lustre client image, a clone is made of the existing software image, for example from `default-image`.

A clone software image is created via `cmgui` (figure 13.1), or using `cmsh` on the head node.

Example

```
[root@mycluster ~]# cmsh
[mycluster]% softwareimage
[mycluster->softwareimage]% clone default-image lustre-client-image
[mycluster->softwareimage*[lustre-client-image*]]% commit
```

The source package can be downloaded from the Whamcloud website. The same Lustre version used for Lustre servers is used for the Lustre clients.

Instead of selecting a Linux distribution and architecture, a source package with the DDN version em<version> to download is chosen:

- `lustre-client-source-<version>.rpm`: Lustre source code

The source file is copied to the image:

Example

```
[root@mycluster ~]# mkdir /cm/images/lustre-client-image/root/lustre
[root@mycluster ~]# cp lustre-client-source-*.rpm /cm/images/lustre-cli\
ent-image/root/lustre
```

If the `kernel-devel` package is not installed on the client image, it is first installed so that the kernel can be compiled. To check the `lustre-client-image` software image among others, for the installation status of the `kernel-devel` package, the DirectMon utility `kernel-devel-check` is used (section 10.3.5).

To determine the kernel version used by the software image:

Example

```
[root@mycluster ~]# cmsh -c "softwareimage get lustre-client-image kern\
elversion"
2.6.18-274.7.1.el5
```

The Lustre software is then built and installed:

Example

```
[root@mycluster ~]# export RPM_BUILD_NCPUS=$(grep -c "^processor" /proc\
/cpuinfo)  # do not skip this step
[root@mycluster ~]# chroot /cm/images/lustre-client-image
[root@mycluster /]# cd /root/lustre
[root@mycluster lustre]# rpm -ivh lustre-client-source-2.1.0-2.6.18_238\
.19.1.el5_g9d71fe8.x86_64.rpm
[root@mycluster lustre]# rm lustre-client-source-2.1.0-2.6.18_238.19.1.\
el5_g9d71fe8.x86_64.rpm
[root@mycluster lustre]# cd /usr/src
[root@mycluster src]# ln -s kernels/2.6.18-274.7.1.el5-x86_64 linux
[root@mycluster src]# cd lustre-2.1.0
[root@mycluster lustre-2.1.0]# ./configure --disable-server
[root@mycluster lustre-2.1.0]# make rpms
[root@mycluster lustre-2.1.0]# cd ..
[root@mycluster src]# rm linux
[root@mycluster src]# rpm -e lustre-client-source
[root@mycluster src]# rm -rf lustre-2.1.0
[root@mycluster src]# cd /usr/src/redhat/RPMS/x86_64
```

```
[root@mycluster x86_64]# rpm -ivh --nodeps lustre-modules-2.1.0-2.6.18_\
274.7.1.el5_g9d71fe8.x86_64.rpm
[root@mycluster x86_64]# rpm -ivh lustre-2.1.0-2.6.18_274.7.1.el5_g9d71\
fe8.x86_64.rpm
[root@mycluster x86_64]# rm lustre-*
[root@mycluster x86_64]# cd ../../SRPMS
[root@mycluster SRPMS]# rm lustre-*
[root@mycluster SRPMS]# cd ../BUILD
[root@mycluster BUILD]# rm -rf lustre-*
```

To configure the `lnet` kernel module to use TCP/IP interface eth1, the string "`options lnet networks=tcp(eth1)`" is added to the `/etc/modprobe.conf` file of the client image:

```
[root@mycluster ~]# echo "options lnet networks=tcp(eth1)" >> /cm/image\
s/lustre-client-image/etc/modprobe.conf
```

To specify that a Lustre node uses a TCP/IP interface and an InfiniBand interface, the string "`options lnet networks=tcp0(eth0),o2ib(ib0)`" is added to the `/etc/modprobe.conf` file of the client image:

```
[root@mycluster ~]# echo "options lnet networks=tcp0(eth0),o2ib(ib0)" >\
> /cm/images/lustre-client-image/etc/modprobe.conf
```

Creating The Lustre Client Category

A node category is cloned. For example: `default` to a new category `lustre-client`. The software image in this category is set to the Lustre client image, `lustre-client`:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% category
[mycluster->category]% clone default lustre-client
[mycluster->category*[lustre-client*]]% set softwareimage lustre-client\
-image
[mycluster->category*[lustre-client*]]% commit
```

Configuring The Lustre Mount On The Client For A Category

The Lustre client category can then be configured to mount the Lustre filesystem (some text in the display here is elided):

Example

```
[root@mycluster ~]# cmsh
[mycluster]% category
[mycluster->category]% use lustre-client
[mycluster->category[lustre-client]]% fsmounts
[mycl...fsmounts]% add /mnt/lustre00
[myc...fsmounts*[/mnt/lustre00*]]% set device 10.141.16.1@tcp0:/lustre00
[myc...fsmounts*[/mnt/lustre00*]]% set filesystem lustre
[myc...fsmounts*[/mnt/lustre00*]]% set mountoptions rw,_netdev
[myc...fsmounts*[/mnt/lustre00*]]% commit
```

The configured fsmounts device is the MGS, which in the example has IP address `10.141.16.1`. The network type used in the example is TCP/IP.

Creating Lustre Client Nodes

A client node is created as follows:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% device
[mycluster->device]% add physicalnode lclient001 10.141.48.1
[mycluster->device*[lclient001*]]% set category lustre-client
[mycluster->device*[lclient001*]]% commit
```

The Lustre client is booted and checked to see if the Lustre filesystem is mounted. The Lustre file stripe configuration of the filesystem can be checked with "`lfs getstripe`", and it can be set with "`lfs setstripe`":

Example

```
[root@lclient001 ~]# lfs getstripe /mnt/lustre00
[root@lclient001 ~]# lfs setstripe -s 1M -o -1 -c -1 /mnt/lustre00
```

The "`lfs setstripe`" command in the example sets the filesystem to use 1MB blocks, the start OST is chosen by the MDS, data is striped over all available OSTs.

13.8 ScaleMP

This section describes how to use ScaleMP's vSMP for Cloud product to create virtual SMP nodes in a cluster.

13.8.1 Installing vSMP For Cloud

Before virtual SMP nodes can be created, the ScaleMP vSMP for Cloud software needs to be installed on the head node of the cluster. The vSMP for Cloud software consists of two components:

- The `image_manager` utility
- The vSMP image

Both components have to be copied to the `/cm/local/apps/vsmp` directory on the head node. In addition, the `/cm/local/apps/vsmp/vSMP.img` symbolic link should point to the vSMP image that should be used.

Example

Installing `image_manager` and version 3.5.155 of the vSMP image:

```
[root@mc ~]# cp image_manager /cm/local/apps/vsmp/
[root@mc ~]# cp vSMP-3.5.155.img /cm/local/apps/vsmp
[root@mc ~]# ln -sf vSMP-3.5.155.img /cm/local/apps/vsmp/vSMP.img
```

13.8.2 Creating Virtual SMP Nodes

After the vSMP for Cloud software has been installed, virtual SMP nodes may be created using cmsh or cmgui.

Creating a virtual SMP node in cmgui is done by clicking the Virtual SMP Nodes folder, clicking the Add button and entering a hostname (e.g. vsmp001). A virtual SMP node behaves like any other physical node, but has an extra configuration tab: Virtual SMP. This tab can be used to configure which physical nodes should be used as components of the virtual SMP node.

Nodes that are made members of a virtual SMP node, go into the Aggregated state and when booted load the vSMP kernel. After all members of a vSMP nodes have booted the vSMP kernel, the virtual SMP node boots as a single (large) node.

Example

Creating and powering up a virtual SMP node using cmsh is done as follows:

```
[mc]% device add virtualsmpnode vsmp001
[mc->device*[vsmp001*]]% set members node001 node002 node003
[mc->device*[vsmp001*]]% interfaces
[mc->device*[vsmp001*]->interfaces]% add physical BOOTIF
[mc->device*[vsmp001*]->interfaces*[BOOTIF*]]% set ip 10.141.10.1
[mc->device*[vsmp001*]->interfaces*[BOOTIF*]]% set network internalnet
[mc->device*[vsmp001*]->interfaces*[BOOTIF*]]% exit
[mc->device*[vsmp001*]->interfaces*]% exit
[mc->device*[vsmp001*]]% set provisioninginterface BOOTIF
[mc->device*[vsmp001*]]% commit
...
[mc->device[vsmp001]]% power reset -n vsmp001
```

After the virtual SMP node boots, it must be identified in the same way that a new physical node has to be identified at first boot. Section 6.4.2 has more information on node identification and selection.

13.8.3 Virtual SMP Node Settings

The vSMP nodes can have their settings accessed from device mode.

For example, using cmsh the value of the system memory can be changed as follows:

Example

```
mycluster:~ # cmsh
[mc]% device use vnode001
[mc->device[vnode001]]% vsmpsettings
[mc->device[vnode001]->vsmpsettings]% show
Parameter                      Value
------------------------------ ----------------------------------------
Boot device
Console redirection            all
Extended acpi                  no
Fault tolerance                RESTART
Hyperthreading                 no
List of boards                 -
```

```
Minimal boards              2
Restart on failure          no
Revision
System memory               100
Type                        VScaleMPSettings
Update backplane            no
[mc->device[vnode001]->vsmpsettings]% set systemmemory 90
[mc->device[vnode001]->vsmpsettings*]% commit
```

Using `cmgui`, the equivalent can be carried out via selection of the vSMP node from the resource tree. This brings up the vSMP tabbed pane, within which settings can be modified and saved.

14 MIC Configuration

14.1 Introduction

The Intel Many Integrated Core (MIC) architecture combines many Intel CPU cores onto a single chip. DirectMon supports the PCIe implementation of the Intel MIC architecture, called the Intel Xeon Phi. In this section the terms MIC, coprocessor, or (MIC) card, mean the Intel Xeon Phi, while a MIC host means a node where a MIC card is installed.

The MIC architecture, k1om, is quite similar to x86_64. Usually an application written for x86_64 CPU can simply be rebuilt using special compiler flags. Thus, Linux can run inside a coprocessor. DirectMon provides special packages with names ending in `-k1om`, which provide executable binary files built to run only inside a MIC. Another set of packages with names beginning with `intel-mic-` is installed on the MIC host.

To install and configure MICs to work with DirectMon, the administrator:

- installs the MIC software using YUM or zypper;
- does the initial configuration of a set of MICs in a cluster using `cm-mic-setup` or `cmgui`;

The following can be done for the MIC, using `cmgui` or `cmsh`:

- tuning of coprocessor parameters. These parameters form the `mic<`*`INDEX`*`>.conf` (section 14.3.1) configuration file.
- monitoring of metrics/health checks of MIC cards.
- configuring workload manager queues for MICs just as for regular nodes. For now, only Slurm supports this feature.
- addition/removal of NFS mount points using `cmsh` or `cmgui`.
- enabling/disabling third party software overlays loaded during MIC booting.
- tuning the network interface parameters.

In the preceding list, the value of <*`INDEX`*> is a number 0,1,2,...,31, and is called the *container index parameter* (section 3) of the MIC host. It corresponds to the MIC identification number of the card on the host. Thus the first card on the host has an <*`INDEX`*> value of 0, and the card is then identified as `mic0`.

14.2 MIC Software Installation

In order for MIC cards to function properly on a host hardware, large BAR (Base Address Registers) support (MMIO addressing greater than 4GB) must be enabled. The platform and/or BIOS vendor should be contacted to check if changing this setting is applicable to the platform used.

14.2.1 MIC Software Packages

As software for the MIC, Intel provides the Intel MIC Platform Software Stack (MPSS). This is made up of system software and libraries for low-level coprocessor management. Distribution-specific builds of the MPSS source packages are used by DirectMon to manage MIC cards, and are picked up from the DDN repositories. The DirectMon administrator should not pick up the MPSS packages from Intel directly.

MIC Software Packages: Distribution Naming Convention

Linux distributions currently supported by DirectMon are those based on RHEL 6.3 (CentOS 6.3 and Scientific Linux 6.3), RHEL 6.4, and SUSE Linux Enterprise Server 11 SP2. The packages suffix, which is either `-rhel6.3`, `-rhel6.4` or `-suse11.2`, identifies the MIC package that is to be installed appropriately.

Distribution Based On	Suffix On Package
`RHEL 6.3`	`-rhel6.3`
`RHEL 6.4`	`-rhel6.4`
`SLES 11 SP2`	`-suse11.2`

For example, the `intel-mic-cross` tar.gz package distributed with MPSS, is repackaged by DirectMon as rpms:

- `intel-mic-cross-rhel6.3`
- `intel-mic-cross-rhel6.4`
- `intel-mic-cross-suse11.2`

The suffix is omitted for convenience in this manual.

MIC Software Packages: Names

The full list of packages, without the distribution suffix tags, is:

- `intel-mic-cross`: Intel MIC GNU tool chain to perform cross compilation (for x86_64 and for k1om);
- `intel-mic-driver`: Intel MIC driver, built during host boot;
- `intel-mic-flash`: Intel MIC flash files, to flash the cards (section 14.4);
- `intel-mic-kerneldev`: Intel MIC kernel development tools and sources;
- `intel-mic-ofed`: OFED drivers and libraries for MIC. The OFED drivers on the running systems are built during host boot, from source packages packed inside the `intel-mic-ofed` package. The source packages are:

  - `intel-mic-ofed-kmod`: Host-side MIC InfiniBand Drivers
  - `intel-mic-ofed-libibscif`: Intel SCIF Userspace Driver
  - `intel-mic-ofed-ibpd`: InfiniBand Proxy Daemon

- `intel-mic-runtime`: Intel MIC runtime;
- `intel-mic-sdk`: Intel MIC SDK.
- `intel-mic-perf`: Benchmarks for measuring the performance of the Intel MIC cards.

The minimum set of packages to be installed in a host software image—or on a head node, if coprocessors are installed on the head node—consists of `intel-mic-cross`, `intel-mic-driver`, `intel-mic-ofed`, `intel-mic-runtime`, and `intel-mic-flash`.

DDN also provides a set of k1om pre-built packages, which are installed on the head node. They can be removed if Slurm is not used:

- `libgcrypt-k1om`
- `slurm-client-k1om`
- `munge-k1om`
- `libgpg-error-k1om`

An additional k1om pre-built package that is installed is `strace-k1om`. This can typically be used for debugging issues, and may be used by DDN support when tracing problems.

Because the k1om packages are installed in `/cm/shared/`, they are available to each node in the cluster, as well as to the MIC cards. For now, only the Slurm Workload Manager can be started within the MIC in DirectMon, and requires all of these k1om packages to be installed on the head node. The other workload managers that DirectMon supports are supported with the MIC, but cannot be started from within the MIC.

14.2.2 MIC Environment MIC Commands

The module (section 3.2) `intel/mic/runtime` can be used to add MIC environment variables:

```
module add intel/mic/runtime
```

These environment variables allow the following Intel MPSS commands, from the `intel-mic-runtime` and other `intel-mic-*` packages, to execute:

- `micinfo`: retrieves information about a MIC system configuration. If executed without a path to the firmware file or directory, then a compatible image is searched for inside `/opt/intel/mic/flash/`.
- `micflash`: updates the coprocessor flash, queries the flash version on the coprocessor, and saves a copy of the flash image loaded in the coprocessor to a directory on the host. If `-device all` is specified as an option, then the firmware of all MIC cards on a host will be updated. Available options are described in `man(1) micflash`.

- `micsmc`: monitors coprocessors.
- `micctrl`: controls and configures a MIC.
- `miccheck`: verifies a coprocessor system configuration by running a diagnostic test suite.
- `micnativeloadex`: copies a MIC native binary to a specified coprocessor and executes it.
- `micrasd`: logs hardware errors reported by MIC devices.

14.2.3 DDN MIC Tools

The DDN repository also provides:

- `cm-mic-setup`: This tool is provided by the `cluster-tools` package. This configures MICs in a DirectMon enabled cluster, and is described further in section 14.3.1
- `cm-mic-postcode`: This tool is provided by the `cluster-tools-slave` package. This returns a human-readable description of the POST code returned by the coprocessor during power on and boot, thus identifying the stage that the card is at during the boot process. A hex representation of the MIC POST code can be found by reading the `/sys/class/mic/mic<`*`INDEX`*`>/post_code file.`

14.2.4 MIC OFED Installation

OFED communication can improve data throughput because there is then no additional copy into the host system memory. Direct data transfers can also take place on the SCIF (Symmetric Communications Interface) between the MIC OFED HCA driver on the host machine and the corresponding MIC OFED HCA driver on the MIC.

For MICs that are on the same host, the MIC OFED kernel modules can be used to enable MIC-to-MIC RDMA communication over OFED, if there is a supported OFED hardware on the host.

There are 3 widely-used OFED software stack implementations: OFA, QLogic, and Mellanox. Currently, Mellanox is not supported for MICs by Intel.

Packages Installation

By default, the parent distribution version of the OFED stack is installed on DirectMon. A QLogic OFED stack packaged by DDN can also be used. Its installation on the host is described in section 13.6. To use a MIC with OFED, however, requires the installation of a MIC-capable OFED stack, provided by the Intel `intel-mic-ofed` package listed previously (section 14.2.1).

To install the `intel-mic-ofed` package on a host requires that the `kernel-ib-devel` and `kernel-ib` OFED packages, if they are installed, must first be removed manually. They can be removed as follows:

- For a head node:

```
rpm -e kernel-ib-devel kernel-ib
```

- for a regular node with image <*image*>:

  ```
  rpm -e kernel-ib-devel kernel-ib --root=/cm/images/<image>
  ```

Currently the Intel Xeon Phi requires an OFED stack version of 1.5.4.1. The distribution stacks will work (provide RDMA support) if they satisfy this condition. DirectMon provides the optional QLogic OFED stack, which is based on OFA (OpenFabrics Alliance) OFED 1.5.4.1, and which works.

The manual installation of the OFA OFED stack, as described at `http://www.openfabrics.org`, works for SLES11SP2, and also works for RHEL and compatible versions up to 6.3. For version 6.4, for QLogic hardware, the distribution version of the OFED stack works, and it may be possible to have Mellanox hardware work using the OFA OFED stack.

After installation of the MIC OFED stack, the host must be rebooted to implement it. The MIC OFED stack is compiled and installed on boot.

The `ofed-mic` Service

The `ofed-mic` service running on the MIC host serves the MIC OFED kernel modules. If one of the MICs belonging to that host is not in an UP state, then:

- The `ofed-mic` service will fail. That is, its return code will not be 0, and the `ManagedServicesOK` healthcheck (page 647) response is `FAIL`.
- No attempt to start the `ofed-mic` service on a host is carried out by DirectMon during this `FAIL` state.

14.3 MIC Configuration

14.3.1 Using `cm-mic-setup` To Configure MICs

Introduction

The `cm-mic-setup` utility is used to set up the initial configuration of MIC hosts and cards in DirectMon. Adding new cards is done interactively by default, or otherwise automatically.

- Everything done by `cm-mic-setup` can be done manually using `cmsh`. However, using `cmsh` is not recommended because it is easy for an administrator to forget something and make a mistake. Configuring MICs via `cmsh` is covered in section 14.3.2.
- A `cmgui` alternative to `cm-mic-setup` is to use the `Create MICs` wizard (page 526).
- For SLES11SP2 and SLES11SP3, the `netaddr` Python module needs to be added explicitly for `cm-mic-setup` to work. It can be installed as follows:

  Example

  ```
  [root@ddnmon61 ~]# zypper install python-setuptools
  ...
  [root@ddnmon61 ~]# easy_install netaddr
  ...
  ```

Running The Command

A command synopsis of `cm-mic-setup` is:

```
cm-mic-setup --add-mic <MIC>... --mic-host <MICHOSTNAME>... [OPTIONS]
```

Here, `<MIC>` is the MIC card, and `<MICHOSTNAME>` is the MIC host, as specified according to the *hostname range format*. The details of the command syntax and hostname range format are given in the manual page, `man(8) cm-mic-setup`.

- In interactive mode, questions are asked about the configuration settings of particular card(s) and host(s) (IP addresses of host and card, bridge name and so on).
- In automatic mode, all parameters are configured by default, except that default values can be overwritten using extra options in the command line. The log file of all `cm-mic-setup` actions is kept in `/var/log/cm-mic-setup.log.`

When adding a new MIC, `cm-mic-setup` goes through these steps:

1. The new MIC device is added to DirectMon
2. If it has not already been done
   - a network bridge interface is added to a MIC host interface
   - the management physical network interface is added to the bridge
   - the physical network interface IP address is assigned to the bridge
   - and the IP address of the physical network interface is cleared.
3. A range of free IP addresses can be assigned to the MIC card interfaces for a specific MIC host (`mic0`, `mic1`, `mic2`...). The ranges are calculated using
   - a host management network base address (x.y.0.0)
   - network base offset (typically 0.0.z.0, with z $\neq$ 0. The offset must be specified for netmasks other than a CIDR value of /16)
   - and stride (s).

   The dotted quad representation of the range is then as indicated by the series:

   ```
     x.y.z.(s*n), x.y.z.(s*n+1), x.y.z.(s*n+2), ... x.y.z.(s*n+(s-1))
   = x.y.z.(s*n), ... x.y.z.(s*(n+1)-1)
   ```

   In the series, n is a number (0,1,2,3...) identified with the MIC host. The stride is thus the number of addresses in the range associated with that MIC host. The first line shows that the range starts here from `s*n+0` and ends at `s*n+(s-1)` in the end quad of the IP address. The MIC interfaces are given these IP addresses by default. The stride is thus a way of regularizing the IP addresses of the MIC cards associated with a MIC host.

A worked example of this follows: If the host management network is the internal network, then its base address is `10.141.0.0` with a netmask of in CIDR notation of /16, and the network base offset defaults to `0.0.128.0`. For a stride of `8`, the IP address range that will be allocated for the second (i.e., `n=1`) node is:

```
   x.y.z.(s*n) ... x.y.z(s*(n+1)-1)
= 10.141.128.8 ... 10.141.128.15
```

Here, `10.141.128.8` is the first MIC card IP address at the second node, and `10.141.128.9` is the next MIC card IP address at the second node.

If the stride is not specified, then a default stride is calculated as the maximum taken from the numbers of MICs per MIC host.

4. The `mic`*<INDEX>* physical network interface is added to the MIC host interfaces (i.e. on the regular nodes). The value of *<INDEX>* is typically the "number" of the MIC card on the host, starting with `0` for the first card, `1` for the second and so on. An example of adding a network interface is shown on page 106.

```
[ddnmon61->device[node001]->interfaces]% add physical mic0
```

5. The `mic0` physical network interface is added to the MIC card interfaces (i.e., to the MIC card nodes). The IP address is picked up from the previously calculated IP range to the interface in the earlier step 3.

```
[ddnmon61->device[node001-mic0]->interfaces% add physical mic0
```

6. Default filesystem mount points are added to the MIC device. These are `/dev/pts` for pts, `/dev/shm` for tempfs, `/sys` for sysfs, `/proc` for proc, and `/cm/shared` and `/home` for NFS.

7. Default overlays are added to the MIC. These are `cm-mounts` and a set of workload manager overlays. For Slurm the overlays `cm-munge` and `cm-slurm` are added.

8. The `MICHost` role is assigned to the MIC host.

9. IP forwarding is enabled on the MIC host. This is done by setting `net.ipv4.ip_forward=1` in the `/etc/sysctl.conf` configuration file using a host finalize script.

A reboot of the MIC hosts is prompted for in the interactive mode. The non-interactive mode does not prompt for a reboot, and does not carry out a reboot. A reboot of the MIC hosts is however needed after `cm-mic-setup` is run, for the MIC hosts to function correctly.

14.3.2 Using `cmsh` To Configure Some MIC Properties

A MIC is a first class device in DirectMon, which means it is very similar to a regular cluster node, rather than like, for example, a relatively dumb switch. So, when a new card is added, a cluster administrator can manage the MIC card with `cmsh` like the other nodes in the cluster:

1. MIC network interfaces can be managed from the `interfaces` submode:

   ```
   [ddnmon61->device[node002-mic0]->interfaces]% list
   Type      Network device name  IP            Network
   --------- -------------------- ------------- -----------
   physical  mic0                 10.141.128.4  internalnet
   ```

2. A MIC has a category type, `MIC`, associated with it. This is similar to nodes, which have a category type of `node`. MIC categories are analogous to node categories (section 3.1.3), and are therefore accessed from the `category` mode of `cmsh` . The idea behind a MIC category is essentially the same as that for a node category: to manage configuration settings in the cluster in a single place (i.e. in `cmgui` or `cmsh`) for a group of items of a particular type (i.e. nodes or MIC cards).

   By default, the `mic-default` MIC category is created when the `cm-mic-setup` script or the MIC wizard from `cmgui` is run for the first time. Within the `mic-default` category are defined, among others:

   - the default file system mounts (`fsmounts`)
   - MIC settings (`micsettings`)
     - overlays (`overlay`)
     - powermanagement (`powermanagement`)
   - roles (`roles`).

   Just as for regular node categories, properties are inherited by default from the category. Thus, if some property is set in `micsettings`, then this property is used by default in the particular MIC item unless it is overwritten.

3. MIC-specific options can be tuned in the `micsettings` submode of the `device` mode or `category` mode:

   ```
   [ddnmon61->category[mic-default]->micsettings]% show
   Parameter                     Value
   ----------------------------- ------------------------------------
   Boot On Start                 yes
   Console                       hvc0
   Crash Dump Dir                /var/crash/mic
   Crash Dump Limit              16
   Extra Command Line
   Overlays                      <3 in submode>
   Power Management              cpufreq_on;corec6_off;pc3_on;pc6_off
   Revision
   Root Device                   /opt/intel/mic/filesystem/mic${index\
   ```

```
                                }.image
Root Device Type                ramfs
Shutdown Timeout                300
Type                            XeonPhiSettings
User Authentication High UID    65000
User Authentication Low UID     500
User Authentication Type        Local
Verbose Logging                 yes
Base Dir                        /opt/intel/mic/filesystem/base
Base Dir File List              /opt/intel/mic/filesystem/base.file\
                                list
Common Dir                      /opt/intel/mic/filesystem/common
Common Dir File List            /opt/intel/mic/filesystem/common.fi\
                                lelist
Mic Dir                         /opt/intel/mic/filesystem/mic${inde\
                                x}
Mic Dir File List               /opt/intel/mic/filesystem/mic${inde\
                                x}.filelist
OS Image                        /lib/firmware/mic/uos.img
```

In the preceding listing, the text `${index}` is the container index parameter of the MIC node (section 14.1). The text is replaced by the value of when the MIC identification number is set for the card on the host.

Example

```
[ddnmon61]% device use node001-mic0
[ddnmon61->device[node001-mic0]]% get containerindex
0
```

4. Overlays which are used to enable and disable third party software in the MIC can be changed from the `overlays` submode of the `micsettings` submode.

5. Mount points can be configured just like for regular node mount points, but with one exception: If a directory is to be mounted from a MIC host to its own MIC, then the `$michost` variable can be used. On MIC boot this variable is substituted with the hostname of the MIC host.

All other MIC node parameters are similar to the parameters of a regular node.

14.3.3 Using `cmgui` To Configure Some MIC Properties

The `cmgui` front end treats a MIC very like a regular node. The differences are highlighted in the following:

- The `Overview` tabbed pane is the default view if the MIC name is clicked upon in the resource tree The `Overview` information for a MIC is displayed similar to that for regular nodes. The `Running Jobs` pane for the MIC lists only jobs executed inside the MIC (that is, for k1om applications, explained in section 14.5.2). Jobs executed on the MIC host, but offloaded to the MIC (offloaded applications, explained in section 14.5.2), are not shown here, but are instead listed in the `Running Jobs` pane of the MIC host.

- The MIC `Tasks` tab (figure 14.1) lets the administrator carry out some of the regular tasks operations for a node. This includes power cycling operations, adding or removing the MIC from node groups, draining and undraining (Appendix H.3.1) the device, and accessing the MIC card via `ssh` or the serial console using `minicom` running on the host.

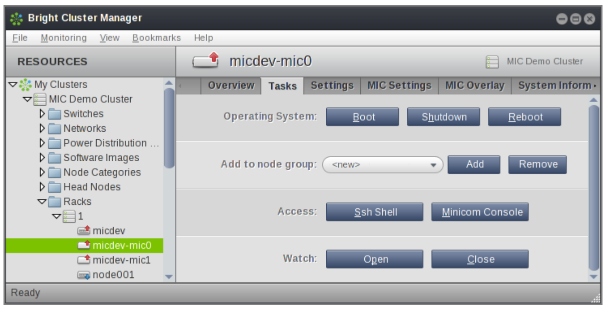
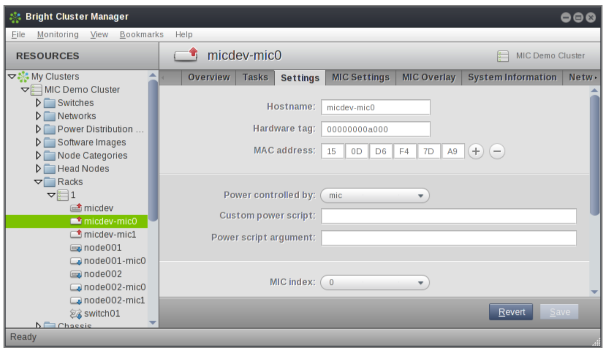


Figure 14.1: `cmgui`: The `MIC Tasks` Tab

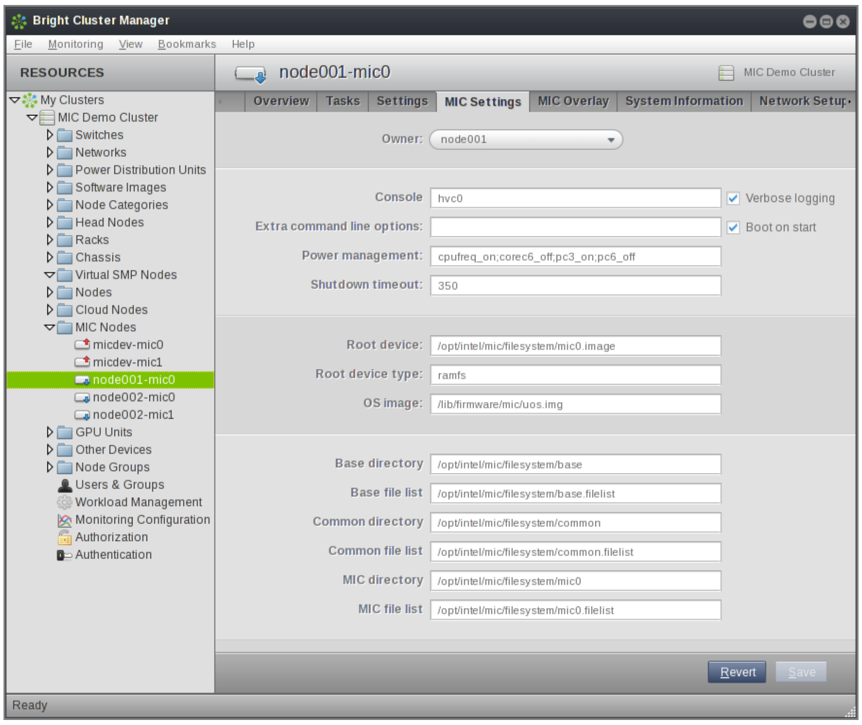
- The `Settings` tab (figure 14.2) allows configuration of common node parameters `MIC index` here is number of the coprocessor, and should be unique for the host. This is the same as the value of <*INDEX*>, which was introduced in section 14.1 as the MIC identification number set for the card on the host.



Hostname: micdev-mic0
Hardware tag: 00000000a000
MAC address: 15 0D D6 F4 7D A9
Power controlled by: mic
Custom power script:
Power script argument:
MIC index: 0

Figure 14.2: `cmgui`: The `Settings` Tab For A MIC

- The `MIC Settings` tab displays MIC-specific parameters (figure 14.3).

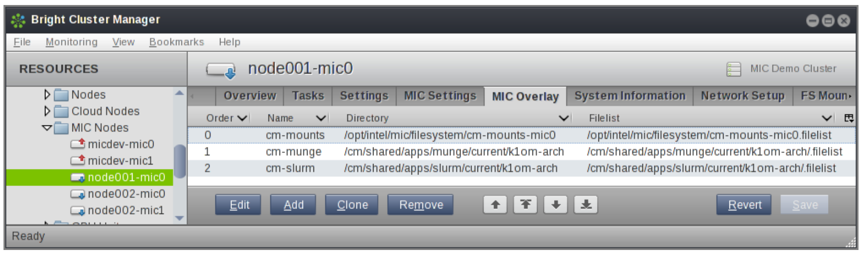


Owner:	node001
Console	hvc0
Extra command line options:	
Power management:	cpufreq_on;corec6_off;pc3_on;pc6_off
Shutdown timeout:	350
Verbose logging	✓
Boot on start	✓
Root device:	/opt/intel/mic/filesystem/mic0.image
Root device type:	ramfs
OS image:	/lib/firmware/mic/uos.img
Base directory	/opt/intel/mic/filesystem/base
Base file list	/opt/intel/mic/filesystem/base.filelist
Common directory	/opt/intel/mic/filesystem/common
Common file list	/opt/intel/mic/filesystem/common.filelist
MIC directory	/opt/intel/mic/filesystem/mic0
MIC file list	/opt/intel/mic/filesystem/mic0.filelist

Figure 14.3: `cmgui`: The `MIC Settings` Tab For A MIC

When the parameters of this tab are saved, then `/etc/sysconfig/mic/mic<`*`INDEX`*`>.conf` is updated on the host and the MIC device marked with a restart required flag. The corresponding message is: "`configuration has changed`". CMDaemon never restarts the MPSS service or individual MICs automatically, so to apply changes an administrator must reboot the coprocessor using "`power reset`" in `cmsh`, or the `Reboot` button within the `Tasks` tab of the MIC in `cmgui`. The rebooting of a MIC card can take several minutes. During the reboot, the state of the card is DOWN and it is inaccessible using `ssh`.

- The `MIC Overlay` tab (figure 14.4) allows overlays to be configured, so that third party software can be enabled or disabled.



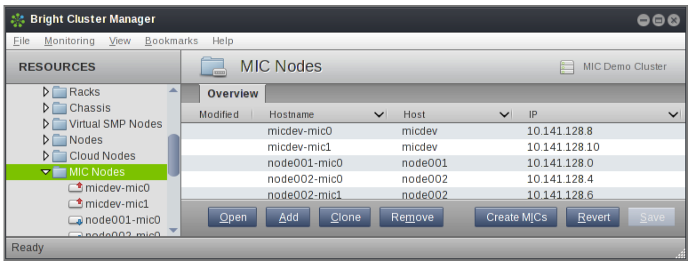
Order	Name	Directory	Filelist
0	cm-mounts	/opt/intel/mic/filesystem/cm-mounts-mic0	/opt/intel/mic/filesystem/cm-mounts-mic0.filelist
1	cm-munge	/cm/shared/apps/munge/current/k1om-arch	/cm/shared/apps/munge/current/k1om-arch/.filelist
2	cm-slurm	/cm/shared/apps/slurm/current/k1om-arch	/cm/shared/apps/slurm/current/k1om-arch/.filelist

Figure 14.4: `cmgui`: The `MIC Overlay` Tab

Each overlay has two important parameters: `Directory` and `Filelist`. The directory entry is a top directory where software files, links and directories are located. The filelist entry is a path to a file whose contents include files, directories, links.

Overlays and their parameters are explained in greater detail in section 14.3.4.

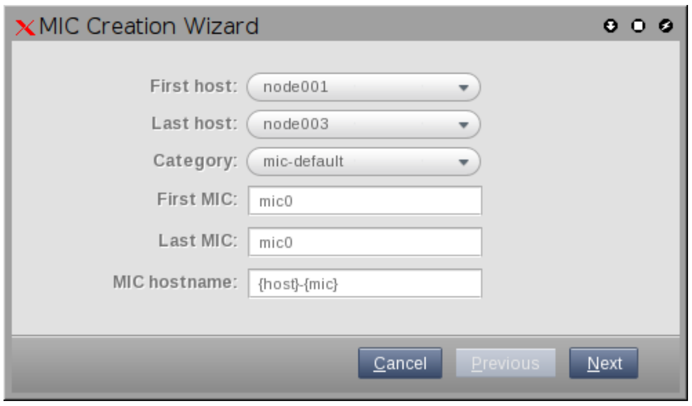
- The `FS Mounts` tab allows the mounting of filesystems to the MIC. To apply changes, the MIC card should be rebooted.
- Instead of using the `cm-mic-setup` script (section 14.3.1) to add new pre-configured MIC cards it is also possible to use the `cmgui` MIC wizard. The wizard is launched by clicking on the "`Create MICs`" button in the `Overview` tab that is displayed when the `MIC Nodes` resource is selected (figure 14.5).



Modified	Hostname	Host	IP
	micdev-mic0	micdev	10.141.128.8
	micdev-mic1	micdev	10.141.128.10
	node001-mic0	node001	10.141.128.0
	node002-mic0	node002	10.141.128.4
	node002-mic1	node002	10.141.128.6

Figure 14.5: `cmgui`: MIC Nodes `Overview` Tab

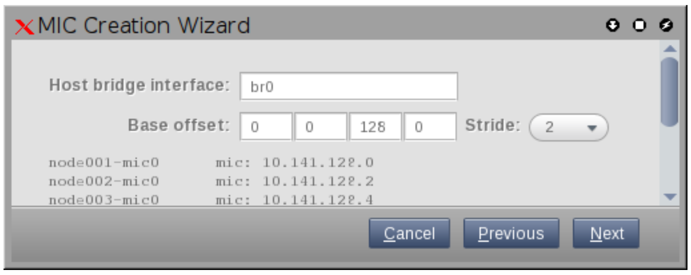
1. When the "`Create MICs`" button is clicked, the administrator can select a range of hosts on which the MIC cards are installed. The administrator can also set the name and range of the MIC suffixes to be used for the MICs. The default pattern is `mic`<*INDEX*>. A MIC hostname template is suggested using a the MIC host hostnames and MIC card names as parameters.



First host:	node001
Last host:	node003
Category:	mic-default
First MIC:	mic0
Last MIC:	mic0
MIC hostname:	{host}-{mic}

Figure 14.6: `cmgui`: MIC Creation Wizard: Hostname Screen

2. Clicking on the `Next` button brings the administrator to the second step of the wizard. This allows the administrator to set the host network bridge interface, the network base offset, and stride. A default stride value is calculated as the maximum taken from the numbers of MICs per MIC host. The IP addresses used for the MIC devices are listed. The list of IP addresses in this window change dynamically as changes are made to the base offset or stride.



Host bridge interface: br0

Base offset: 0 0 128 0 Stride: 2

```
node001-mic0       mic: 10.141.128.0
node002-mic0       mic: 10.141.128.2
node003-mic0       mic: 10.141.128.4
```

Figure 14.7: `cmgui`: MIC Creation Wizard: Suggested IP Addresses Screen

3. Clicking on the "Next" button brings up the final window. This window shows warnings. If the changes are acceptable, then the `Finish` button can be clicked. The MICs are then added to DirectMon.

14.3.4 Using MIC Overlays To Place Software On The MIC

A MIC overlay is used to make software available from a MIC host to a MIC. Usually this is software that is heavily accessed during a code run, such as drivers or parts of applications where access speed is a priority. It can also be software that can only be executed on the MIC, such as init scripts. The memory available to the MIC (8 to 16 GB at the time of writing in November 2013) is much lower than that of a regular node, so that large amounts of data cannot be placed in it. A relatively much slower remote-mounted filesystem is normally what is used instead for access to larger amounts of data.

Ignoring mounts, the MIC filesystem on the MIC is built up in several layers, starting with the fundamental binaries layer, adding some more required layers, and ending with a software overlay that the administrator installs for the end user applications. Each layer on the MIC is configured by using a "layer" directory on the MIC host. The directory used is assigned a corresponding *directory configuration parameter*. The parameter is one of `BaseDir`, `CommonDir`, `MicDir`, and `Overlay`. Each of these parameters has a default setting, but if needed, the administrator can modify them via the CMDaemon front ends, `cmsh` and `cmgui`.

Each directory assigned to its parameter contains a *filelist* descriptor file, typically given a name ending in `.filelist`. The filelist content describes the files, directories, links, that are to be copied over from the layer directory on the MIC host, and laid out in the MIC filesystem on the MIC. The filelist content also includes *nods*. Nods (with a nod to the `mknod(1)` command) are used to set , for special devices on the MIC, such as `/dev/console`:

- the device type: character `c` or block, `b`
- the `major` and `minor` values
- `perms`, `uid` and `gid`.

The canonical details on these and other parameters at the time of writing (November 2013) can be found in Intel document number 328344-001US, provided with each MPSS release: *Intel Xeon Phi Coprocessor Many-core Platform Software Stack (MPSS) Boot Configuration Guide*. In revision

0.6 of that guide the details are found in section 4.4.1: *File Location Parameters*, and section 6.1: *The File System Creation Process*.

An example syntax for an overlay can be seen in `/etc/sysconfig/mic/conf.d/coi.conf`.

The layers are laid out in the MIC according to the following sequence of directory configuration parameters:

1. `BaseDir`: This MIC host directory contains the basic MIC Coprocessor binaries that were installed during RPM installation. Files added by the user in this directory will be wiped out during MPSS installation updates.

2. `CommonDir`: This MIC host directory contains a common filesystem structure that does not change during use of the MIC for compute purposes. Files in this directory are not wiped out during MPSS installation updates.

3. `MicDir`: This MIC host directory contains per MIC information files. Most of these files are created during MIC configuration. These values do not change if the MIC hardware does not change. However restarting the mpssd daemon will reconfigure the values, so changing them by hand is not advisable.

4. `Overlay`: This MIC host directory contains additional sets of files that may be laid over the MIC file system that has been built in the layers so far. More than one `Overlay` parameter can be assigned by an administrator, who can thus use MIC overlays to make application software available to the MIC.

   - The default assignment is named `cm-mounts`. The associated directory and filelist are configured with default values when `cm-mic-setup` (section 14.3.1, step 7) or the `Create MICs` wizard (page 526) is used. A MIC using these defaults creates the default mount points for the NFS directories.
   - Slurm is offered as a configuration choice when `cm-mic-setup` or `Create MICs` are run. If selected, then the names `cm-munge` and `cm-slurm` are also assigned to the `Overlay` parameter, and their associated directories and filelists are configured with default values too. Among others, these overlays place `slurm` and `munge` init scripts, and also the configuration file `gres.conf`, inside the MICs.
   - The optional `bash-k1om` package provides the `bash-k1om` overlay, which allows the Bash shell to be used inside MICs.

   These overlays have the following default directory and filelist settings:

```
[ddnmon61->category[mic-default]->micsettings->overlay]% list -f \
name:10,directory:47
name (key) directory
---------- -----------------------------------------------
cm-mounts  /opt/intel/mic/filesystem/cm-mounts-mic${index}
cm-munge   /cm/shared/apps/munge/current/k1om-arch
cm-slurm   /cm/shared/apps/slurm/current/k1om-arch
```

```
bash-k1om  /cm/shared/apps/intel-mic/current/native

[ddnmon61->category[mic-default]->micsettings->overlay]% list -f \
name:10,filelist:56
name (key) filelist
---------- --------------------------------------------------------
cm-mounts  /opt/intel/mic/filesystem/cm-mounts-mic${index}.filelist
cm-munge   /cm/shared/apps/munge/current/k1om-arch/munge.filelist
cm-slurm   /cm/shared/apps/slurm/current/k1om-arch/slurm.filelist
bash-k1om  /home/cmsupport/bash-4.1-k1om.filelist
```

The administrator generally only needs to consider the overlay layer(s), because MPSS and DirectMon configuration takes care of the other "OS-like" layers with sensible default values. Thus, adding a MIC application as an overlay controlled by DirectMon can be done, for example, according to the following procedure:

Example

1. A new layer directory can be created on the MIC host:
   `mkdir /cm/local/apps/myapp/k1om/`
2. Files, directories, and links for the MIC application can then be placed inside the layer directory:
   `cp -r k1omapplication /cm/local/apps/myapp/k1om/`
3. A descriptor file, `myapp.filelist`, can be constructed according to the filelist syntax, and placed in the layer directory's topmost level.
4. A new overlay object is created via `cmsh` or `cmgui`, with the `Overlay` directory configuration parameter set to `/cm/local/apps/myapp/k1om/`, and the filelist descriptor set to `/cm/local/apps/myapp/k1om/myapp.filelist`
5. On restarting the MIC, the MIC application becomes available on the MIC.

14.4 MIC Card Flash Updates

After all the necessary `intel-mic-*` and `*-k1om` packages have been installed for the first time, the hosts should be rebooted to make the MIC commands available for use. The card flash images should then be updated, and the hosts rebooted after that too.

The flash update procedure on, for example, two MIC hosts `node001` and `node002`, can be carried out as follows:

Backslashes in the following 5-step procedure indicate line splits, which means that any commands running across such line splits should be input as one long line:

1. The MPSS service on the MIC hosts is stopped:

```
[root@ddnmon61 ~]$ cmsh
[ddnmon61]% device
[ddnmon61->device]% foreach -n node001..node002 (services; stop mpss)
...
```

2. Using the `micctrl` command with appropriate options, the `ready` state is awaited (`-w`) on each MIC host after a forced (`-f`) reset (`-r`) of the host:

```
[ddnmon61->device]% pexec -n node001..node002 "module load
intel/mic/runtime && micctrl -r -f -w"

[node001] :
mic0: ready
mic1: ready

[node002] :
mic0: ready
mic1: ready

[ddnmon61->device]%
```

For the sake of clarity: The reset option brings a MIC to a `ready` state (ready for boot), which means it is `DOWN` in DirectMon terminology. The `-h` option shows an options help text for the `micctrl` command.

3. The firmware can then be updated. Some output has been elided for brevity in the listed output:

```
[ddnmon61->device]% pexec -n node001..node002 "module load\
 intel/mic/runtime && micflash -v -update -noreboot\
 -device all"
[node001]
No image path specified - Searching: /opt/intel/mic/flash
mic0: Flash image: /opt/intel/mic/flash/EXT_HP2_B0_0383-02.rom.smc
mic1: Flash image: /opt/intel/mic/flash/EXT_HP2_B0_0383-02.rom.smc
mic0: Resetting: : POST code: 3d
mic1: Resetting: : POST code: 3d
...
mic0: Resetting: : POST code: 12
mic1: Resetting: : POST code: 12
mic0: Updating SMC: 0%
mic1: Updating SMC: 0%
...
mic0: Updating SMC: 97%
mic1: Updating SMC: 96%
mic0: Resetting: : POST code: 3C
mic1: Resetting: : POST code: 3C
...
mic0: Resetting: : POST code: 09
mic1: Resetting: : POST code: 09
update successful

[node002] :
```

```
...
```

Some older MIC cards need a modified `micflash` command from that used in the preceding output. Intel designates MIC card core step versions with A0, A1, B0 and so on. For card versions older than C0, the `-smcbootloader` option must be added to the `micflash` command. This brings the SMC bootloader software up to or beyond version 1.8, which is a requirement.

4. The MPSS service is started:

```
[ddnmon61->device]% foreach -n node001..node002 (services; start\
 mpss)
```

5. The hosts are rebooted:

```
[ddnmon61->device]% foreach -n node001..node002 (power reset)
```

Rebooting the hosts also automatically reboots the MICs.

The preceding 5-step procedure can also be carried out from within the `Parallel Shell` in `cmgui` (section 12.1.3).

Further details on flashing the MIC are given in the *Intel Xeon Phi Coprocessor Many-core Platform Software Stack (MPSS) Getting Started Guide*.

14.5 Other MIC Administrative Tasks

After the procedures of installing and configuring the MPSS software, and re-flashing the MIC, the DirectMon is in an understandably disturbed state. It should be allowed about a 10 to 15 minute period to settle down, so that it can start populating its metrics and health checks with sensible values. Warnings during the settling down period can be ignored. After the system has settled down to a steady state, any persistent warnings can be taken more seriously.

14.5.1 How CMDaemon Manages MIC Cards

When the `michost` role is assigned to a host, CMDaemon starts and monitors these services:

- `mpss` services. This is an Intel Manycore Platform Software Stack (Intel MPSS) service which runs the mpssd daemon. The currently supported version of MPSS is 2.1.6720.

  The mpssd daemon controls the initialization and booting of coprocessor cards based on a set of configuration files located at `/etc/sysconfig/mic/`. The daemon is started and stopped as an operating system service and instructs the cards to boot or shutdown. It supplies the final file system image to the cards when requested. Further details about the mpssd daemon and its configuration files are given in the *Intel Xeon Phi Coprocessor Many-core Platform Software Stack (MPSS) Getting Started Guide*.

- The `ofed-mic` service. This provides the OFED interface between a coprocessor and ofed drivers running on its host.

Near the end of a successful boot of the coprocessor, CMDaemon executes the finalize script `/cm/local/apps/cmd/scripts/finalize-mic` for the MIC. The finalize script prepares the card to be used by users: It clears the IP address of the MIC interface, adds the MIC interface to to the network bridge interface, adds the workload manager user (if necessary), and finally executes the `prepare-mic` script on the card via `ssh`.

The `prepare-mic` script mounts NFS shared directories, starts workload manager daemons, and makes sure that `coi_daemon` is running with `micuser` user permissions. The script also forbids all regular users from accessing the MIC via SSH, apart from the user running a job via the job prolog, or a job running as `micuser`. The `root` user always has access to the MIC card.

14.5.2 Using Workload Managers With MIC

After a new MIC card is added and the `michost` role is assigned to its host, CMDaemon can configure the workload manager that is currently in use to use this card as follows:

- For Slurm:
  - The MIC is added to the workload manager as a new computing node.
  - The MIC is added to job queues if the workload manager client role is assigned to the MIC in `cmsh` or `cmgui`.
  - Node properties `michost` and `miccard` are assigned to the MIC host and the MIC card correspondingly.
  - A special prolog script is enabled for the host, if it has not already been enabled. The prolog script is at `/cm/local/apps/cmd/scripts/prolog-michost`.
- For all workload managers:
  - A new generic resource type `mic` is added, and the workload manager daemon on the host is configured to be aware about new available generic resources.

DirectMon supports two main use case scenarios for coprocessors:

1. Native k1om application.

   This currently only works with Slurm.

   The user's application is built to be used on k1om CPU architecture and executed inside MIC card. In this case the user should request nodes with the property `miccard` on its job submission. The workload manager processes this job just like a regular job executed inside the regular node. The user should remember that the job script should use the `/bin/sh` interpreter, which in turn invokes the `ash` shell in a BusyBox environment.

   Before the job is executed, the MIC hosts prolog script (`/cm/local/apps/cmd/scripts/prolog-michost`) is invoked on all hosts which are used by the job, in order to prepare MIC cards for the job.

2. Code offloading by the application.

   To carry out code offloading, the user's application is created to be executed on an x86_64 host, but uses special instructions to offload part of its code to one or more coprocessors installed on the host locally. The user needs to request the `michost` property or a particular number of MIC generic resources for each host during its job submission.

   The application is executed on specified hosts and the workload manager sets an OFFLOAD_DEVICES environment variable, which controls the selection of MICs available to a job's offloading code.

   The offloaded code is executed with the user's permissions inside the MICs. If no MICs are configured on the host, the OFFLOAD_DEVICES variable is set to -1. This causes the code to ignore the offload directives and run its routines on the host's CPU(s).

   Before the job is executed, the MIC prolog script (`/cm/local/apps/cmd/scripts/prolog-mic`) is invoked on MIC to prepare the card for the job.

In both cases, prolog scripts eventually execute the `prepare-mic` script (`/cm/local/apps/cmd/scripts/prepare-mic`), located on the MIC (the same script as executed within finalize script). This script prepares the card for use by a particular user.

14.5.3 Mounting The Root Filesystem For A MIC Over NFS

In section 4.11.4, the configuration of a diskless node with its root filesystem provided over NFS was described.

This section (section 14.5.3), assumes that the steps described in section 4.11.4 have been carried out, so that a diskless node is now up and running. If a MIC is installed on such a diskless host, then this host, which is now a MIC host, can be configured so that NFS provides it with a root filesystem, as follows:

1. The MIC and OFED MIC drivers should be built and installed in an appropriate software image. This is to reduce the number of directories mounted to the tmpfs filesystem.

   The drivers can be built and installed in a software image called `root-over-nfs` as follows:

   Example

```
[root@ddnmon61 ~]# chroot /cm/images/root-over-nfs
[root@ddnmon61 /]# chkconfig --del intel-mic-driver
[root@ddnmon61 /]# chkconfig --del intel-mic-ofed
[root@ddnmon61 /]# /etc/init.d/intel-mic-driver install
Building and installing mic driver                       [  OK  ]
[root@ddnmon61 /]# /etc/init.d/intel-mic-ofed install
Building and installing intel-mic-ofed-kmod              [  OK  ]
Building and installing intel-mic-ofed-ibpd              [  OK  ]
Building and installing intel-mic-ofed-libibscif         [  OK  ]
```

The `intel-mic-ofed` command should not be run with a `start` argument when the drivers are installed inside the software image. This is because the script would then restart the openibd service.

Log entries for the install process can be seen under `/var/log`, in `intel-mic-driver.log`, `intel-mic-ofed-driver.log`, and `intel-mic-ofed-driver-build.log`.

2. The new filesystem mount point, accessible under the `fsmounts` submode, should be set to point to the appropriate node category. If the node category name is `root-over-nfs`, then the configuration can be done using `cmsh` as follows:

   Example

   ```
   [root@ddnmon61 ~]# cmsh
   [ddnmon61]% category fsmounts root-over-nfs
   [ddnmon61->category[root-over-nfs]->fsmounts]% add /cm/local/apps/\
   intel-mic/current/filesystem
   [ddnmon61->...intel-mic/current/filesystem*]]% set device tmpfs
   [ddnmon61->...intel-mic/current/filesystem*]]% set filesystem tmpfs
   [ddnmon61->...intel-mic/current/filesystem*]]% commit
   ```

3. The MIC host must be rebooted to implement the changes.

14.5.4 MIC Metrics

Metrics are listed in Appendix H.1.1. The use of the metric collection `mic` (page 639) means that MIC metrics with working sensors are able to monitor MIC-related values.

The exact metrics that can be monitored vary according to the MIC hardware.

14.5.5 User Management On The MIC

A current user in the context of MIC use is a user that starts a job via a workload manager. By default, a current user is added to the `micuser` group within the MIC, in `/etc/group`, and the value:

```
AllowGroups root micuser
```

is appended to `/etc/ssh/sshd_config` within the MIC at the same time. This then allows only users from `micuser` group and the `root` user to have ssh access to the MIC during job execution.

The default user management behavior can be altered in `cmsh` and `cmgui` by adding the following parameters to `mic<N>.conf`:

- `UserAuthenticationType`: user authentication type used for a MIC. Possible values: Local, None. Default: `Local`
- `UserAuthenticationLowUID`: lowest ID of a user which will be added. Possible values: any 32-bit positive number. Default: `500`
- `UserAuthenticationHighUID`: highest ID of a user which will be added. Possible values: any 32-bit positive number. Default: `65000`

Users with user IDs in the range:

`UserAuthenticationLowUID-UserAuthenticationHighUID`

will be added to the MIC. Although any 32-bit user ID may be entered, a value of `500` or more is recommended for the lowest value. If `UserAuthenticationType=None` is specified, the `/etc/passwd` file on the card will default to one containing the `root`, `sshd`, and `micuser` accounts.

In `/etc/sysconfig/mic/cm-mpss.conf`, the parameters that are used by MIC related scripts provided by DirectMon are:

- `SSHD_ALLOW_GROUPS`: indicates what will be set as the value to `AllowGroups` in `/etc/ssh/sshd_config` within all MICs of the host. Possible values are:
  - <*any quoted string*>: Default: `"root micuser"`
  - `UNUSED`: This value means that
    - `AllowGroups` will not be appended in `/etc/ssh/sshd_config`
    - the current user will not be added to the `micuser` group
    - all users can ssh into the MIC even if they have no job running on it.
- `RESTART_COI_DAEMON`: indicates if `coi_daemon` will be restarted with the current user permissions. If the value is `YES`, then `coi_daemon` restarts when the MIC is booted, and also each time when `prolog-mic` script is executed on MIC or the `prolog-michost` script is executed on the host. Default: `YES`

If `/etc/sysconfig/mic/cm-mpss.conf` is changed, then the MICs should be restarted to apply the changes.

LDAP support: The current version of Intel MPSS does not support LDAP. However, there is a workaround for that provided by the micctrl utility. When UserAuthenticationType is set to `Local`, and MIC is booted using `micctrl`, then the micctrl utility will attempt to find all users in the host's `/etc/passwd` file and `/home` directories. It will then populate the MIC file system with them. Thus, if the LDAP user home exists on the host, then the user and its group will be added within the MIC. However, the user's password will not be added, and in order to have access to the MIC the user must use ssh keys.

15 High Availability

In a cluster with a single head node, the head node is a single point of failure for the entire cluster. It is often unacceptable that the failure of a single machine can disrupt the daily operations of a cluster.

The high availability (HA) feature of DirectMon therefore allows clusters to be set up with two head nodes configured as a failover pair.

In this chapter:

- Section 15.1 describes the concepts behind HA, keeping the DirectMon configuration in mind.
- Section 15.2 describes the normal user-interactive way in which the DirectMon implementation of a failover setup is configured.
- Section 15.3 describes the implementation of the DirectMon failover setup in a less user-interactive way, which avoids using the Ncurses dialogs of section 15.2
- Section 15.4 describes how HA is managed with DirectMon after it has been set up.

15.1 HA Concepts

15.1.1 Primary, Secondary, Active, Passive

In a cluster with an HA setup, one of the head nodes is named the *primary* head node and the other head node is named the *secondary* head node. Under normal operation, one of the two head nodes is in *active* mode, whereas the other is in *passive* mode.

The difference between naming versus mode is illustrated by realizing that while a head node which is primary always remains primary, the mode that the node is in may change. Thus, the primary head node can be in passive mode when the secondary is in active mode. Similarly the primary head node may be in active mode while the secondary head node is in passive mode.

The difference between active and passive is that the active head takes the lead in cluster-related activity, while the passive follows it. Thus, for example, with MySQL transactions, CMDaemon carries them out with MySQL running on the active, while the passive trails the changes. This naturally means that the active corresponds to the master, and the passive to the slave, in the MySQL master-slave replication mode that MySQL is run as.

15.1.2 Monitoring The Active Head Node, Initiating Failover

In HA the passive head node continuously monitors the active head node. If the passive finds that the active is no longer operational, it will initiate a *failover sequence*. A failover sequence involves taking over resources, services and network addresses from the active head node. The goal is to continue providing services to compute nodes, so that jobs running on these nodes keep running.

15.1.3 Services In DirectMon HA Setups

There are several services being offered by a head node to the cluster and its users.

Services Running On Both Head Nodes

One of the design features of the HA implementation in DirectMon is that whenever possible, services are offered on both the active as well as the passive head node. This allows the capacity of both machines to be used for certain tasks (e.g. provisioning), but it also means that there are fewer services to move in the event of a failover sequence.

On a default HA setup, the following key services for cluster operations are always running on both head nodes:

- **CMDaemon**: providing certain functionality on both head nodes (e.g. provisioning)
- **DHCP**: load balanced setup
- **TFTP**: requests answered on demand, under xinetd
- **LDAP**: running in replication mode (the active head node LDAP database is pulled by the passive)
- **MySQL**: running in master-slave replication mode (the active head node MySQL database is pulled by the passive)
- **NTP**
- **DNS**

When an HA setup is created from a single head node setup, the above services are automatically reconfigured to run in the HA environment over two head nodes.

Provisioning role runs on both head nodes In addition, both head nodes also take up the *provisioning role*, which means that nodes can be provisioned from both head nodes. As the passive head node is then also provisioned from the active, and the active can switch between primary and secondary, it means both heads are also given a value for `provisioninginterface` (section 6.4.7).

For a head node in a single-headed setup, there is no value set by default. For head nodes in an HA setup, the value of `provisioninginterface` for each head node is automatically set up by default to the interface device name over which the image can be received when the head node is passive.

The implications of running a cluster with multiple provisioning nodes are described in detail in section 6.2. One important aspect described in that section is how to make provisioning nodes aware of image changes.

From the administrator's point of view, achieving awareness of image changes for provisioning nodes in HA clusters is dealt with in the same way as for single-headed clusters. Thus, if using `cmsh`, the `updateprovisioners` command from within `softwareimage` mode is used, while if using `cmgui`, it is done from the `Software Images` resource, then selecting the `Provisioning Status` tab, and clicking on the `Update Provisioning Nodes` button (section 6.2.4).

Services That Migrate To The Active Node

Although it is possible to configure any service to migrate from one head node to another in the event of a failover, in a typical HA setup only the following services migrate:

- NFS
- The User Portal
- Workload Management (e.g. SGE, Torque/Maui)

15.1.4 Failover Network Topology

A two-head failover network layout is illustrated in figure 15.1. In the illustration, the primary `head1` is originally a head node before the failover design is implemented. It is originally set up as part of a Type 1 network (section 2.3.6), with an internal interface eth0, and an external interface eth1.

When the secondary head is connected up to help form the failover system, several changes are made.

HA: Network Interfaces

Each head node in an HA setup typically has at least an external and an internal network interface, each configured with an IP address.

In addition, an HA setup uses two virtual IP interfaces, each of which has an associated virtual IP address: the external shared IP address and the internal shared IP address. These are shared between the head nodes, but only one head node can host the address and its interface at any time.

In a normal HA setup, a shared IP address has its interface hosted on the head node that is operating in active mode. On failover, the interface migrates and is hosted on the head node that then becomes active.

When head nodes are also being used as login nodes, users outside of the cluster are encouraged to use the shared external IP address for connecting to the cluster. This ensures that they always reach whichever head node is active. Similarly, inside the cluster, nodes use the shared internal IP address wherever possible for referring to the head node. For example, nodes mount NFS filesystems on the shared internal IP interface so that the imported filesystems continue to be accessible in the event of a failover.

Shared interfaces are implemented as alias interfaces on the physical interfaces (e.g. `eth0:0`). They are activated when a head node becomes active, and deactivated when a head node becomes passive.

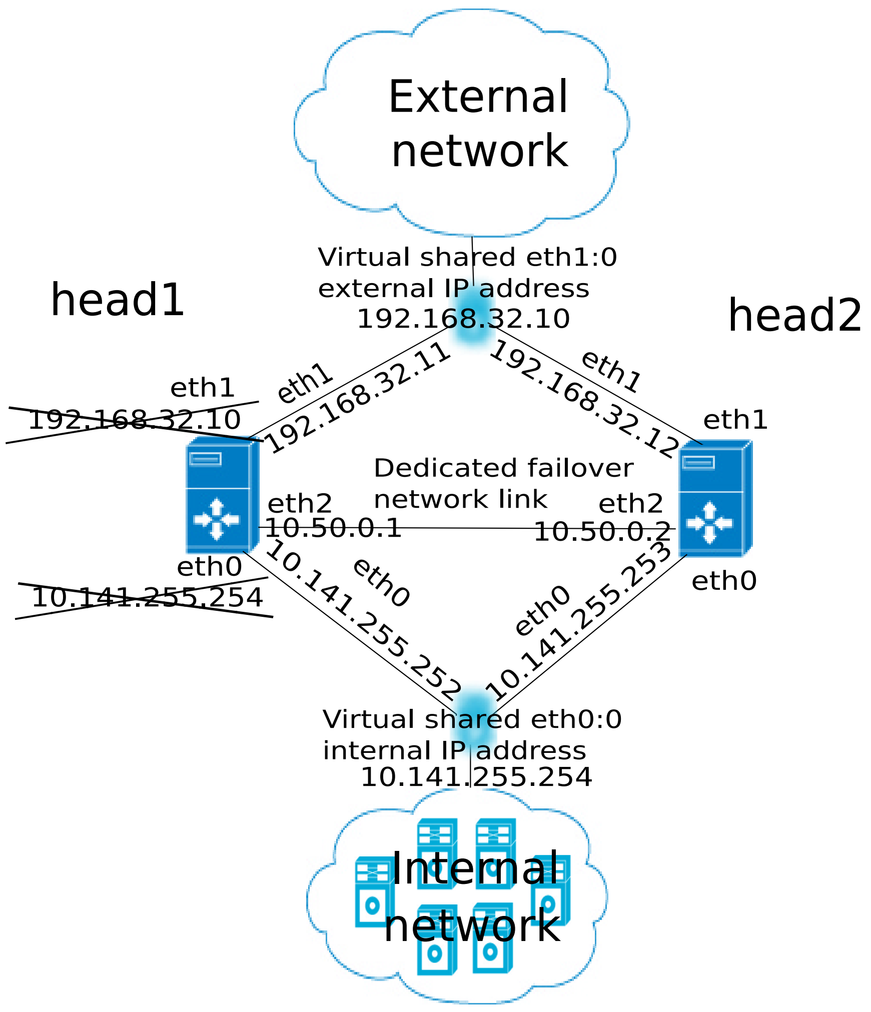


Figure 15.1: High Availability: Two-Head Failover Network Topology

HA: Dedicated Failover Network

In addition to the normal internal and external network interfaces on both head nodes, the two head nodes are usually also connected using a direct dedicated network connection, `eth2` in figure 15.1. This connection is used between the two head nodes to monitor their counterpart's availability. It is called a *heartbeat* connection because the monitoring is usually done with a regular heartbeat-like signal between the nodes such as a ping, and if the signal is not detected, it suggests a head node is dead.

To set up a failover network, it is highly recommended to simply run a UTP cable directly from the NIC of one head node to the NIC of the other, because not using a switch means there is no disruption of the connection in the event of a switch reset.

15.1.5 Shared Storage

Almost any HA setup also involves some form of shared storage between two head nodes to preserve state after a failover sequence. For example, user home directories must always be available to the cluster in the event of a failover.

In the most common HA setup, the following two directories are shared:

- User home directories (i.e. `/home`)

- Shared tree containing applications and libraries that are made available to the nodes (i.e. `/cm/shared`)

The shared filesystems are only available on the active head node. For this reason, it is generally recommended that users login via the shared IP address, rather than ever using the direct primary or secondary IP address. End-users logging into the passive head node by direct login may run into confusing behavior due to unmounted filesystems.

Although DirectMon gives the administrator full flexibility on how shared storage is implemented between two head nodes, there are generally three types of storage used: NAS, DAS and DRBD.

NAS

In a Network Attached Storage (NAS) setup, both head nodes mount a shared volume from an external network attached storage device. In the most common situation this would be an NFS server either inside or outside of the cluster.

Because imported mounts can typically not be re-exported (which is true at least for NFS), nodes typically mount filesystems directly from the NAS device.

DAS

In a Direct Attached Storage (DAS) setup, both head nodes share access to a block device that is usually accessed through a SCSI interface. This could be a disk-array that is connected to both head nodes, or it could be a block device that is exported by a corporate SAN infrastructure.

Although the block device is visible and can physically be accessed simultaneously on both head nodes, the filesystem that is used on the block device is typically not suited for simultaneous access. Simultaneous access to a filesystem from two head nodes must therefore be avoided because it generally leads to filesystem corruption. Only special purpose parallel filesystems such as GPFS and Lustre are capable of being accessed by two head nodes simultaneously.

DRBD

In a setup with DRBD (Distributed Replicated Block Device), both head nodes mirror a physical block device on each node device over a network interface. This results in a virtual shared DRBD block device. A DRBD block devices is effectively a DAS block device simulated via a network. DRBD is a cost-effective solution for implementing shared storage in an HA setup. While a DRBD device itself is not configured by the cluster manager, a DRBD device that is already configured is supported by the cluster manager.

Custom Shared Storage With Mount And Unmount Scripts

The cluster management daemon on the two head nodes deals with shared storage through a *mount script* and an *unmount script*. When a head node is moving to active mode, it must acquire the shared filesystems. To accomplish this, the other head node first needs to relinquish any shared filesystems that may still be mounted. After this has been done, the head node that is moving to active mode invokes the *mount script* which has been configured during the HA setup procedure. When an active head node is requested to become *passive* (e.g. because the administrator wants to take it down for maintenance without disrupting jobs), the *unmount script* is invoked to release all shared filesystems.

By customizing the *mount* and *unmount* scripts, an administrator has full control over the form of shared storage that is used. Also an administrator can control which filesystems are shared.

Mount scripts paths can be set via `cmsh` or `cmgui` (section 15.4.6).

15.1.6 Guaranteeing One Active Head At All Times

Because of the risks involved in accessing a shared filesystem simultaneously from two head nodes, it is vital that only one head node is in active mode at any time. To guarantee that a head node that is about to switch to active mode will be the only head node in active mode, it must either receive confirmation from the other head node that it is in passive mode, or it must make sure that the other head node is powered off.

What Is A Split Brain?

When the passive head node determines that the active head node is no longer reachable, it must also take into consideration that there could be a communication disruption between the two head nodes. Because the "brains" of the cluster are communicatively "split" from each other, this is called a *split brain* situation.

Since the normal communication channel between the passive and active may not be working correctly, it is not possible to use only that channel to determine either an inactive head or a split brain with certainty. It can only be suspected.

Thus, on the one hand, it is possible that the head node has, for example, completely crashed, becoming totally inactive and thereby causing the lack of response. On the other hand, it is also possible that, for example, a switch between both head nodes is malfunctioning, and that the active head node is still up and running, looking after the cluster as usual, and that the head node in turn observes that the passive head node seems to have split away from the network.

Further supporting evidence from the dedicated failover network channel is therefore helpful. Some administrators find this supporting evidence an acceptable level of certainty, and configure the cluster to decide to automatically proceed with the failover sequence, while others may instead wish to examine the situation first before manually proceeding with the failover sequence. The implementation of automatic vs manual failover is described in section 15.1.7. In either implementation, *fencing*, described next, takes place until the formerly active node is powered off.

Going Into Fencing Mode

To deal with a suspected inactive head or split brain, a passive head node that notices that its active counterpart is no longer responding, first goes into *fencing* mode from that time onwards. While a node is fencing, it will try to obtain proof via another method that its counterpart is indeed inactive.

Fencing, incidentally, does not refer to a thrust-and-parry imagery derived from fencing swordplay. Instead, it refers to the way all subsequent actions are tagged and effectively fenced-off as a backlog of actions to be carried out later. If the head nodes are able to communicate with each other before the passive decides that its counterpart is now inactive, then the fenced-off backlog is compared and synced until the head nodes are once again consistent.

Ensuring That The Unresponsive Active Is Indeed Inactive

There are two ways in which "proof" can be obtained that an unresponsive active is inactive:

1. By asking the administrator to manually confirm that the active head node is indeed powered off
2. By performing a power-off operation on the active head node, and then checking that the power is indeed off. This is also referred to as a STONITH (Shoot The Other Node In The Head) procedure

Once a guarantee has been obtained that the active head node is powered off, the fencing head node (i.e. the previously passive head node) moves to active mode.

Improving The Decision To Initiate A Failover With A Quorum Process

While the preceding approach guarantees one active head, a problem remains.

In situations where the passive head node loses its connectivity to the active head node, but the active head node is communicating without a problem to the entire cluster, there is no reason to initiate a failover. It can even result in undesirable situations where the cluster is rendered unusable if, for example, a passive head node decides to power down an active head node just because the passive head node is unable to communicate with any of the outside world (except for the PDU feeding the active head node).

One technique used by DirectMon to reduce the chances of a passive head node powering off an active head node unnecessarily is to have the passive head node carry out a quorum procedure. All nodes in the cluster are asked by the passive node to confirm that they also cannot communicate with the active head node. If more than half of the total number of nodes confirm that they are also unable to communicate with the active head node, then the passive head node initiates the STONITH procedure and moves to active mode.

15.1.7 Automatic Vs Manual Failover

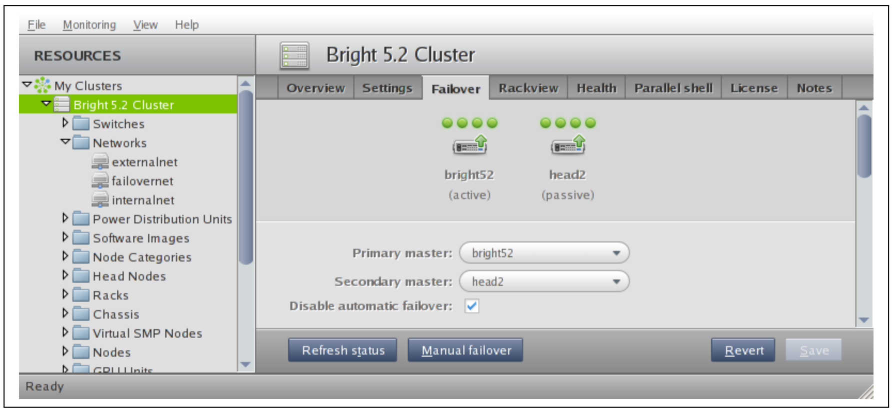
Administrators have a choice between creating an HA setup with automatic or manual failover. In case of automatic failover, an active head node is powered off when it is no longer responding at all, and a failover sequence is initiated automatically.

In case of manual failover, the administrator is responsible for initiating the failover when the active head node is no longer responding. No automatic power off is done, so the administrator is asked to confirm that the previously active node is powered off.

For automatic failover to be possible, power control must be defined for both head nodes. If power control is defined for the head nodes, automatic failover is attempted by default. However, it is possible to disable automatic failover. In `cmsh` this is done by setting the `disableautomaticfailover` property:

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% partition failover base
[ddnmon61->partition[base]->failover]% set disableautomaticfailover yes
[ddnmon61->partition*[base*]->failover*]% commit
```

With `cmgui` it is done by selecting the cluster resource, then selecting the `Failover` tab. Within the tab, the `Disable automatic failover` checkbox is ticked, and the change saved with a click on the `Save` button (figure 15.2).



Primary master: bright52
Secondary master: head2
Disable automatic failover: ✓

Figure 15.2: `cmgui` High Availability: Disable Automatic Failover

If no power control has been defined, or if automatic failover has been disabled, or if the power control mechanism is not working (for example due to inappropriate, broken or missing electronics or hardware), then a failover sequence must always be initiated manually by the administrator.

In addition, if automatic failover is enabled, but the active head is still slightly responsive (the so-called *mostly dead* state, described in section 15.4.2), then the failover sequence must also be initiated manually by the administrator.

15.1.8 HA And Cloud Nodes

As far as the administrator is concerned, HA setup remains the same whether a Cluster Extension is configured or not, and whether a Cluster-On-Demand is configured or not. Behind the scenes, on failover, any networks associated with the cloud requirements are taken care of by DirectMon.

15.2 HA Setup Procedure Using `cmha-setup`

After installation (Chapter 2) and license activation (section 4.1.3) an administrator may wish to add a new head node, and convert DirectMon from managing an existing single-headed cluster to managing an HA cluster.

Is An HA-Enabled License Required?

To convert a single-headed cluster to an HA cluster, the existing cluster license should first be checked to see if it allows HA. The `verify-license` command run with the `info` option can reveal this in the MAC address field:

Example

```
verify-license /cm/local/apps/cmd/etc/cert.{pem,key} info | grep ^MAC
```

HA-enabled clusters display two MAC addresses in the output. Single-headed clusters show only one.

If an HA license is not present, it should be obtained from the DirectMon reseller, and then be activated and installed (section 4.1.3).

Existing User Certificates Become Invalid

Installing the new license means that any existing user certificates will lose their validity (page 89) on `cmgui` session logout. This means:

- If LDAP is managed by DirectMon, then on logout, new user certificates are generated, and a new `cmgui` login session picks up the new certificates automatically.
- For LDAPs other than that of DirectMon, the user certificates need to be regenerated.

It is therefore generally good practice to have an HA-enabled license in place before creating user certificates and profiles if there is an intention of moving from a single-headed to an HA-enabled cluster later on.

The `cmha-setup` Utility For Configuring HA

The `cmha-setup` utility is a special tool that guides the administrator in building an HA setup from a single head cluster. It interacts with the cluster management environment by using `cmsh` to create an HA setup. Although it is in theory also possible to create an HA setup manually, using either `cmgui` or `cmsh` along with additional steps, this is not supported, and should not be attempted as it is error-prone.

A basic HA setup is created in three stages:

1. **Preparation** (section 15.2.1): the configuration parameters are set for the shared interface and for the secondary head node that is about to be installed.
2. **Cloning** (section 15.2.2): the secondary head node is installed by cloning it from the primary head node.
3. **Shared Storage Setup** (section 15.2.3): the method for shared storage is chosen and set up.

An optional extra stage is:

4. **Automated Failover Setup** (section 15.2.4): Power control to allow automated failover is set up.

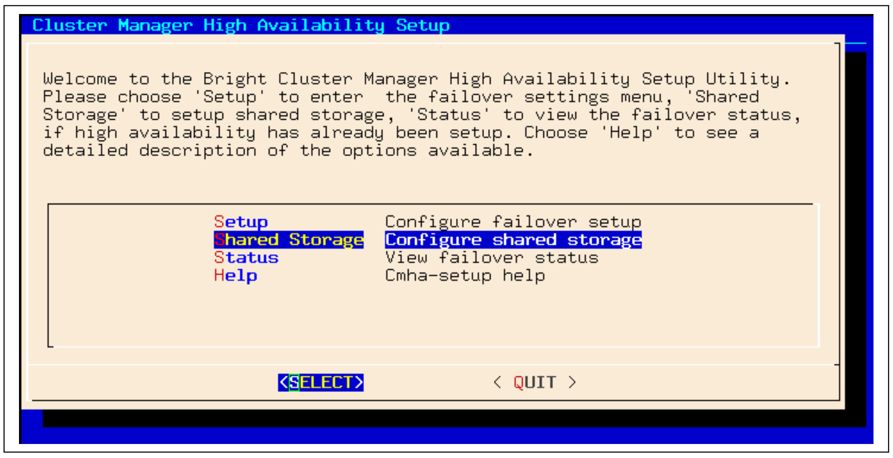
15.2.1 Preparation

The following steps prepare the primary head node for the cloning of the secondary. The preparation is done only on the primary, so that the presence of the secondary is not actually needed during this stage.

0. It is recommended that all nodes except for the primary head node are powered off, in order to simplify matters. The nodes should in any case be power cycled or powered back on after the basic HA setup stages (sections 15.2.1-15.2.3, and possibly section 15.2.4) are complete.

1. To start the HA setup, the `cmha-setup` command is run from a `root` shell on the primary head node.

2. `Setup` is selected from the main menu (figure 15.3).



```
Cluster Manager High Availability Setup

Welcome to the Bright Cluster Manager High Availability Setup Utility.
Please choose 'Setup' to enter  the failover settings menu, 'Shared
Storage' to setup shared storage, 'Status' to view the failover status,
if high availability has already been setup. Choose 'Help' to see a
detailed description of the options available.

              Setup          Configure failover setup
              Shared Storage Configure shared storage
              Status         View failover status
              Help           Cmha-setup help

              <SELECT>               < QUIT >
```

Figure 15.3: `cmha-setup` Main menu

3. `Configure` is selected from the `Setup` menu.

4. A license check is done. Only if successful does the setup proceed further. If the cluster has no HA-enabled license, a new HA-enabled license must first be obtained from the DirectMon reseller, and activated (section 4.1.3).

5. The virtual shared internal alias interface name and virtual shared internal IP alias address are set.

6. The virtual shared external alias interface name and virtual shared external IP alias address are set. To use DHCP assignments on external interfaces, 0.0.0.0 can be used.

7. The hostname of the passive is set.

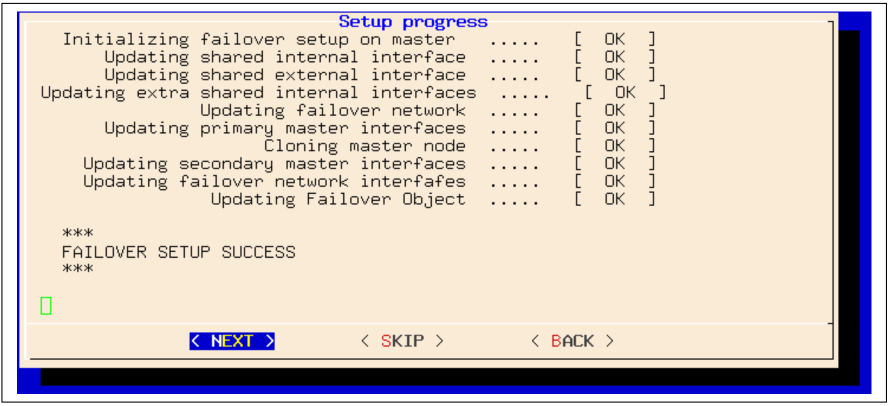
8. Failover network parameters are set. The failover network physical interface should exist, but the interface need not be up. The network name, its base address, its netmask, and domain name are set. This is the network used for optional heartbeat monitoring.

9. Failover network interfaces have their name and IP address set for the active and passive nodes.

10. The primary head node may have other network interfaces (e.g. InfiniBand interfaces, a BMC interface, alias interface on the BMC network). These interfaces are also created on the secondary head node, but the IP address of the interfaces still need to be configured. For each such interface, when prompted, a unique IP address for the secondary head node is configured.

11. The network interfaces of the passive head node are reviewed and can be adjusted as required. DHCP assignments on external interfaces can be set by using an IP address of 0.0.0.0.

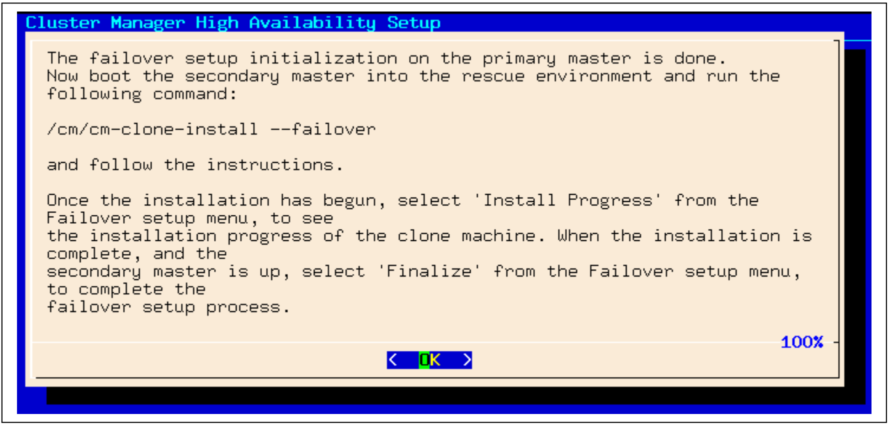
12. A summary screen displays the planned failover configuration. If alterations need to be made, they can be done via the next step.
13. The administrator is prompted to set the planned failover configuration. If it is not set, the main menu of `cmha-setup` is redisplayed.
14. If the option to set the planned failover configuration is chosen, then a password for the MySQL root user is requested. The procedure continues further after the password is entered.
15. Setup progress for the planned configuration is displayed (figure 15.4).



```
                              Setup progress
  Initializing failover setup on master   .....   [  OK  ]
      Updating shared internal interface  .....   [  OK  ]
      Updating shared external interface  .....   [  OK  ]
Updating extra shared internal interfaces  .....   [  OK  ]
               Updating failover network  .....   [  OK  ]
      Updating primary master interfaces  .....   [  OK  ]
                     Cloning master node  .....   [  OK  ]
    Updating secondary master interfaces  .....   [  OK  ]
    Updating failover network interfafes  .....   [  OK  ]
                Updating Failover Object  .....   [  OK  ]

  ***
  FAILOVER SETUP SUCCESS
  ***
```

Figure 15.4: `cmha-setup` Setup Progress For Planned Configuration

16. Instructions on what to run on the secondary to clone it from the primary are displayed (figure 15.5).



```
Cluster Manager High Availability Setup
 The failover setup initialization on the primary master is done.
 Now boot the secondary master into the rescue environment and run the
 following command:

 /cm/cm-clone-install --failover

 and follow the instructions.

 Once the installation has begun, select 'Install Progress' from the
 Failover setup menu, to see
 the installation progress of the clone machine. When the installation is
 complete, and the
 secondary master is up, select 'Finalize' from the Failover setup menu,
 to complete the
 failover setup process.
```

Figure 15.5: `cmha-setup` Instructions To Run On Secondary For Cloning

15.2.2 Cloning

After the preparation has been done by configuring parameters as outlined in section 15.2.1, the cloning of the head nodes is carried out. In the cloning instructions that follow, the active node refers to the primary node and the passive node refers to the secondary node. However this correlation is only true for when an HA setup is created for the first time, and it is not necessarily true if head nodes are replaced later on by cloning.

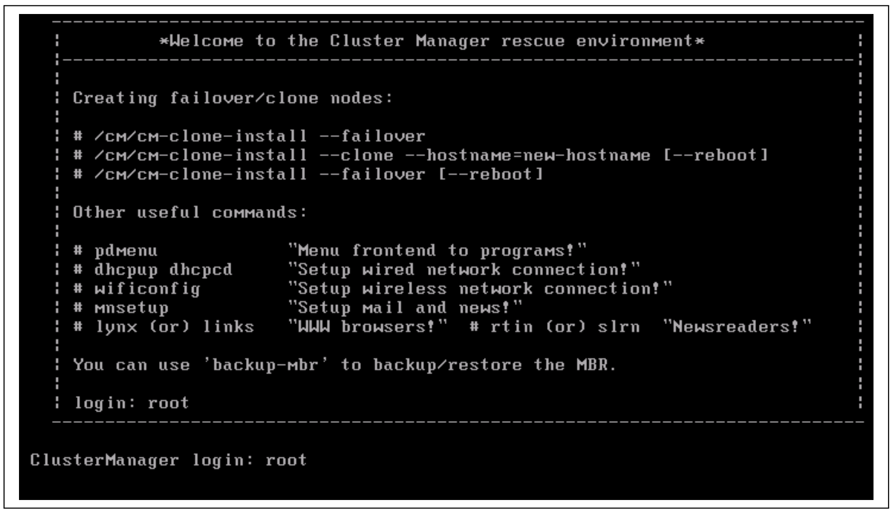
These cloning instructions may also be repeated later on if a passive head node ever needs to be replaced, for example, if the hardware is defective. In that case the active head node can be either the primary or secondary.

The process described PXE boots the passive from the active, thereby loading a special rescue image from the active that allows cloning from the active to the passive to take place.

1. The passive head node is PXE booted off the internal cluster network, from the active head node. It is highly recommended that the active and passive head nodes have identical hardware configurations. Typically, the BIOS of both head nodes is configured so that a hard disk boot is attempted first, and a PXE boot is attempted after a hard disk boot failure, leading to the `Cluster Manager PXE Environment` menu of options. This menu has a 5s time-out.

2. In the `Cluster Manager PXE Environment` menu, before the 5s time-out, "`Start Rescue Environment`" is selected to boot the node into a Linux ramdisk environment.

3. Once the rescue environment has finished booting, a login as root is done. No password is required (figure 15.6).

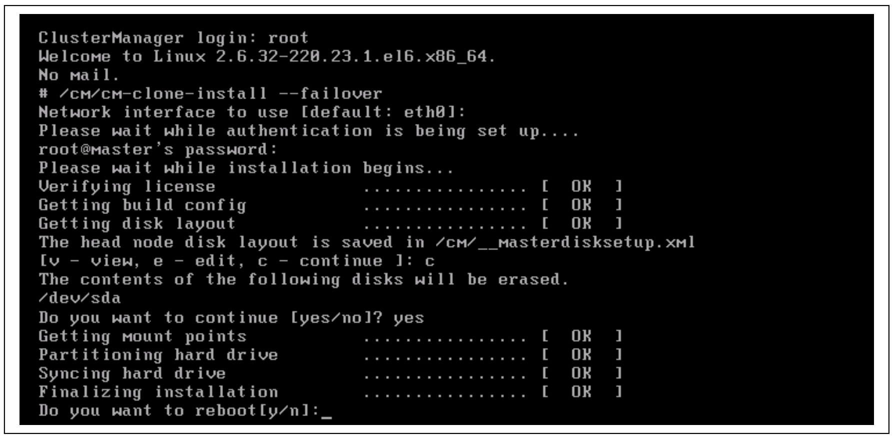


```
 -------------------------------------------------------------------------
 |         *Welcome to the Cluster Manager rescue environment*           |
 |-----------------------------------------------------------------------|
 |                                                                       |
 | Creating failover/clone nodes:                                        |
 |                                                                       |
 | # /cm/cm-clone-install --failover                                     |
 | # /cm/cm-clone-install --clone --hostname=new-hostname [--reboot]     |
 | # /cm/cm-clone-install --failover [--reboot]                          |
 |                                                                       |
 | Other useful commands:                                                |
 |                                                                       |
 | # pdmenu              "Menu frontend to programs!"                    |
 | # dhcpup dhcpcd       "Setup wired network connection!"               |
 | # wificonfig          "Setup wireless network connection!"            |
 | # mnsetup             "Setup mail and news!"                          |
 | # lynx (or) links     "WWW browsers!"  # rtin (or) slrn  "Newsreaders!"|
 |                                                                       |
 | You can use 'backup-mbr' to backup/restore the MBR.                   |
 |                                                                       |
 | login: root                                                           |
 -------------------------------------------------------------------------

ClusterManager login: root
```

Figure 15.6: Login Screen After Booting Passive Into Rescue Mode From Active

4. The following command is executed (figure 15.7):
   `/cm/cm-clone-install --failover`

5. When prompted to enter a network interface to use, the interface that was used to boot from the internal cluster network (e.g. `eth0`, `eth1`, ...) is entered. There is often uncertainty about what interface name corresponds to what physical port. This can be resolved by switching to another console and using "`ethtool -p <interface>`", which makes the NIC corresponding to the interface blink.



```
ClusterManager login: root
Welcome to Linux 2.6.32-220.23.1.el6.x86_64.
No mail.
# /cm/cm-clone-install --failover
Network interface to use [default: eth0]:
Please wait while authentication is being set up....
root@master's password:
Please wait while installation begins...
Verifying license             ................ [  OK  ]
Getting build config          ................ [  OK  ]
Getting disk layout           ................ [  OK  ]
The head node disk layout is saved in /cm/__masterdisksetup.xml
[v - view, e - edit, c - continue ]: c
The contents of the following disks will be erased.
/dev/sda
Do you want to continue [yes/no]? yes
Getting mount points          ................ [  OK  ]
Partitioning hard drive       ................ [  OK  ]
Syncing hard drive            ................ [  OK  ]
Finalizing installation       ................ [  OK  ]
Do you want to reboot[y/n]:_
```

Figure 15.7: Cloning The Passive From The Active Via A Rescue Mode Session

6. If the provided network interface is correct, a `root@master's password` prompt appears. The administrator should enter the root password.

7. An opportunity to view or edit the master disk layout is offered.

8. A confirmation that the contents of the specified disk are to be erased is asked for.

9. The cloning takes place. The "syncing" stage usually takes the most time. Cloning progress can also be viewed on the active by selecting the "`Install Progress`" option from the `Setup` menu. When viewing progress using this option, the display is automatically updated as changes occur.

10. After the cloning process has finished, a prompt at the console of the passive asks if a reboot is to be carried out. A "`y`" is typed in response to this. The passive node should be set to reboot off its hard drive. This may require an interrupt during reboot, to enter a change in the BIOS setting, if for example, the passive node is set to network boot first.

11. Continuing on now on the active head node, `Finalize` is selected from the `Setup` menu of `cmha-setup`.

12. The MySQL root password is requested. After entering the MySQL password, the progress of the `Finalize` procedure is displayed, and the cloning procedure continues.

13. The cloning procedure of `cmha-setup` pauses to offer the option to reboot the passive. The administrator should accept the reboot option. After reboot, the cloning procedure is complete. The administrator can then go to the `main` menu and `quit` from there or go on to configure "`Shared Storage`" (section 15.2.3) from there.

A check can already be done at this stage on the failover status of the head nodes with the `cmha` command, run from either head node:

Example

```
[root@ddnmon61 ~]# cmha status
Node Status: running in active master mode

Failover status:
ddnmon61* -> master2
  backupping    [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
master2 -> ddnmon61*
  backupping    [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
```

Here, the `mysql`, `ping` and `status` states indicate that HA setup completed successfully. The `backupping` (backup ping) state uses the dedicated failover network route for its checks, and starts working as soon as the passive head node has been rebooted.

15.2.3 Shared Storage Setup

After cloning the head node (section 15.2.2), the last basic stage of creating an HA setup is setting up shared storage. The available shared storage forms are NAS, DAS, and DRBD.

NAS

1. In the `cmha-setup` main menu, the "`Shared Storage`" option is selected.
2. `NAS` is selected.
3. The parts of the head node filesystem that are to be copied to the NAS filesystems are selected. By default, these are `/home` and `/cm/shared` as suggested in section 15.1.5. The point in the filesystem where the copying is done is the future mount path to where the NAS will share the shared filesystem.

   An already-configured export that is not shared is disabled in `/etc/exports` by `cmha-setup`. This is done to prevent the creation of stale NFS file handles during a failover. Sharing already-existing exports is therefore recommended. Storage can however be dealt with in a customized manner with mount and unmount scripts (page 541).
4. The NFS hostname is configured. Also, for each head node filesystem that is to be copied to the NAS filesystem, there is an associated path on the NAS filesystem where the share is to be served from. These NFS volume paths are now configured.
5. If the configured NFS filesystems can be correctly mounted from the NAS server, the process of copying the local filesystems onto the NAS server begins.

DAS

A prerequisite to the DAS configuration steps that follow is that the partitions exist on the DAS device that is to be used for shared storage. These should be created manually if required, before running `cmha-setup`.

1. In the `cmha-setup` main menu, the "`Shared Storage`" option is selected.

2. `DAS` is selected.

3. The filesystems that are to be shared over the cluster are selected by the administrator. The filesystems that are shared are typically `/cm/shared` and `/home`, but this is not mandatory.

4. The filesystems that are to be shared are assigned DAS partitions by the administrator. For example, the administrator may specify these as `/dev/sdc1` for `/home` and `/dev/sdd3` for `/cm/shared`.

5. The administrator can choose to create a filesystem on the proposed DAS device.

   - Creating the filesystem on the device means any existing filesystems and files on the device are wiped during creation.
   - Otherwise, the existing filesystem and files on the DAS device remain.

6. A filesystem type is set from a choice of `ext3`, `ext4`, `xfs`.

7. A summary screen is presented showing the proposed changes.

8. After filesystems have been created, the current contents of the shared directories are copied onto the shared filesystems and the shared filesystems are mounted over the old non-shared filesystems.

9. The administrator should check that the partitions are visible from both head nodes using, for example, the `fdisk -l` command on the device. If the partitions on the DAS are created or modified, or appear only after the passive head is running due to a hardware-related reason after the passive head is powered on, then the kernel on the passive head may not have reread the partition table. A power cycle of the head nodes is recommended if the partitions are not seen properly.

DRBD

The DRBD device should be prepared by the administrator before the following setup procedure is carried out:

1. In the `cmha-setup` main menu, the "`Shared Storage`" option is selected.

2. `DRBD` is selected.

3. The parts of the filesystem that should be placed on DRBD filesystems are selected.

4. The hostnames of the primary and secondary head nodes and the physical disk partitions to use on both head nodes are entered.

5. That the contents of the listed partitions can be erased on both head nodes is confirmed. After DRBD based filesystems have been created, the current contents of the shared directories are copied onto

the DRBD based filesystems and the DRBD based filesystems are mounted over the old non-shared filesystems.

6. Once the setup process has completed, "`DRBD Status/Overview`" is selected to verify the status of the DRBD block devices.

15.2.4 Automated Failover And Relevant Testing

For automatic failover to work, the two head nodes must be able to power off their counterpart. This is done by setting up power control (Chapter 5).

Testing If Power Control Is Working

The "`device power status`" command in `cmsh` can be used to verify that power control is functional:

Example

```
[master1]% device power status -n mycluster1,mycluster2
 apc03:21 ............ [    ON    ] mycluster1
 apc04:18 ............ [    ON    ] mycluster2
```

Testing The BMC Interface Is Working

If a BMC (Baseboard Management Controller, section 4.8) such as IPMI or iLO is used for power control, it is possible that a head node is not able to reach its own BMC interface over the network. This is especially true when no dedicated BMC network port is used. In this case, `cmsh -c "device power status"` reports a failure for the active head node. This does not necessarily mean that the head nodes cannot reach the BMC interface of their counterpart. Pinging a BMC interface can be used to verify that the BMC interface of a head node is reachable from its counterpart.

Example

Verifying that the BMC interface of `mycluster2` is reachable from `mycluster1`:

```
[root@mycluster1 ~]# ping -c 1 mycluster2.bmc.cluster
PING mycluster2.bmc.cluster (10.148.255.253) 56(84) bytes of data.
64 bytes from mycluster2.bmc.cluster (10.148.255.253): icmp_seq=1
ttl=64 time=0.033 ms
```

Verifying that the BMC interface of `mycluster1` is reachable from `mycluster2`:

```
[root@mycluster2 ~]# ping -c 1 mycluster1.bmc.cluster
PING mycluster1.bmc.cluster (10.148.255.254) 56(84) bytes of data.
64 bytes from mycluster1.bmc.cluster (10.148.255.254): icmp_seq=1
ttl=64 time=0.028 ms
```

Testing Automated Failover Against A Simulated Crash

A normal (graceful) shutdown of an active head node, does not cause the passive to become active, because HA assumes a graceful failover means there is no intention to trigger a failover. To carry out testing of an HA setup with automated failover, it is therefore useful to simulate a kernel crash on one of the head nodes. The following command crashes a head node instantly:

```
echo c > /proc/sysrq-trigger
```

After the active head node freezes as a result of the crash, the passive head node powers off the machine that has frozen and switches to active mode. A hard crash like this can cause a database replication inconsistency when the crashed head node is brought back up and running again, this time passively, alongside the node that took over. This is normally indicated by a `FAILED` status for the output of `cmha status` for MySQL (section 15.4). Database administration with the `dbreclone` command (section 15.4) may therefore be needed to synchronize the databases on both head nodes to a consistent state. Because `dbreclone` is a resource-intensive utility, it is best used during a period when there are few or no users. It is generally only used by administrators when they are instructed to do so by DDN support.

A passive node can also be made active without a crash of the active-until-then node, by using the "`cmha makeactive`" command on the passive (section 15.4.2). Manually running this is not needed in the case of a head node crash in a cluster where power management has been set up for the head nodes, and the automatic failover setting is not disabled.

15.3 Running `cmha-setup` Without Ncurses, Using An XML Specification

15.3.1 Why Run It Without Ncurses?

The text-based ncurses GUI for `cmha-setup` is normally how administrators should set up a failover configuration.

The express mode of `cmha-setup` allows an administrator to skip the GUI. This is useful, for example, for scripting purposes and speeding deployment. A further convenience is that this mode uses a human-editable XML file to specify the network and storage definitions for failover.

Running `cmha-setup` without the GUI still requires some user intervention, such as entering the root password for MySQL. The intervention required is scriptable with, for example, Expect, and is minimized if relevant options are specified for `cmha-setup` from the `-x` options.

15.3.2 The Syntax Of `cmha-setup` Without Ncurses

The express mode (`-x`) options are displayed when "`cmha-setup -h`" is run. The syntax of the `-x` options is indicated by:

```
cmha-setup [ -x -c <configfile> [-s <type>]
             <-i|-f[-r]> [-p <mysqlrootpassword>] ]
```

The `-x` options are:

- `-c|--config` <*configfile*>: specifies the location of <*configfile*>, which is the failover configuration XML file for `cmha-setup`. The file stores the values to be used for setting up a failover head node. The recommended location is at `/cm/local/apps/cluster-tools/ha/conf/failoverconf.xml`.

- `-i|--initialize`: prepares a failover setup by setting values in the CMDaemon database to the values specified in the configuration file. This corresponds to section 15.2.1. The administrator is prompted for the MySQL root password unless the `-p` option is used. The `-i` option of the script then updates the interfaces in the database, and clones the head node in the CMDaemon database. After this option in the script is done, the administrator normally carries clones the passive node from the active, as described in steps 1 to 10 of section 15.2.2.

- `-f|--finalize`: After the passive node is cloned as described in steps 1 to 10 of section 15.2.2, the finalize option is run on the active node to run the non-GUI finalize procedure. This is the non-GUI version of steps 11 to 13 of section 15.2.2.
  - `-r|--finalizereboot`: makes the passive reboot after the finalize step completes.
- `-s|--sharedstorage` <*type*>: specifies the shared storage <*type*> out of a choice of `nas`, `das` or `drbd`.
- `-p|--pass` <*mysqlrootpassword*>: specifies the MySQL root password. Leaving this out means the administrator is prompted to type in the password during a run of the `cmha-setup` script when using the `-x` options.

There is little attempt at validation with the express mode, and invalid entries can cause the command to hang.

15.3.3 Example `cmha-setup` Run Without Ncurses

Preparation And Initialization:

After shutting down all nodes except for the active head node, a configuration is prepared by the administrator in `/cm/local/apps/cluster-tools/ha/conf/failoverconf.xml`. The administrator then runs `cmha-setup` with the initialization option on the active:

```
[root@ddnmon61 ~]# cd /cm/local/apps/cluster-tools/ha/conf
[root@ddnmon61 conf]# cmha-setup -x -c failoverconf.xml -i
Please enter the mysql root password:
   Initializing failover setup on master   .....    [  OK  ]
       Updating shared internal interface  .....    [  OK  ]
       Updating shared external interface  .....    [  OK  ]
Updating extra shared internal interfaces  .....    [  OK  ]
                Updating failover network  .....    [  OK  ]
       Updating primary master interfaces  .....    [  OK  ]
                      Cloning master node  .....    [  OK  ]
     Updating secondary master interfaces  .....    [  OK  ]
     Updating failover network interfaces  .....    [  OK  ]
                 Updating Failover Object  .....    [  OK  ]
```

The preceding corresponds to the steps in section 15.2.1.

PXE Booting And Cloning The Passive:

The passive head node is then booted up via PXE and cloned as described in steps 1 to 10 of section 15.2.2.

Finalizing On The Active And Rebooting The Passive:
Then, back on the active head node the administrator continues the session there, by running the finalization option with a reboot option:

```
[root@ddnmon61 conf]# cmha-setup -x -c failoverconf.xml -f -r
Please enter the mysql root password:
    Updating secondary master mac address  .....    [  OK  ]
   Initializing failover setup on master2  .....    [  OK  ]
                         Cloning database  .....    [  OK  ]
                    Update DB permissions  .....    [  OK  ]
  Checking for dedicated failover network  .....    [  OK  ]
A reboot has been issued on master2
```

The preceding corresponds to steps 11 to 13 of section 15.2.2.

Adding Storage:
Continuing with the session on the active, setting up a shared storage could be done with:

```
[root@ddnmon61 conf]# cmha-setup -x -c failoverconf.xml -s nas
```

The preceding corresponds to carrying out the NAS procedure of section 15.2.3.

15.4 Managing HA

Once an HA setup has been created, the tools in this section can be used to manage the HA aspects of the cluster.

15.4.1 Changing An Existing Failover Configuration

Changing an existing failover configuration is usually done most simply by running through the HA setup procedure of section 15.2 again, with one exception. The exception is that the existing failover configuration must be removed by using the "`Undo Failover`" menu option between steps 2 and 3 of the procedure described in section 15.2.1.

15.4.2 `cmha` Utility

A major command-line utility for interacting with the HA subsystem is `cmha`. Its usage information is:

```
[root@mycluster1 ~]# cmha
Usage:
  /cm/local/apps/cmd/sbin/cmha status | makeactive | dbreclone <node>
```

The options do the following:

- **status**: query the status of the HA subsystem on the local machine
- **makeactive**: initiate failover manually, making the current machine active
- **dbreclone**: clone the CMDaemon and Slurm database to another head node

The `status` information and `makeactive` functionalities of `cmha` are also accessible via `cmgui`, as described in section 15.4.7. The `dbreclone` option cannot carried out in `cmgui` or `cmsh` because it requires stopping CMDaemon.

The `cmha` options are looked at in greater detail next:

`cmha status`: Querying HA Status

Information on the failover status is displayed thus:

Example

```
[root@mycluster1 ~]# cmha status
Node Status: running in active master mode

Failover status:
mycluster1* -> mycluster2
  backupping    [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
mycluster2 -> mycluster1*
  backupping    [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
```

The * in the output indicates the head node which is currently active. The `status` output shows 4 aspects of the HA subsystem from the perspective of each head nodes:

HA Status	Description
`backupping`	the other head node is reachable via the dedicated failover network. This backup ping uses the failover route instead of the internal net route. It is a SYN ping.
`mysql`	MySQL replication status
`ping`	the other head node is reachable over the primary management network. It is a SYN ping.
`status`	CMDaemon running on the other head node responds to SOAP calls

By default, DirectMon prepares to carry out the failover sequence (the sequence that includes a STONITH) when all three of `ping`, `backupping` and `status` are not `OK` on a head node. One way of initiating failover is thus by causing a system crash (section 15.2.4).

It can typically take about 30s for the `cmha status` command to output its findings in the case of a recently crashed head node.

`cmha makeactive`: Initiate Failover

If automatic failover is enabled (section 15.1.7), then the failover sequence attempts to complete automatically if power management is working properly, and the `cmha status` shows `ping`, `backupping` and `status` as failed.

If automatic failover is disabled, then a manual failover operation must be executed to have a failover operation take place. A manual failover operation can be carried out from `cmgui`, as described in section 15.4.7, or it can be carried out with the "`cmha makeactive`" command:

Example

To initiate a failover manually:

```
[root@mycluster2 ~]# cmha makeactive
Proceeding will initiate a failover sequence which will make this node
(mycluster2) the active master.

Are you sure ? [Y/N]
y
Your session ended because: CMDaemon failover, no longer master
mycluster2 became active master, reconnecting your cmsh ...
```

On successful execution of the command, the former active head node simply continues to run as a passive head node.

The `cmha makeactive` command assumes both head nodes have no problems preventing the execution of the command.

`cmha makeactive` **edge case—the mostly dead active:**

- For a manual failover operation, if the execution of the `cmha makeactive` command has problems, then it can mean that there is a problem with the initially active head node being in a sluggish state. That is, neither fully functioning, nor *all dead*. The active head node is thus in a state that is still powered on, but what can be called *mostly dead*. Mostly dead means slightly alive (at least one of `ping`, `backupping`, and `status` are OK), while all dead means there is only one thing that can sensibly be done to make sure the cluster keeps running—that is, to make the old passive the new active.

  Making an old passive the new active is only safe if the old active is guaranteed to not come back as an active head node. This guarantee is set by a STONITH (page 543) for the old active head node, and results in a former active that is now all dead. STONITH thus guarantees that head nodes are not in conflict about their active and passive states. STONITH can however still fail in achieving a clean shutdown when acting on a mostly dead active head node, which can result in unclean filesystem or database states.

  Thus, the mostly dead active head node may still be in the middle of a transaction, so that shutting it down may cause filesystem or database corruption. Making the passive node also active then in this case carries risks such as mounting filesystems accidentally on both head nodes, or carrying out database transactions on both nodes. This can also result in filesystem and database corruption.

  It is therefore left to the administrator to examine the situation for corruption risk. The decision is either to power off a mostly dead head node, i.e. STONITH to make sure it is all dead, or whether to wait for a recovery to take place. When carrying out a STONITH on the mostly dead active head node, the administrator must power it off *before* the passive becomes active for a manual failover to take place with minimal errors. The `cmha dbreclone` option may still be needed to restore a corrupted database after such a power off, after bringing the system back up.

- For an automated failover configuration, powering off the mostly dead active head node is not carried out automatically due to the risk of filesystem and database corruption. A mostly dead active node with automatic failover configuration therefore stays mostly dead either until it recovers, or until the administrator decides to do a STONITH manually to ensure it is all dead. Here, too, the `cmha dbreclone` option may still be needed to restore a corrupted database after such a power off, after bringing the system back up.

`cmha dbreclone`**: Cloning The CMDaemon Database**

The `dbreclone` option of `cmha` clones the CMDaemon state database from the head node on which `cmha` runs to the head node specified after the option. It is normally run in order to clone the database from the active head node to the passive—running it from the passive to the active can cause a loss of database entries. Running the `dbreclone` option can be used to retrieve the MySQL CMDaemon state database tables, if they are, for example, unsalvageably corrupted on the destination node, and the source node has a known good database state. Because it is resource intensive, it is best run when there are few or no users. It is typically only used by administrators after being instructed to do so by DDN support.

Example

```
[root@ddnmon61 ~]# cmha status
Node Status: running in active master mode

Failover status:
ddnmon61* -> head2
  backupping   [  OK  ]
  mysql        [  OK  ]
  ping         [  OK  ]
  status       [  OK  ]
head2 -> ddnmon61*
  backupping   [  OK  ]
  mysql        [FAILED] (11)
  ping         [  OK  ]
  status       [  OK  ]
[root@ddnmon61 ~]# cmha dbreclone head2
Proceeding will cause the contents of the cmdaemon state database on he\
ad2 to be resynchronized from this node (i.e. ddnmon61 -> head2)

Are you sure ? [Y/N]
Y
Waiting for CMDaemon (3113) to terminate...
[  OK  ]
Waiting for CMDaemon (7967) to terminate...
                                                            [  OK  ]
cmdaemon.dump.8853.sql                        100%  253KB 252.9KB/s   00:00
slurmacctdb.dump.8853.sql                     100%   11KB  10.7KB/s   00:00
Waiting for CMDaemon to start...                            [  OK  ]
Waiting for CMDaemon to start...[  OK  ]
[root@ddnmon61 ~]# cmha status
Node Status: running in active master mode

Failover status:
```

```
ddnmon61* -> head2
  backupping    [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
head2 -> ddnmon61*
  backupping    [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
```

15.4.3 States

The state a head node is in can be determined in three different ways:

1 By looking at the message being displayed at login time.

Example

```
----------------------------------------------------------------------

Node Status: running in active master mode

----------------------------------------------------------------------
```

2 By executing `cmha status`.

Example

```
[root@mycluster ~]# cmha status
Node Status: running in active master mode
...
```

3 By examining `/var/spool/cmdaemon/state`.

There are a number of possible states that a head node can be in:

State	Description
`INIT`	Head node is initializing
`FENCING`	Head node is trying to determine whether it should try to become active
`ACTIVE`	Head node is in active mode
`PASSIVE`	Head node is in passive mode
`BECOMEACTIVE`	Head node is in the process of becoming active
`BECOMEPASSIVE`	Head node is in the process of becoming passive
`UNABLETOBECOMEACTIVE`	Head node tried to become active but failed
`ERROR`	Head node is in error state due to unknown problem

Especially when developing custom mount and unmount scripts, it is quite possible for a head node to go into the `UNABLETOBECOMEACTIVE` state. This generally means that the mount and/or unmount script are not working properly or are returning incorrect exit codes. To debug these situations, it is helpful to examine the output in `/var/log/cmdaemon`. The "`cmha makeactive`" shell command can be used to instruct a head node to become active again.

15.4.4 Failover Action Decisions

A table summarizing the scenarios that decide when a passive head should take over is helpful:

Event on active	Reaction on passive	Reason
Reboot	Nothing	Event is usually an administrator action. To make the passive turn active, an administrator would run "`cmha makeactive`" on it.
Shutdown	Nothing	As above.
Unusably sluggish or freezing system but state pingable with SYN packets	Nothing	1. Active may still unfreeze. 2. Shared filesystems may still be in use by the active. Concurrent use by the passive taking over therefore risks corruption. 3. Mostly dead head can be powered off by administrator after examining situation (section 15.4.2).
Become passive in response to "`cmha makeactive`"	Become active when former active becomes passive	As ordered by administrator

...continues

...continued

Event on active	Reaction on passive	Reason
run on passive		
Active dies	Quorum called, may lead to passive becoming new active	Confirms if active head is dead according to other nodes too. If so, then a "power off" command is sent to it. If the command is successful, the passive head becomes the new active head.

15.4.5 Keeping Head Nodes In Sync

What Should Be Kept In Sync?

If filesystem changes are made on an active head node without using CMDaemon (`cmsh` or `cmgui`), and if the changes are outside the shared filesystem, then these changes should normally also be made on the passive head node. For example:

- RPM installations/updates (section 10.2)
- Applications installed locally
- Files (such as drivers or values) placed in the `/cm/node-installer/` directory and referred to by `initialize` (section 6.4.5) and `finalize` scripts (section 6.4.11)
- Any other configuration file changes outside of the shared filesystems

If the cluster is being built on bare metal, a sensible way to minimize the amount of work to be done is to install a single head cluster first. All packages and applications should then be placed, updated and configured on that single head node until it is in a satisfactory state. Only then should HA be set up as described in section 15.2, where the cloning of data from the initial head node to the secondary is described. The result is then that the secondary node gets a well-prepared system with the effort to prepare it having only been carried out once.

Avoiding Encounters With The Old Filesystems

It should be noted that when the shared storage setup is made, the contents of the shared directories (at that time) are copied over from the local filesystem to the newly created shared filesystems. The shared filesystems are then mounted on the mountpoints on the active head node, effectively hiding the local contents.

Since the shared filesystems are only mounted on the active machine, the old filesystem contents remain visible when a head node is operating in passive mode. Logging into the passive head node may thus confuse users and is therefore best avoided.

Updating Services On The Head Nodes And Associated Syncing

The services running on the head nodes described in section 15.1.3 should also have their packages updated on both head nodes.

For the services that run simultaneously on the head nodes, such as CMDaemon, DHCP, LDAP, MySQL, NTP and DNS, their packages should be updated on both head nodes at about the same time. A suggested procedure is to stop the service on both nodes around the same time, update the service and ensure that it is restarted.

The provisioning node service is part of the CMDaemon package. The service updates images from the active head node to all provisioning nodes, including the passive head node, if the administrator runs the command to update provisioners. How to update provisioners is described in section 15.1.3.

For services that migrate across head nodes during failover, such as NFS and Workload Management, it is recommended (but not mandated) to carry out this procedure: the package on the passive node (called the secondary for the sake of this example) is updated to check for any broken package behavior. The secondary is then made active with `cmha makeactive` (section 15.4.2), which automatically migrates users cleanly off from being serviced by the active to the secondary. The package is then updated on the primary. If desired, the primary can then be made active again. The reason for recommending this procedure for services that migrate is that, in case the update has issues, the situation can be inspected somewhat better with this procedure.

15.4.6 High Availability Parameters

There are several HA-related parameters that can be tuned. Accessing these via `cmgui` is described in section 15.4.7. In `cmsh` the settings can be accessed in the `failover` submode of the `base` partition.

Example

```
[mycluster1]% partition failover base
[mycluster1->partition[base]->failover]% show
Parameter                      Value
------------------------------ ----------------------------
Dead time                      10
Disable automatic failover     no
Failover network               failovernet
Init dead                      30
Keep alive                     1
Mount script
Postfailover script
Prefailover script
Quorum time                    60
Revision
Secondary master
Unmount script
Warn time                      5
```

`Dead time`

When a passive head node determines that the active head node is not responding to any of the periodic checks for a period longer than the `Dead time` seconds, the active head node is considered dead and a quorum procedure starts. Depending on the outcome of the quorum, a failover sequence may be initiated.

`Disable automatic failover`
Setting this to yes disables automated failover. Section 15.1.7 covers this further.

`Failover network`
The `Failover network` setting determines which network is used as a dedicated network for the `backupping` (backup ping) heartbeat check. The heartbeat connection is normally a direct cable from a NIC on one head node to a NIC on the other head node. The network can be selected via tab-completion suggestions. By default, without a dedicated failover network, the possibilities are nothing, `externalnet` and `internalnet`.

`Init dead`
When head nodes are booted simultaneously, the standard `Dead time` might be too strict if one head node requires a bit more time for booting than the other. For this reason, when a head node boots (or more exactly, when the cluster management daemon is starting), a time of `Init dead` seconds is used rather than the `Dead time` to determine whether the other node is alive.

`Keep alive`
The `Keep alive` value is the time interval, in seconds, over which the passive head node carries out a check that the active head node is still up. If a dedicated failover network is used, 3 separate heartbeat checks are carried out to determine if a head node is reachable.

`Mount script`
The script pointed to by the `Mount script` setting is responsible for bringing up and mounting the shared filesystems.

`Postfailover script`
The script pointed to by the `Postfailover script` setting is run by `cmdaemon` on both head nodes. The script first runs on the head that is now passive, then on the head that is now active. It runs as soon as the former passive has become active. It is typically used by scripts mounting an NFS shared storage so that no more than one head node exports a filesystem to NFS clients at a time.

`Prefailover script`
The script pointed to by the `Prefailover script` setting is run by `cmdaemon` on both head nodes. The script first runs on the (still) active head, then on the (still) passive head. It runs as soon as the decision for the passive to become active has been made, but before the changes are implemented. It is typically used by scripts unmounting an NFS shared storage so that no more than one head node exports a filesystem to NFS clients at a time. When unmounting shared storage, it is very important to ensure that a non-zero exit code is returned if unmounting has problems, or the storage may become mounted twice during the `Postfailover script` stage, resulting in data corruption.

`Quorum time`
When a node is asked what head nodes it is able to reach over the network, the node has `Quorum time` seconds to respond. If a node does not respond to a call for quorum within that time, it is no longer considered for the results of the quorum check.

`Secondary master`
The `Secondary master` setting is used to define the secondary head node to the cluster.

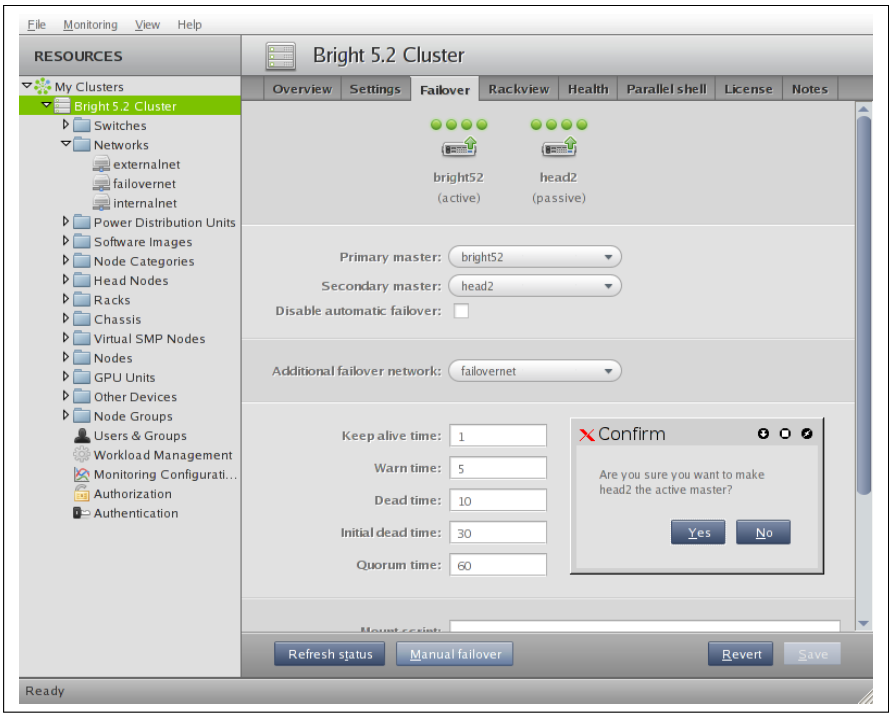
`Unmount script`
The script pointed to by the `Unmount script` setting is responsible for bringing down and unmounting the shared filesystems.

`Warn time`
When a passive head node determines that the active head node is not responding to any of the periodic checks for a period longer than `Warn time` seconds, a warning is logged that the active head node might become unreachable soon.

15.4.7 Handling And Viewing Failover Via `cmgui`

Accessing `cmha` Functionality Via `cmgui`

The equivalent functions of `cmha` (section 15.4.2) are available under `cmgui` by selecting a cluster from the resource tree, and then choosing the `Failover` tab (figure 15.8).



bright52 (active)
head2 (passive)

Primary master: bright52
Secondary master: head2
Disable automatic failover: ☐

Additional failover network: failovernet

Keep alive time: 1
Warn time: 5
Dead time: 10
Initial dead time: 30
Quorum time: 60

Mount script:

Confirm
Are you sure you want to make head2 the active master?
Yes No

Figure 15.8: Accessing HA Cluster Parameters And Functions Via `cmgui`

The states of the head nodes are indicated in the first section of the tabbed pane, where the machines are shown along with their modes and states, and with LED lights. Hovering the mouse cursor over the LED lights causes hovertext to appear, indicating which check is associated with which light.

Manual failover can be initiated from `cmgui` by clicking on the `Manual Failover` button (figure 15.8).

Accessing `cmsh` HA Parameters (`partition failover base`) Via `cmgui`

The `cmgui` equivalents of the `cmsh` HA parameters in section 15.4.6 are accessed from the same `cmgui` tab as described earlier in this section (15.4.7).

15.4.8 Re-cloning A Head Node

Some time after an HA setup has gone into production, it may become necessary to re-install one of the head nodes, for example if one of the head nodes were replaced due to hardware failure.

To re-clone a head node from an existing active head node, `cmha-setup` is entered, `Setup` is selected, and then “`Clone Install`” is selected. The displayed instructions are then followed (i.e. the instructions in section 15.2.2 are repeated).

If the MAC address of one of the head nodes has changed, it is typically necessary to request that the product key is unlocked, so that a new license can be obtained (section 4.1.3).

Exclude Lists And Cloning

Some files are normally excluded from being copied across from the head node to the clone, because syncing them is not appropriate.

The following exclude files are read from inside the directory `/cm/` on the clone node when the `cm-clone-install` command is run (step 4 in section 15.2.2).

- `excludelistnormal`: used to exclude files to help generate a clone of the other head node. It is read when running the `cm-clone-install` command without the `--failover` option.
- `excludelistfailover`: used to exclude files to help generate a passive head node from an active head node. It is read when running the `cm-clone-install --failover` command.

In a default cluster, there is no need to alter these exclude files. However some custom head node configurations may require appending a path to the list.

Exclude Lists In Perspective

The exclude lists `excludelistfailover` and `excludelistnormal` described in the preceding paragraphs should not be confused with the exclude lists of section 6.6.1. The exclude lists of section 6.6.1:

- `excludelistupdate`
- `excludelistfullinstall`
- `excludelistsyncinstall`
- `excludelistgrabnew`
- `excludelistgrab`

are `cmgui` or `cmsh` options, and are maintained by CMDaemon. On the other hand, the exclude lists introduced in this section (15.4.8):

- `excludelistfailover`
- `excludelistnormal`

are not `cmgui` or `cmsh` options, and are not maintained by CMDaemon, but are made use of when running the `cm-clone-install` command.

A

Generated Files

This appendix contains lists of all system configuration files which are generated automatically by the node-installer or CMDaemon. These are files generated on the head nodes (section A.1), in the software images (section A.2), and on the regular nodes (section A.3). These files should not be confused with configuration files that are merely installed (section A.4).

Section 3.6.4 describes how system configuration files on all nodes are written out using the Cluster Management Daemon. The Cluster Management Daemon is introduced in section 3.6.4 and its configuration directives are listed in Appendix C.

All of these configuration files may be listed as `Frozen Files` in the Cluster Management Daemon configuration file to prevent them from being generated automatically. The files can be frozen for the head node by setting the directive at `/cm/local/apps/cmd/etc/cmd.conf`. They can also be frozen on the regular nodes by setting the directive in the software image, by default at `/cm/images/default-image/cm/local/apps/cmd/etc/cmd.conf`.

A.1 Files Generated Automatically On Head Nodes

Files generated or modified automatically on the head node by CMDaemon:

File	Method	Comment
`/etc/aliases`	Section	
`/etc/dhclient.conf`	Entire file	Red Hat only
`/etc/dhcpd.conf`	Entire file	
`/etc/dhcpd.internalnet.conf`	Entire file	

...continues

...continued

File	Method	Comment
`/etc/exports`	Section	
`/etc/fstab`	Section	
`/etc/hosts`	Section	
`/etc/hosts.equiv`	Section	
`/etc/localtime`	Entire file	Copied from zoneinfo
`/etc/named.conf`	Entire file	For zone additions use `/etc/named.conf.include`[1]. For options additions, use `/etc/named.conf.global.options.include`
`/etc/ntp.conf`	Section	
`/etc/ntp/step-tickers`	Section	Red Hat only
`/etc/postfix/canonical`	Section	
`/etc/postfix/main.cf`	Section	
`/etc/postfix/generic`	Section	
`/etc/resolv.conf`	Section	
`/etc/shorewall/interfaces`	Section	
`/etc/shorewall/masq`	Section	
`/etc/shorewall/netmap`	Section	
`/etc/shorewall/policy`	Section	
`/etc/shorewall/zones`	Section	
`/etc/slurm/gres.conf`	Section	
`/etc/slurm/slurm.conf`	Section	
`/etc/slurm/slurmdbd.conf`	Section	
`/etc/slurm/topology.conf`	Section	
`/etc/sysconfig/bmccfg`	Entire file	BMC configuration
`/etc/sysconfig/clock`	Section	
`/etc/sysconfig/dhcpd`	Entire file	
`/etc/sysconfig/network`	Entire file	Red Hat only
`/etc/sysconfig/network/routes`	Section	SUSE only

. . . continues

...continued

File	Method	Comment
`/etc/sysconfig/network/ifcfg-*`	Section	SUSE only
`/etc/sysconfig/network/dhcp`	Section	SUSE only
`/etc/sysconfig/network-scripts/ifcfg-*`	Section	Red Hat only
`/etc/HOSTNAME`	Entire file	SUSE only
`/tftpboot/mtu.conf`	Entire file	DDN configuration
`/var/named/*.zone`[1]	Entire file	Red Hat only. For custom additions use `/var/named/*.zone.include`
`/var/lib/named/*.zone`[1]	Entire file	SUSE only. For custom additions use `/var/lib/named/*.zone.include`

[1] User-added zone files ending in `*.zone` that are placed for a corresponding zone statement in the include file `/etc/named.conf.include` are wiped by CMDaemon activity. Another pattern, eg: `*.myzone`, must therefore be used instead.

A.2 Files Generated Automatically In Software Images:

Files generated automatically in software images

File	Generated By	Method	Comment
/boot/vmlinuz	CMDaemon	Symlink	
/boot/initrd	CMDaemon	Symlink	
/boot/initrd-*	CMDaemon	Entire file	
/etc/aliases	CMDaemon	Section	
/etc/hosts	CMDaemon	Section	
/etc/init/serial.conf	CMDaemon	Section	Red Hat only
/etc/inittab	CMDaemon	Section	SUSE only
/etc/localtime	CMDaemon	Entire file	
/etc/mkinitrd_cm.conf	CMDaemon	Section	Red Hat only
/etc/modprobe.d/ddn-cmdaemon.conf	CMDaemon	Section	
/etc/postfix/main.cf	CMDaemon	Section	
/etc/securetty	CMDaemon	Section	
/etc/sysconfig/clock	CMDaemon	Section	
/etc/sysconfig/init	CMDaemon	Section	Red Hat only
/etc/sysconfig/network/dhcp	CMDaemon	Section	SUSE only

A.3 Files Generated Automatically On Regular Nodes

Files generated automatically on regular nodes

File	Generated By	Method	Comment
`/etc/aliases`	CMDaemon	Section	
`/etc/exports`	CMDaemon	Section	
`/etc/fstab`	Node-installer	Section	
`/etc/hosts`	Node-installer	Section	
`/etc/init/serial.conf`	CMDaemon	Section	Red Hat only
`/etc/inittab`	CMDaemon	Section	SUSE only
`/etc/mkinitrd_cm.conf`	CMDaemon	Section	Red Hat only
`/etc/modprobe.d/bright-cmdaemon.conf`	CMDaemon	Section	
`/etc/pam.d/sshd`	CMDaemon	Section	
`/etc/ntp.conf`	Node-installer	Entire file	
`/etc/ntp/step-tickers`	Node-installer	Entire file	Red Hat only
`/etc/postfix/main.cf`	Node-installer and CMDaemon	Section	
`/etc/resolv.conf`	Node-installer	Entire file	
`/etc/securetty`	CMDaemon	Section	
`/etc/slurm/gres.conf`	CMDaemon	Section	
`/etc/slurm/slurm.conf`	CMDaemon	Section	
`/etc/sysconfig/clock`	CMDaemon	Section	
`/etc/sysconfig/init`	CMDaemon	Section	Red Hat Only
`/etc/sysconfig/network`	Node-installer	Entire file	
`/etc/sysconfig/network/dhcp`	CMDaemon	Section	SUSE only
`/etc/sysconfig/network/ifcfg-*`	Node-installer	Entire file	SUSE only, not `ifcfg-lo`
`/etc/sysconfig/network-scripts/ifcfg-*`	Node-installer	Entire file	Red Hat only, not `ifcfg-lo`
`/etc/HOSTNAME`	Node-installer	Entire file	

A.4 Files Not Generated, But Installed.

This appendix (Appendix A) is mainly about generated configuration files. This section (A.4) of the appendix discusses a class of files that is not generated, but may still be confused with generated files. The discussion in this section clarifies the issue, and explains how to check if non-generated installed files differ from the standard distribution installation.

A design goal of DirectMon is that of minimal interference. That is, to stay out of the way of the distributions that it works with as much as is reasonable. Still, there are inevitably cluster manager configuration files that are not generated, but installed from a cluster manager package. A cluster manager configuration file of this kind overwrites the distribution configuration file with its own special settings to get the cluster running, and the file is then not maintained by the node-installer or CMDaemon. Such files are therefore not listed on any of the tables in this chapter.

Sometimes the cluster file version may differ unexpectedly from the distribution version. To look into this, the following steps may be followed:

Is the configuration file a DirectMon version or a distribution version? A convenient way to check if a particular file is a cluster file version is to grep for it in the packages list for the cluster packages. For example, for `nsswitch.conf`:

```
[root@ddnmon61 ~]# repoquery -l $(repoquery -a | grep -F _cm) | grep\
 nsswitch.conf$
```

The inner `repoquery` displays a list of all the packages. By grepping for the cluster manager version string, for example `_cm` for DirectMon™, the list of cluster manager packages is found. The outer `repoquery` displays the list of files within each package in the list of cluster manager packages. By grepping for `nsswitch.conf$`, any file paths ending in `nsswitch.conf` in the cluster manager packages are displayed. The output is:

```
/cm/conf/etc/nsswitch.conf
/cm/conf/etc/nsswitch.conf
/cm/node-installer/etc/nsswitch.conf
```

Files under `/cm/conf` are placed by DirectMon packages when updating the head node. From there they are copied over during the post-install section of the RPM to where the distribution version configuration files are located by the cluster manager, but only during the initial installation of the cluster. The distribution version file is overwritten in this way to prevent RPM dependency conflicts of the DirectMon version with the distribution version. The configuration files are not copied over from `/cm/conf` during subsequent reboots after the initial installation. The `cm/conf` files are however updated when the DirectMon packages are updated. During such a DirectMon update, a notification is displayed that new configuration files are available.

Inverting the cluster manager version string match displays the files not provided by DirectMon. These are normally the files provided by the distribution:

```
[root@ddnmon61 ~]# repoquery -l $(repoquery -a | grep -F -v _cm) | \
grep nsswitch.conf$
/etc/nsswitch.conf
/etc/nsswitch.conf
/usr/share/doc/yp-tools-2.9/nsswitch.conf
```

Which package provides the file in DirectMon and in the distribution? The packages that provide these files can be found by running the "`yum whatprovides *`" command on the paths given by the preceding output, for example:

```
~# yum whatprovides */cm/conf/etc/nsswitch.conf
```

This reveals that some DirectMon LDAP packages can provide an `nsswitch.conf` file. The file is a plain file provided by the unpacking and placement that takes place when the package is installed. The file is not generated or maintained periodically after placement, which is the reason why this file is not seen in the tables of sections A.1, A.2, and A.3 of this appendix.

Similarly, looking through the output for

```
~# yum whatprovides */etc/nsswitch.conf
```

shows that `glibc` provides the distribution version of the `nsswitch.conf` file, and that there is also a node-installer version of this file available from the DirectMon packages.

What are the differences between the DirectMon version and the distribution versions of the file? Sometimes it is helpful to compare a distribution version and cluster version of `nsswitch.conf` to show the differences in configuration. The versions of the RPM packages containing the `nsswitch.conf` can be downloaded, their contents extracted, and their differences compared as follows:

```
~# mkdir yumextracted ; cd yumextracted
~# yumdownloader  glibc-2.12-1.107.el6.x86_64.rpm
~# rpm2cpio glibc-2.12-1.107.el6.x86_64.rpm | cpio -idmv
~# yumdownloader cm-config-ldap-client-6.0-45_cm.noarch.rpm
~# rpm2cpio cm-config-ldap-client-6.0-45_cm.noarch.rpm | cpio -idmv
~# diff etc/nsswitch.conf cm/conf/etc/nsswitch.conf
...
```

What are the configuration files in an RPM package? An RPM package allows files within it to be marked as configuration files. Files marked as configuration files can be listed with `rpm -qc` <*package*>. Optionally, piping the list through "`sort -u`" filters out duplicates.

Example

```
~# rpm -qc glibc | sort -u
/etc/ld.so.cache
/etc/ld.so.conf
/etc/localtime
/etc/nsswitch.conf
/usr/lib64/gconv/gconv-modules
/var/cache/ldconfig/aux-cache
```

How does an RPM installation deal with local configuration changes? Are there configuration files or critical files that DirectMon misses? Whenever an RPM installation detects a file with local changes, it can treat the local system file as if:

1. the local system file is frozen[1]. The installation does not interfere with the local file, but places the updated file as an .rpmnew file in the same directory.

[1]This freezing should not be confused with the `FrozenFile` directive (Appendix C), where the file or section of a file is being maintained by CMDaemon, and where freezing the file prevents CMDaemon from maintaining it.

2. the local system file is not frozen. The installation changes the local file. It copies the local file to an .rpmsave file in the same directory, and installs a new file from the RPM package.

When building the DirectMon packages, the package builders can specify which of these two methods apply. When dealing with the built package, the system administrator can use an `rpm` query method to determine which of the two methods applies for a particular file in the package. For example, for `glibc`, the following query can be used and grepped:

```
rpm -q --queryformat '[%{FILENAMES}\t%{FILEFLAGS:fflags}\n]' glibc | eg\
rep '[[:space:]].*(c|n).*$' | sort -u
/etc/ld.so.cache cmng
/etc/ld.so.conf cn
/etc/localtime  cn
/etc/nsswitch.conf    cn
/usr/lib64/gconv/gconv-modules  cn
/usr/lib/gconv/gconv-modules    cn
/var/cache/ldconfig/aux-cache   cmng
```

Here, the second column of the output displayed shows which of the files in the package have a configuration (`c`) flag or a noreplace (`n`) flag. The `c` flag without the `n` flag indicates that an .rpmsave file will be created, while a `c` flag together with an `n` flag indicates that an .rpmnew file will be created.

In any case, files that are not marked as configuration files are overwritten during installation.

So:

- If a file is not marked as a configuration file, and it has been customized by the system administrator, and this file is provided by an RPM package, and the RPM package is updated on the system, then the file is overwritten silently.
- If a file is marked as a configuration file, and it has been customized by the system administrator, and this file is provided by an RPM package, and the RPM package is updated on the system, then it is good practice to look for .rpmsave and .rpmnew versions of that file, and run a comparision on detection.

DirectMon should however mark all critical files as configuration files in the DirectMon packages.

Sometimes, RPM updates can overwrite a particular file that the administrator has changed locally and then would like to keep frozen.

To confirm that this is the problem, the following should be checked:

- The `--queryformat` option should be used to check that file can indeed be overwritten by updates. If the file has an `n` flag (regardless of whether it is a configuration file or not) then overwriting due to RPM updates does not happen, and the local file remains frozen. If the file has no `n` flag, then replacement occurs during RPM updates.

For files with no `n` flag, but where the administrator would still like to freeze the file during updates, the following can be considered:

- The file text content should be checked to see if it is a CMDaemon-maintained file (section 3.6.4), or checked against the list of generated files (Appendix A). This is just to make sure to avoid confusion about how changes are occurring in such a file.
  - If it is a CMDaemon-maintained file, then configuration changes put in by the administrator will also not persist in the maintained section of the file unless the `FrozenFile` directive (section C) is used to freeze the change.
  - If it is only a section that CMDaemon maintains, then configuration changes can be placed outside of the maintained section.

  Wherever the changes are placed in such a file, these changes are in any case by default overwritten on RPM updates if the file has no `n` flag.
- Some regular node updates can effectively be maintained in a desired state with the help of a finalize script (Appendix E).
- Updates can be excluded from YUM/zypper (section 10.3.2), thereby avoiding the overwriting of that file by the excluded package.

A request to change the package build flag may be sent to DDN if the preceding suggested options are unworkable.

B

DDN Public Key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.0 (GNU/Linux)

mQGiBEqtYegRBADStdQjn1XxbYorXbFGncF2IcMFiNA7hamARt4w7hjtwZoKGHbC
zSLsQTmgZO+FZs+tXcZa50LjGwhpxT6qhCe8Y7zIh2vwKrKlaAVKj2PUU28vKj1p
2W/OIiG/HKLtahLiCk0L3ahP0evJHh8B7elClrZOTKTBB6qIUbC5vHtjiwCgydm3
THLJsKnwk4qZetluTupldOEEANCzJ1nZxZzN6ZAMkIBrct8GivWClT1nBG4UwjHd
EDcGlREJxpg/OhpEP8TY1e0YUKRWvMqSVChPzkLUTIsd/O4RGTw0PGCo6Q3TLXpM
RVoonYPR1tRymPNZyW8VJeTUEn0kdlCaqZykp1sRb3jFAiJIRCmBRc854i/jRXmo
foTPBACJQyoEH9Qfe3VcqR6+vR2tX9lPvkxS7A5AnJIRs3Sv6yM4oV+7k/HrfYKt
fyl6widtEbQl870s4x3NYXmmne7lz1nGxBfAxzPG9rtjRSXyVxc+KGVd6gKeCV6d
o7kS/LJHRi0Lb5G4NZRFy5CGqg64liJwp/f2J4uyRbC8b+/LQbQ7QnJpZ2h0IENv
bXB1dGluZyBEZXZlbG9wbWVudCBUZWFtIDxkZXZAYnJpZ2h0Y29tcHV0aW5nLmNv
bT6IXgQTEQIAHgUCSq1h6AIbAwYLCQgHAwIDFQIDAxYCAQIeAQIXgAAKCRDvaS9m
+k3m0JO0AKC0GLTZiqoCQ6TRWW2ijjITEQ8CXACgg3o4oVbrG67VFzHUntcA0YTE
DXW5Ag0ESq1h6xAIAMJiaZI/0EqnrhSfiMsMT3sxz3mZkrQQL82Fob7s+S7nnMl8
A8btPzLlK8NzZytCglrIwPCYG6vfza/nkvyKEPh/f2it941bh7qiu4rBLqr+kGx3
zepSMRqIzW5FpIrUgDZOL9J+tWSSUtPW0YQ5jBBJrgJ8LQy9dK2RhAOLuHfbOSVB
JLIwNKxafkhMRwDoUNS4BiZKWyPFu47vd8fM67IPT1nMl0iCOR/QBn29MYuWnBcw
61344pd/IjOu3gM6YBqmRRU6yBeVi0TxxbYYnWcts6tEGAlTjHUOQ7gxVp4RDia2
jLVtbee8H464wxkkC3SSkng216RaBBAoaAykhzcAAwUH/iG4WsJHFw3+CRhUqy51
jnmb1FTFO8KQXI8JlPXM0h6vv0PtP5rw5D5V2cyVe2i4ez9Y8XMVfcbf60lptKyY
bRUjQq+9SNjt12ESU67YyLstSN68ach9Af03PoSZIKkiNwfA0+VBILv2Mhn7xd74
5L0M/eJ7lHSpeJA2Rzs6szc234Ob/VxGfGWjogaK3NElSYOzQo+/k0VMdMWsQm/8
Ras19IA9P5jlSbcZQlHlPjndS4x4XQ8P41ATczsIDyWhsJC51rTuw9/QO7fqvvPn
xsRz1pFmiiN7I4JLjw0nAlXexn4EaeVa7Eb+uTjvxJZNdShs7Td74OmlF7RKFccI
wLuISQQYEQIACQUCSq1h6wIbDAAKCRDvaS9m+k3m0C/oAJsHMmKrLPhjCdZyHbB1
e19+5JABUwCfU0PoawBN0HzDnfr3MLaTgCwjsEE=
=WJX7
-----END PGP PUBLIC KEY BLOCK-----
```

C

CMDaemon Configuration File Directives

This Appendix lists all configuration file directives that may be used in the cluster management daemon configuration file. If a change is needed, then the directives are normally changed on the head node, or head nodes in the case of a high availability configuration, in:

```
/cm/local/apps/cmd/etc/cmd.conf
```

The directives can also be set on the regular nodes, in the software image in `/cm/images/default-image/cm/local/apps/cmd/etc/cmd.conf` on the head node, but changing the defaults already there is not recommended.

To activate changes in a configuration file, the `cmd` service must be restarted. This is normally done with the command:

```
service cmd restart
```

Master directive

Syntax: `Master =` *hostname*
Default: `Master = master`

The cluster management daemon treats the host specified in the `Master` directive as the head node. A cluster management daemon running on a node specified as the head node starts in *head* mode. On a regular node, it starts in *node* mode.

Port directive

Syntax: `Port =` *number*
Default: `Port = 8080`

The *number* used in the syntax above is a number between 0 and 65535. The default value is 8080.

The `Port` directive sets the value of the port of the cluster management daemon to listen for non-SSL HTTP calls. By default, this happens only during init. All other communication with the cluster management daemon is carried out over the SSL port.

SSLPort directive

Syntax: `SSLPort = ` *number*
Default: `SSLPort = 8081`

The *number* used in the syntax above is a number between 0 and 65535. The default value is 8081.

The `SSLPort` directive sets the value of the SSL port of the cluster management daemon to listen for SSL HTTP calls. By default, it is used for all communication of CMDaemon with `cmgui` and `cmsh`, except for when CMDaemon is started up from init.

SSLPortOnly directive

Syntax: `SSLPortOnly = yes|no`
Default: `SSLPortOnly = no`

The `SSLPortOnly` directive allows the non-SSL port to be disabled. During normal running, both SSL and non-SSL ports are listening, but only the SSL port is used. By default, the non-SSL port is only used during CMDaemon start up.

CertificateFile directive

Syntax: `CertificateFile = ` *filename*
Default: `CertificateFile = "/cm/local/apps/cmd/etc/cmd.pem"`

The `CertificateFile` directive specifies the PEM-format certificate which is to be used for authentication purposes. On the head node, the certificate used also serves as a software license.

PrivateKeyFile directive

Syntax: `PrivateKeyFile = ` *filename*
Default: `PrivateKeyFile = "/cm/local/apps/cmd/etc/cmd.key"`

The `PrivateKeyFile` directive specifies the PEM-format private key which corresponds to the certificate that is being used.

CACertificateFile directive

Syntax: `CACertificateFile = ` *filename*
Default: `CACertificateFile = "/cm/local/apps/cmd/etc/cacert.pem"`

The `CACertificateFile` directive specifies the path to the DirectMon PEM-format root certificate. It is normally not necessary to change the root certificate.

RandomSeedFile directive

Syntax: `RandomSeedFile = ` *filename*
Default: `RandomSeedFile = "/dev/urandom"`

The `RandomSeedFile` directive specifies the path to a source of randomness.

DHParamFile directive

Syntax: `DHParamFile = ` *filename*
Default: `DHParamFile = "/cm/local/apps/cmd/etc/dh1024.pem"`

The `DHParamFile` directive specifies the path to the Diffie-Hellman parameters.

SSLHandshakeTimeout directive

Syntax: `SSLHandshakeTimeout = ` *number*
Default: `SSLHandshakeTimeout = 10`

The `SSLHandShakeTimeout` directive controls the time-out period (in seconds) for SSL handshakes.

SSLSessionCacheExpirationTime directive

Syntax: `SSLSessionCacheExpirationTime = ` *number*
Default: `SSLSessionCacheExpirationTime = 300`

The `SSLSessionCacheExpirationTime` directive controls the period (in seconds) for which SSL sessions are cached. Specifying the value `0` can be used to disable SSL session caching.

DBHost directive

Syntax: `DBHost = ` *hostname*
Default: `DBHost = "localhost"`

The `DBHost` directive specifies the hostname of the MySQL database server.

DBPort directive

Syntax: `DBPort = ` *number*
Default: `DBHost = 3306`

The `DBPort` directive specifies the TCP port of the MySQL database server.

DBUser directive

Syntax: `DBUser = ` *username*
Default: `DBUser = cmdaemon`

The `DBUser` directive specifies the username used to connect to the MySQL database server.

DBPass directive

Syntax: `DBPass = ` *password*

Default: `DBPass = "`*`<random string set during installation>`*`"`

The `DBPass` directive specifies the password used to connect to the MySQL database server.

DBName directive

Syntax: `DBName =` *database*
Default: `DBName = "cmdaemon"`

The `DBName` directive specifies the database used on the MySQL database server to store CMDaemon related configuration and status information.

DBMonName directive

Syntax: `DBMonName =` *database*
Default: `DBMonName = "cmdaemon_mon"`

The `DBMonName` directive specifies the database used on the MySQL database server to store monitoring related data.

DBUnixSocket directive

Syntax: `DBUnixSocket =` *filename*
Default: `DBUnixSocket = "/var/lib/mysql/mysql.sock"`

The `DBUnixSocket` directive specifies the named pipe used to connect to the MySQL database server if it is running on the same machine.

DBUpdateFile directive

Syntax: `DBUpdateFile =` *filename*
Default: `DBUpdateFile = "/cm/local/apps/cmd/etc/cmdaemon_upgrade.sql"`

The `DBUpdateFile` directive specifies the path to the file that contains information on how to upgrade the database from one revision to another.

EventBucket directive

Syntax: `EventBucket =` *filename*
Default: `EventBucket = "/var/spool/cmd/eventbucket"`

The `EventBucket` directive (section 11.6.2) specifies the path to the named pipe that is created to listen for incoming events from a user.

EventBucketFilter directive

Syntax: `EventBucketFilter =` *filename*
Default: `EventBucketFilter = "/cm/local/apps/cmd/etc/eventbucket.filter"`

The `EventBucketFilter` directive (section 11.6.2) specifies the path to the file that contains regular expressions used to filter out incoming messages on the event-bucket.

LDAPHost directive

Syntax: `LDAPHost =` *hostname*
Default: `LDAPHost = "localhost"`

The `LDAPHost` directive specifies the hostname of the LDAP server to connect to for user management.

LDAPUser directive

Syntax: `LDAPUser =` *username*
Default: `LDAPUser = "root"`

The `LDAPUser` directive specifies the username used when connecting to the LDAP server.

LDAPPass directive

Syntax: `LDAPPass =` *password*
Default: `LDAPPass = "`*`<random string set during installation>`*`"`

The `LDAPPass` directive specifies the password used when connecting to the LDAP server. It can be changed following the procedure described in Appendix O.

LDAPReadOnlyUser directive

Syntax: `LDAPReadOnlyUser =` *username*
Default: `LDAPReadOnlyUser = "readonlyroot"`

The `LDAPReadOnlyUser` directive specifies the username that will be used when connecting to the LDAP server during LDAP replication. The user is a member of the "rogroup" group, whose members have a read-only access to the whole LDAP directory.

LDAPReadOnlyPass directive

Syntax: `LDAPReadOnlyPass =` *password*
Default: `LDAPReadOnlyPass = "`*`<random string set during installation>`*`"`

The `LDAPReadOnlyPass` directive specifies the password that will be used when connecting to the LDAP server during LDAP replication.

LDAPSearchDN directive

Syntax: `LDAPSearchDN =` *dn*
Default: `LDAPSearchDN = "dc=cm,dc=cluster"`

The `LDAPSearchDN` directive specifies the Distinguished Name (DN) used when querying the LDAP server.

HomeRoot directive

Syntax: `HomeRoot = ` *path*
Default: `HomeRoot = "/home"`

The `HomeRoot` directive specifies the default user home directory used by CMDaemon. It is used for automatic mounts, exports, and when creating new users.

DocumentRoot directive

Syntax: `DocumentRoot = ` *path*
Default: `DocumentRoot = "/cm/local/apps/cmd/etc/htdocs"`

The `DocumentRoot` directive specifies the directory mapped to the web-root of the CMDaemon. The CMDaemon acts as a HTTP-server, and can therefore in principle also be accessed by web-browsers.

SpoolDir directive

Syntax: `SpoolDir = ` *path*
Default: `SpoolDir = "/var/spool/cmd"`

The `SpoolDir` directive specifies the directory which is used by the CMDaemon to store temporary and semi-temporary files.

EnableJSON directive

Syntax: `EnableJSON = true|false`
Default: `EnableJSON = true`

The `EnableJSON` directive allows `cmgui`. If the administrator or other user create other JSON-dependent utilities, these require the directive to be true too.

EnableShellService directive

Syntax: `EnableShellService = true|false`
Default: `EnableShellService = true`

The `EnableShellService` directive allows the use of the `Root Shell`, `Remote Console`, `Telnet Shell`, and `Ssh Shell` buttons in `cmgui` for connections to devices that allow such shells. The connection runs over CMDaemon, which is running over SSL, which means that between `cmgui` and the device, the connection is encrypted.

The directive does not affect `cmsh`'s `rshell`, `rconsole` `telnet`, and `ssh` commands.

CMDaemonAudit directive

Syntax: `CMDaemonAudit = yes|no`
Default: `CMDaemonAudit = no`

When the `CMDaemonAudit` directive is set to `yes`, and a value is set for the CMDaemon auditor file with the `CMDaemonAuditorFile` directive, then CMDaemon actions are time-stamped and logged in the CMDaemon auditor file.

CMDaemonAuditorFile directive

Syntax: `CMDaemonAuditorFile =` *filename*
Default: `CMDaemonAuditorFile = "/var/spool/cmd/audit.log"`

The `CMDaemonAuditorFile` directive sets where the audit logs for CMDaemon actions are logged. The log format is:
(*time stamp*) *profile* [*IP-address*] *action* (*unique key*)

Example

```
(Mon Jan 31 12:41:37 2011) Administrator [127.0.0.1] added Profile: arb\
itprof(4294967301)
```

DisableAuditorForProfiles directive

Syntax: `DisableAuditorForProfiles = {` *profile* [,*profile*]...`}`
Default: `DisableAuditorForProfiles = {node}`

The `DisableAuditorForProfiles` directive sets the profile for which an audit log for CMDaemon actions is disabled. A profile (section 3.3.4) defines the services that CMDaemon provides for that profile user. More than one profile can be set as a comma-separated list. Out of the profiles that are available on a newly-installed system: `node`, `admin`, `cmhealth`, and `readonly`; only the profile `node` is enabled by default. New profiles can also be created via the `profile` mode of `cmsh` or via the `Authorization` resource of `cmgui`, thus making it possible to disable auditing for arbitrary groups of CMDaemon services.

EventLogger directive

Syntax: `EventLogger = true|false`
Default: `EventLogger = false`

The `EventLogger` directive sets whether to log events. If active, then by default it logs events to `/var/spool/cmd/events.log` on the active head. If a failover takes place, then the event logs on both heads should be checked and merged for a complete list of events.

The location of the event log on the filesystem can be changed using the `EventLoggerFile` directive (page 586).

Whether events are logged in files or not, events are cached and accessible using `cmsh` or `cmgui`. The number of events cached by CMDaemon is determined by the parameter `MaxEventHistory` (page 586).

On a related note, `cmgui` users have access to a file in their home directory, with a relative path of `.cm/cmgui/cmgui.settings`. This file contains the `eventCache` parameter, with a default value of `256`. Modifying it changes the limit for the number of events viewed by `cmgui`, which is a front end to CMDaemon.

EventLoggerFile directive

Syntax: `EventLoggerFile =` *filename*
Default: `EventLogger = "/var/spool/cmd/events.log"`

The `EventLogger` directive sets where the events seen in the event viewer (section 11.6) are logged.

MaxEventHistory directive

Syntax: `AdvancedConfig = {"MaxEventHistory=`*number*`", ...}`
Default: `MaxEventHistory=8192`
`MaxEventHistory` is a parameter of the `AdvancedConfig` (page 594) directive.

By default, when not explicitly set, the maximum number of events that is retained by CMDaemon is 8192. Older events are discarded.

The parameter can take a value from 0 to 1000000. However, CMDaemon is less responsive with larger values, so in that case, setting the `EventLogger` directive (page 585) to true, to activate logging to a file, is advised instead.

PublicDNS directive

Syntax: `PublicDNS = true|false`
Default: `PublicDNS = false`

By default, internal hosts are resolved only if requests are from the internal network. Setting `PublicDNS` to true allows the head node name server to resolve internal network hosts for any network. Separate from this directive, port 53/UDP must also be opened up in Shorewall (section 13.2) if DNS is to be provided for queries from an external network.

LockDownDhcpd directive

Syntax: `LockDownDhcpd = true|false`
Default: `LockDownDhcpd = false`

`LockDownDhcpd` is a deprecated legacy directive. If set to true, a global DHCP "deny unknown-clients" option is set. This means no new DHCP leases are granted to unknown clients for all networks. Unknown clients are nodes for which DirectMon has no MAC addresses associated with the node. The directive `LockDownDhcpd` is deprecated because its globality affects clients on all networks managed by DirectMon, which is contrary to the general principle of segregating the network activity of networks.

The recommended way now to deny letting new nodes boot up is to set the option for specific networks by using `cmsh` or `cmgui` (section 4.3.1, figure 4.6, table 4.3.1). Setting the `cmd.conf LockDownDhcpd` directive overrides `lockdowndhcpd` values set by `cmsh` or `cmgui`.

MaxNumberOfProvisioningThreads directive

Syntax: `MaxNumberOfProvisioningThreads =` *number*
Default: `MaxNumberOfProvisioningThreads = 10000`

The `MaxNumberOfProvisioningThreads` directive specifies the cluster-wide total number of nodes that can be provisioned simultaneously. Individual provisioning servers typically define a much lower bound on the number of nodes that may be provisioned simultaneously.

SetupBMC directive

Syntax: `SetupBMC = true|false`
Default: `SetupBMC = true`

Configure the username and password for the BMC interface of the head node and regular nodes automatically. (This should not be confused with the `setupBmc` field of the node-installer configuration file, described in section 6.8.7.)

BMCSessionTimeout directive

Syntax: `BMCSessionTimeout =` *number*
Default: `BMCSessionTimeout = 2000`

The `BMCSessionTimeout` specifies the time-out for BMC calls in milliseconds.

SnmpSessionTimeout directive

Syntax: `SnmpSessionTimeout =` *number*
Default: `SnmpSessionTimeout = 500000`

The `SnmpSessionTimeout` specifies the time-out for SNMP calls in microseconds.

PowerOffPDUOutlet directive

Syntax: `PowerOffPDUOutlet = true|false`
Default: `PowerOffPDUOutlet = false`

Enabling the `PowerOffPDUOutlet` directive allows PDU ports to be powered off for clusters that have both PDU and IPMI power control. Section 5.1.3 has more on this.

MetricAutoDiscover directive

Syntax: `MetricAutoDiscover = true|false`
Default: `MetricAutoDiscover = true`

If this directive is enabled, then when the state of a regular device becomes `UP`, new metrics are scanned for on the device. If new metrics are discovered, then the CMDaemon monitoring configuration is updated to include these, and their data values are monitored with scheduled metrics and health checks.

MaxAgeForDataInMemory directive

Syntax: `MaxAgeForDataInMemory =` *number*
Default: `MaxAgeForDataInMemory = 3600`

Sets the maximum age, in seconds, for keeping monitoring data in memory. After that, the data will still be available, but with slightly more overhead, using MySQL retrieval.

MaxMemoryForBulkInsert directive

Syntax: `MaxMemoryForBulkInsert =` *number*
Default: `MaxMemoryForBulkInsert = 100000000`

Sets the maximum amount of memory, in bytes, reserved to buffer the new monitoring data temporarily in memory, before it is flushed to the database. The default value set for CMDaemon is 100MiB. If no value is specified in `cmd.conf`, then the value defaults to 500MiB. The flush action is also triggered if the period of `BulkInsertFlushInterval` is reached between flushes.

BulkInsertFlushInterval directive

Syntax: `BulkInsertFlushInterval =` *number*
Default: `BulkInsertFlushInterval = 300`

Sets the elapsed time, in seconds, before flushing the monitoring data from memory to the database. The flush action is also triggered if the memory buffer setting of `MaxMemoryForBulkInsert` is reached between flushes).

PDUMetricAutoDiscover directive

Syntax: `PDUMetricAutoDiscover = true|false`
Default: `PDUMetricAutoDiscover= true`

If this directive is enabled, then when the state of a PDU device becomes `UP`, new PDU metrics are scanned for on the device. If new PDU metrics are found, then the monitoring configuration is updated to include these, and their data values are monitored with scheduled metrics and health checks.

SensorMetricAutoDiscover directive

Syntax: `SensorMetricAutoDiscover = true|false`
Default: `SensorMetricAutoDiscover = true`

If this directive is enabled, then when the state of a rack sensor device becomes `UP`, new rack sensor metrics are scanned for on the device. If new rack sensor metrics are found, then the monitoring configuration is updated to include these, and their data values are monitored with scheduled metrics and health checks.

DisableBootLogo directive

Syntax: `DisableBootLogo = true|false`
Default: `DisableBootLogo = false`

When `DisableBootLogo` is set to `true`, the DirectMon logo is not displayed on the first boot menu when nodes PXE boot.

StoreBIOSTimeInUTC directive

Syntax: `StoreBIOSTimeInUTC = true|false`
Default: `StoreBIOSTimeInUTC = false`

When `StoreBIOSTimeInUTC` is set to `true`, the system relies on the time being stored in BIOS as being UTC rather than local time.

FreezeChangesToSGEConfig directive

Syntax: `FreezeChangesToSGEConfig = true|false`
Default: `FreezeChangesToSGEConfig = false`

When `FreezeChangesToSGEConfig` is set to `true`, CMDaemon does not make any modifications to the SGE configuration.

FreezeChangesToPBSProConfig directive

Syntax: `FreezeChangesToPBSProConfig = true|false`
Default: `FreezeChangesToPBSProConfig = false`

When `FreezeChangesToPBSProConfig` is set to `true`, CMDaemon does not make any modifications to the PBS Pro configuration.

FreezeChangesToTorqueConfig directive

Syntax: `FreezeChangesToTorqueConfig = true|false`
Default: `FreezeChangesToTorqueConfig = false`

When `FreezeChangesToTorqueConfig` is set to `true`, CMDaemon does not make any modifications to the Torque configuration.

FreezeChangesToSlurmConfig directive

Syntax: `FreezeChangesToSlurmConfig = true|false`
Default: `FreezeChangesToSlurmConfig = false`

When `FreezeChangesToSlurmConfig` is set to `true`, CMDaemon does not make any modifications to the Slurm configuration.

FreezeChangesToLSFConfig directive

Syntax: `FreezeChangesToLSFConfig = true|false`
Default: `FreezeChangesToLSFConfig = false`

When `FreezeChangesToLSFConfig` is set to `true`, then LSF configuration changes that would otherwise be carried out by CMDaemon as outlined in section 9.5.6, are not carried out on the following LSF server parameters:

- `prefix` (default: `/cm/shared/apps/lsf/current`): this points to the root directory of LSF.
- `var` (default: `/cm/shared/apps/lsf/var`): The LSF profile file, `profile.lsf`, is located at `/cm/shared/apps/lsf/var/conf/profile.lsf`, or `$var/conf/profile.lsf`.

If `FreezeChangesToLSFConfig` has the value `false`, then these parameters can be changed by CMDaemon.

FreezeChangesToOpenLavaConfig directive

Syntax: `FreezeChangesToOpenLavaConfig = true|false`
Default: `FreezeChangesToOpenLavaConfig = false`

When `FreezeChangesToOpenLavaConfig` is set to `true`, CMDaemon does not make any modifications to the openlava configuration.

FrozenFile directive

Syntax: `FrozenFile = {` *filename*[,*filename*]...`}`
Example: `FrozenFile = {"/etc/dhcpd.conf","/etc/postfix/main.cf"}`
The `FrozenFile` directive is used to prevent the CMDaemon-maintained sections of configuration files from being automatically generated. This is useful when site-specific modifications to configuration files have to be made.

To avoid problems, the file that is frozen should not be a symlink, but should be the ultimate destination file. The `readlink -f` <*symlinkname*> command returns the ultimate destination file of a symlink called <*symlinkname*>. This is also the case for an ultimate destination file that is reached via several chained symlinks.

A Note On The Necessity Of `FrozenFile`

Independent of whether `FrozenFile` is being used for a configuration file is the following useful behavior: A statement in an earlier part of a configuration file, for example, in `/etc/postfix/main.cf`:

```
mydomain = eth.cluster
```

can often be overridden by a statement later on:

```
mydomain = eth.gig.cluster
```

Whether overriding is possible depends on the software being configured. It is true for postfix configuration files, for example, but it may not be so for the configuration files of other applications.

If this behavior is true for the configuration file under consideration, then it also true when one statement is in the CMDaemon-maintained part and the other statement is outside the CMDaemon-maintained part.

The use of `FrozenFile` to override what CMDaemon enforces is thus sometimes avoidable by the use of such overriding statements.

EaseNetworkValidation directive

Syntax: `EaseNetworkValidation = 0|1|2`
Default: `EaseNetworkValidation = 0`

CMDaemon enforces certain requirements on network interfaces and management/node-booting networks by default. In heavily customized setups the user may wish to disable these requirements.

- `0` enforces all requirements.
- `1` allows violation of the requirements, with validation warnings. This value should never be set except under instructions from DDN support.
- `2` allows violation of the requirements, without validation warnings. This value should never be set except under instructions from DDN support.

CustomUpdateConfigFileScript directive

Syntax: `CustomUpdateConfigFileScript =` *filename*
Default: *commented out in the default* `cmd.conf` *file*

Whenever one or more entities have changed, the custom script at *filename*, specified by a full path, is called 30s later. Python bindings can be used to get information on the current setup.

ConfigDumpPath directive

Syntax: `ConfigDumpPath =` *filename*
Default: *commented out in the default* `cmd.conf` *file*

The `ConfigDumpPath` directive sets a dump file for dumping the configuration used by the power control script `/cm/local/apps/cmd/scripts/pctl/pctl`. The `pctl` script is a fallback script to allow power operations if CMDaemon is not running.

- If no directive is set, then no dump is done.
- If a directive is set, then the administrator must match the variable `cmdconfigfile` in the powercontrol configuration file `/cm/local/apps/cmd/scripts/pctl/config.py` to the value of `ConfigDumpPath`. By default, the value of `cmdconfigfile` is set to `/var/spool/cmd/cmdaemon.config.dump`.

SyslogHost directive

Syntax: `SyslogHost = ` *hostname*
Default: `SyslogHost = "localhost"`

The `SyslogHost` directive specifies the hostname of the syslog host.

SyslogFacility directive

Syntax: `SyslogFacility = ` *facility*
Default: `SyslogFacility = "LOG_LOCAL6"`

The default value of `LOG_LOCAL6` is set in:

- `/etc/syslog.conf` in Red Hat 5 and variants
- `/etc/rsyslog.conf` in Red Hat 6 and variants
- `/etc/syslog-ng/syslog-ng.conf` in SLES versions

These are the configuration files for the default syslog daemons `syslog`, `rsyslog`, and `syslog-ng`, respectively, that come with the distribution. DirectMon redirects messages from CMDaemon to `/var/log/cmdaemon` only for the default syslog daemon that the distribution provides. So, if another syslog daemon other than the default is used, then the administrator has to configure the non-default syslog daemon facilities manually.

The value of *facility* must be one of: `LOG_KERN`, `LOG_USER`, `LOG_MAIL`, `LOG_DAEMON`, `LOG_AUTH`, `LOG_SYSLOG` or `LOG_LOCAL0..7`

ResolveToExternalName directive

Syntax: `ResolveToExternalName = true|false`
Default: `ResolveToExternalName = false`

The value of the `ResolveToExternalName` directive determines under which domain name the primary and secondary head node hostnames are visible from within the head nodes, and to which IP addresses their hostnames are resolved. Enabling this directive resolves the head nodes' hostnames to the IP addresses of their external interfaces.

- On head nodes and regular nodes in both single-head and failover clusters with `ResolveToExternalName` disabled, the `master` hostname and the actual hostname of the head node (e.g. `head1`, `head2`) by default always resolve to the internal IP address of the head node.
- The `ResolveToExternalName` directive has no effect on the regular nodes unless they query the DNS on the head node, bypassing the `/etc/hosts` file. What this means is that with `ResolveToExternalName` enabled or disabled, the regular nodes resolve the hostname of any head node to an IP address located on the internal cluster network by default when resolved using the `/etc/hosts` file, for example using standard `ping` with the default node name resolution settings sequence. The regular nodes

however resolve according to the DNS server's configuration if querying the DNS server querying directly.

The following points describe more exhaustively how this directive influences the resolution of hostnames to IP addresses for head and regular nodes in the cluster:

- On the head node of a single-head cluster, if `ResolveToExternalName` is enabled, the `master` hostname and the actual hostname of the head node, e.g. `head`, then both resolve to the external IP address of the head node.
- On head nodes of a failover cluster, if `ResolveToExternalName` is enabled, the actual head hostnames, e.g. `head1` and `head2`, then also both resolve to their external IP address. However, the `master` hostname stays resolved to the internal shared IP address. This is because DirectMon uses `master` within its code base, and there is no need to have the failover configuration have a shared external IP address in the code base.
- On regular nodes in a single-head cluster, if `ResolveToExternalName` is enabled, then the `master` hostname or the actual hostname of the head node (e.g. `head`) both stay resolved to the internal IP address of the head node when using `/etc/hosts`. However they resolve to the external IP address of the head node when the DNS service is used.
- On regular nodes in a failover cluster, if `ResolveToExternalName` is enabled, the `master` hostname then still resolves to the shared internal IP. The actual hostnames of the head nodes, e.g. `head1` and `head2`, also still resolve to the internal IP address of the respective head nodes if resolved via `/etc/hosts`. However, they resolve to the external IP addresses of the respective head nodes if resolved by querying the DNS.

The behavior discussed in the preceding points can be summarized by Table C:

Table C: ResolveToExternalName Directive Effects

on simple head, resolving:		**on failover head, resolving:**			**on regular node, resolving:**		**Using the DNS?**
master	**head**	**master**	**head1**	**head2**	**master**	**head(s)**	
ResolveToExternalName = False							
I	I	I	I	I	I	I	No
I	I	I	I	I	I	I	Yes
ResolveToExternalName = True							
E	E	I	E	E	I	I	No
E	E	I	E	E	I	E	Yes

Key:	I: resolves to internal IP address of head
	E: resolves to external IP address of head

The system configuration files on the head nodes that get affected by this directive include `/etc/hosts` and, on SLES systems, also the `/etc/HOSTNAME`. Also, the DNS zone configuration files get affected.

Additionally, in both the single-head and failover clusters, using the "`hostname -f`" command on a head node while `ResolveToExternalName` is enabled results in the host's Fully Qualified Domain Domain (FQDN) being returned with the host's external domain name. That is, the domain name of the network that is specified as the "External network" in the base partition in `cmsh` (the output of "`cmsh -c "partition use base; get externalnetwork"`").

Modifying the value of the `ResolveToExternalName` directive and restarting the CMDaemon while important system services (e.g. Torque) are running should not be done. Doing so is likely to cause problems with accessing such services due to them then running with a different domain name than the one with which they originally started.

On a tangential note that is closely, but not directly related to the `ResolveToExternalName` directive: the cluster can be configured so that the "`hostname -f`" command executed on a regular node returns the FQDN of that node, and so that the FQDN in turn resolves to an external IP for that regular node. The details on how to do this are in the DirectMon Knowledge Base at `http://kb.ddn.com/`. A search query for FQDN leads to the relevant entry.

AdvancedConfig directive

Syntax: `AdvancedConfig = { "<key1>=<value1>", "<key2>=<value2>", ... }`
Default: *Commented out in the default* `cmd.conf` *file*
The `AdvancedConfig` directive is not part of the standard directives. It takes a set of key/value pairs as parameters, with each key/value pair allowing a particular functionality, and is quite normal in that respect. However, the functionality of a parameter to this directive is often applicable only under restricted conditions, or for non-standard configurations. The `AdvancedConfig` parameters are therefore generally not recommended, nor are they documented, for use by the administrator.

Like for the other directives, only one `AdvancedConfig` directive line is used. This means that whatever functionality is to be enabled by this directive, its corresponding parameters must be added to that one line. These key/value pairs are therefore added by appending them to any existing `AdvancedConfig` key/value pairs, which means that the directive line can be a long list of key/value pairs to do with a variety of configurations.

ScriptEnvironment directive

Syntax: `ScriptEnvironment = { "CMD_ENV1=<value1>", "CMD_ENV2=<value2>", ... }`
Default: *Commented out in the default* `cmd.conf` *file*

The `ScriptEnvironment` directive sets extra environment variables for CMDaemon and child processes.

For example, if CMDaemon is running behind a web proxy, then the environment variable `http_proxy` may need to be set for it. If, for example, the proxy is the host `brawndo`, and it is accessed via port 8080 using a username/password pair of `joe/electrolytes`, then the directive becomes:

```
ScriptEnvironment = { "http_proxy=joe:electrolytes@brawndo:8080" }
```

BurnSpoolDir directive

Syntax: `BurnSpoolDir =` *path*
Default: `BurnSpoolDir = "/var/spool/burn/"`

The `BurnSpoolDir` directive specifies the directory under which node burn log files are placed. The log files are logged under a directory named after the booting MAC address of the NIC of the node. For example, for a MAC address of 00:0c:29:92:55:5e the directory is `/var/spool/burn/00-0c-29-92-55-5e`.

IdleThreshold directive

Syntax: `IdleThreshold =` *number*
Default: `IdleThreshold = 1.0`

The `IdleThreshold` directive sets a threshold value for `loadone`. If `loadone` exceeds this value, then metric or health checks that have `Only when idle` (page 644) set to `true` will not run.

PhaseLoadMetrics directive

Syntax: `AdvancedConfig = {"PhaseLoadMetrics=`*scriptoutput*`", ...}`
Default: *none*

`PhaseLoadMetrics` is a parameter of the `AdvancedConfig` (page 594) directive.

By default, DirectMon sums up the phase load in amps for all APC-compatible PDUs. APC-compatible PDUs are expected to display an SNMP OID with the string `PowerNet-MIB`, like the PDUs described by APC MIB files at `ftp://ftp.apc.com/apc/public/software/pnetmib/mib/`.

Other PDUs are not supported by default, but can be added as generic devices, and this is recommended for several reasons (page 107). For these PDUs, their current use in amps can be retrieved by creating a custom script, say `load()`. So, if the PDUs, say PDU1, PDU2..., are sampled by the custom script running from the head node, this means that the script output, which is a list of currents, is: `load(PDU1), load(PDU2)`, ...

Then setting:

```
AdvancedConfig = { "PhaseLoadMetrics=load(PDU1),load(PDU2)..." }
```

sums the current for these PDUs, and adds it to the clusterwide `PhaseLoad` (page 640).

MaxServiceFailureCount directive

Syntax: `AdvancedConfig = {"MaxServiceFailureCount=`*number*`", ...}`
Default: Implicit value: `"MaxServiceFailureCount=10"`

`MaxServiceFailureCount` is a parameter of the `AdvancedConfig` (page 594) directive.

Its value determines the number of times a service failure event is logged (page 139). Restart attempts on the service still continue when this number is exceeded.

InitdScriptTimeout directive

Syntax: `AdvancedConfig = {"InitdScriptTimeout[.`*service*`]=`*timeout*`", ...}`
Default: Implicit value: `"InitdScriptTimeout=30"`

`InitdScriptTimeout` is a parameter of the `AdvancedConfig` (page 594) directive. It can be set globally or locally:

- **Global (all services)**
  `InitdScriptTimeout` can be set as a global value for init scripts, by assigning *timeout* as a period in seconds. If an init script fails to start up its service within this period, then CMDaemon kills the service and attempts to restart it.
  - If `InitdScriptTimeout` has a value for *timeout* set, then all init scripts have a default timeout of *timeout* seconds.
  - If `InitdScriptTimeout` has no *timeout* value set, then all init scripts have a default timeout of 30 seconds.
- **Local (for a specific service)**
  If `InitdScriptTimeout.`*service* is assigned a *timeout* value, then the init script for that *service* times out in *timeout* seconds. This timeout overrides, for that service only, any existing global default timeout.

When a timeout happens for an init script attempting to start a service, the event is logged. If the number of restart attempts exceeds the value determined by the `MaxServiceFailureCount` directive (page 596), then the event is no longer logged, but the restart attempts continue.

Example

An fhgfs startup takes a bit longer that 30 seconds, and therefore times out with the default timeout value of 30s. This results in the following logs in `/var/log/cmdaemon`:

```
cmd: [SERVICE] Debug: ProgramRunner: /etc/init.d/fhgfs-client restart
[DONE] 0 9
cmd: [SERVICE] Debug: /etc/init.d/fhgfs-client restart, exitcode = 0,
signal = 9
```

Here, *service* is `fhgfs-client`, so setting the parameter can be done with:

```
AdvancedConfig = { ..., "initdScriptTimeout.fhgfs-client=60", ...}
```

This allows a more generous timeout of 60 seconds instead.

Restarting CMDaemon then should allow the fhgs startup to complete

```
# service cmd restart
```

A more refined approach that avoids a complete CMDaemon restart could be to execute a `reset` (page 139) on the `fhgfs-client` from within CMDaemon, as follows:

```
[ddnmon61->category[default]->services[fhgfs-client]]% reset fhgfs-client
Successfully reset service fhgfs-client on: node001,node002
[ddnmon61->category[default]->services[fhgfs-client]]%
```

UserDefinedMetricClasses directive

Syntax:
`AdvancedConfig = {"UserDefinedMetricClasses=^`*file with metric scripts*`", ...}`
or
`AdvancedConfig = {"UserDefinedMetricClasses=`*metric1*`,`*metric2*`, ...", ...}`
Default: *none*

`UserDefinedMetricClasses` is a parameter of the `AdvancedConfig` (page 594) directive.

The parameter can be assigned metrics created by the user. The assignment can be done as:

- **a file path:** The file is then a text file with a metric name on each line
- **a list of values:** Each list item is then a metric name.

Example

```
AdvancedConfig = { "UserDefinedMetricClasses=sfa_pool,sfa_vd,sfa_hc,sfa\
_sys,GridScalerNodeStats,GridScalerFileSystemStats,ExaScalerNodeStats,E\
xaScalerOSTStats,ExaScalerMDTStats,ExaScalerMGTStats,ExaScalerOSTBRWSta\
ts,ExaScalerLNET" }
```

DeviceIsUpPowerSaveFail directive

Syntax: `AdvancedConfig = {"DeviceIsUpPowerSaveFail = 0|1", ...}`
Default: `1`

`DeviceIsUpPowerSaveFail` is a parameter of the `AdvancedConfig` (page 594) directive.

It affects the behavior of the `DeviceIsUp` (Appendix H.2.1) health check. If the health check is run for a device that is in a `DOWN` (section 6.5) state, then:

- If `DeviceIsUpPowerSaveFail` is `1` (the default), the healthcheck returns a `FAIL`, always.
- If `DeviceIsUpPowerSaveFail` is `0`, the healthcheck returns a `PASS` if the `DOWN` state is due to Slurm Power Save.

  Slurm Power Save (section 9.9.1) is not active in a default installation of DirectMon.

CMDaemonListenOnInterfaces directive

Syntax: `AdvancedConfig = {"CMDaemonListenOnInterfaces = <interfaces>",...}`
Default: *all interfaces listening to port 8081*

`CMDaemonListenOnInterfaces` is a parameter of the `AdvancedConfig` (page 594) directive.

When set explicitly, CMDaemon listens only to the interfaces listed in `<interfaces>`. The form of `<interfaces>` is a comma-separated list of interface device names:

Example

```
CMDaemonListenOnInterfaces=eth0,eth1,eth0:0,eth0:1
```

If the interface list item `lo` is omitted from the list of names, it will still be listened to. This is because CMDaemon must always be able to talk to itself on the loopback interface.

D Disk Partitioning

DirectMon requires that disk partitionings are specified using the XML format that is described in section D.1.

Disk partitioning is initially implemented on the head node and regular nodes during installation (section 2.3.14).

For the head node it cannot be changed from within the DirectMon after implementation.

For regular nodes partitioning can be changed after the initial implementation, by changing the XML file defining their partitioning scheme. Diskless operation can also be implemented by using an appropriate XML file.

D.1 Structure Of Partitioning Definition

In DirectMon, partitioning setups have their global structure defined using an XML schema. The schema file is installed on the head node in `/cm/node-installer/scripts/disks.xsd`. This section shows the schema and gives some examples of the defined types, while the next sections contain examples of the schema in use:

XML schema for partitioning

```
<?xml version='1.0'?>

<!--
#
# Copyright (c) 2004-2010 DDN, Inc. All Rights Reserved.
#
# This software is the confidential and proprietary information of
# DDN, Inc.("Confidential Information").  You shall not
# disclose such Confidential Information and shall use it only in
# accordance with the terms of the license agreement you entered into
# with DDN, Inc.

This is the XML schema description of the partition layout XML file.
It can be used by software to validate partitioning XML files.
There are however a few things the schema does not check:
- There should be exactly one root mountpoint (/), unless diskless.
- There can only be one partition with a 'max' size on a particular device.
- Something similar applies to logical volumes.
- The 'auto' size can only be used for a swap partition.
- Partitions of type 'linux swap' should not have a filesystem.
- Partitions of type 'linux raid' should not have a filesystem.
- Partitions of type 'linux lvm' should not have a filesystem.
- Partitions of type 'unspecified' should not have a filesystem.
- If a raid is a member of another raid then it can not have a filesystem.
- Partitions, which are listed as raid members, should be of type 'linux raid'.
- If diskless is not set, there should be at least one device.
- The priority tag is only valid for partitions which have type set to
  "linux swap".
-->

<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema' elementFormDefault='qualified'>

  <xs:element name='diskSetup'>

    <xs:complexType>
      <xs:sequence>
        <xs:element name='diskless' type='diskless' minOccurs='0' maxOccurs='1'/>
        <xs:element name='device' type='device' minOccurs='0' maxOccurs='unbounded'/>
        <xs:element name='raid' type='raid' minOccurs='0' maxOccurs='unbounded'/>
        <xs:element name='volumeGroup' type='volumeGroup' minOccurs='0' maxOccurs='unbounded'/>
      </xs:sequence>
    </xs:complexType>

    <xs:key name='partitionAndRaidIds'>
      <xs:selector xpath='.//raid|.//partition'/>
      <xs:field xpath='@id'/>
    </xs:key>

    <xs:keyref name='raidMemberIds' refer='partitionAndRaidIds'>
      <xs:selector xpath='.//raid/member'/>
      <xs:field xpath='.'/>
    </xs:keyref>

    <xs:keyref name='volumeGroupPhysicalVolumes' refer='partitionAndRaidIds'>
      <xs:selector xpath='.//volumeGroup/physicalVolumes/member'/>
      <xs:field xpath='.'/>
    </xs:keyref>

    <xs:unique name='raidAndVolumeMembersUnique'>
      <xs:selector xpath='.//member'/>
      <xs:field xpath='.'/>
    </xs:unique>

    <xs:unique name='deviceNodesUnique'>
      <xs:selector xpath='.//device/blockdev'/>
      <xs:field xpath='.'/>
    </xs:unique>

    <xs:unique name='mountPointsUnique'>
      <xs:selector xpath='.//mountPoint'/>
      <xs:field xpath='.'/>
    </xs:unique>
    <xs:unique name='assertNamesUnique'>
      <xs:selector xpath='.//assert'/>
      <xs:field xpath='@name'/>
    </xs:unique>

  </xs:element>

  <xs:complexType name='diskless'>
    <xs:attribute name='maxMemSize' type='memSize' use='required'/>
  </xs:complexType>

  <xs:simpleType name='memSize'>
    <xs:restriction base='xs:string'>
      <xs:pattern value='([0-9]+[MG])|100%|[0-9][0-9]%|[0-9]%|0'/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name='size'>
    <xs:restriction base='xs:string'>
      <xs:pattern value='max|auto|[0-9]+[MGT]'/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name='extentSize'>
    <xs:restriction base='xs:string'>
      <xs:pattern value='([0-9])+M'/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name='blockdev'>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="mode" default='normal'>
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:pattern value="normal|cloud|both"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name='device'>
    <xs:sequence>
      <xs:element name='blockdev' type='blockdev' minOccurs='1' maxOccurs='unbounded'/>
      <xs:element name='vendor' type='xs:string' minOccurs='0' maxOccurs='1'/>
      <xs:element name='requiredSize' type='size' minOccurs='0' maxOccurs='1'/>
      <xs:element name='assert' minOccurs='0' maxOccurs='unbounded'>
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base='xs:string'>
              <xs:attribute name='name' use='required'>
                <xs:simpleType>
                  <xs:restriction base='xs:string'>
                    <xs:pattern value='[a-zA-Z0-9-_]+'/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:attribute>
              <xs:attribute name='args' type='xs:string'/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:element name='alignMiB' type='xs:boolean' minOccurs='0' maxOccurs='1'/>
      <xs:element name='partition' type='partition' minOccurs='1' maxOccurs='unbounded'/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name='partition'>
    <xs:sequence>
      <xs:element name='size' type='size'/>
      <xs:element name='type'>
        <xs:simpleType>
          <xs:restriction base='xs:string'>
            <xs:enumeration value='linux'/>
            <xs:enumeration value='linux swap'/>
            <xs:enumeration value='linux raid'/>
            <xs:enumeration value='linux lvm'/>
            <xs:enumeration value='unspecified'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:group ref='filesystem' minOccurs='0' maxOccurs='1'/>
      <xs:element name='priority' minOccurs='0' maxOccurs='1'>
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="32767"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name='id' type='xs:string' use='required'/>
  </xs:complexType>

  <xs:group name='filesystem'>
    <xs:sequence>
      <xs:element name='filesystem'>
        <xs:simpleType>
          <xs:restriction base='xs:string'>
            <xs:enumeration value='ext2'/>
            <xs:enumeration value='ext3'/>
            <xs:enumeration value='ext4'/>
            <xs:enumeration value='xfs'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name='mkfsFlags' type='xs:string' minOccurs='0' maxOccurs='1'/>
```

```
      <xs:element name='mountPoint' type='xs:string'/>
      <xs:element name='mountOptions' type='xs:string' default='defaults'/>
    </xs:sequence>
  </xs:group>

  <xs:complexType name='raid'>
    <xs:sequence>
      <xs:element name='member' type='xs:string' minOccurs='2' maxOccurs='unbounded'/>
      <xs:element name='level' type='xs:int'/>
      <xs:choice minOccurs='0' maxOccurs='1'>
        <xs:group ref='filesystem'/>
        <xs:sequence>
          <xs:element name='swap'><xs:complexType /></xs:element>
          <xs:element name='priority' minOccurs='0' maxOccurs='1'>
            <xs:simpleType>
              <xs:restriction base="xs:integer">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="32767"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
        </xs:sequence>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name='id' type='xs:string' use='required'/>
  </xs:complexType>

  <xs:complexType name='volumeGroup'>
    <xs:sequence>
      <xs:element name='name' type='xs:string'/>
      <xs:element name='extentSize' type='extentSize'/>
      <xs:element name='physicalVolumes'>
        <xs:complexType>
          <xs:sequence>
            <xs:element name='member' type='xs:string' minOccurs='1' maxOccurs='unbounded'/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name='logicalVolumes'>
        <xs:complexType>
          <xs:sequence>
            <xs:element name='volume' type='logicalVolume' minOccurs='1' maxOccurs='unbounded'/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name='logicalVolume'>
    <xs:sequence>
      <xs:element name='name' type='xs:string'/>
      <xs:element name='size' type='size'/>
      <xs:group ref='filesystem' minOccurs='0' maxOccurs='1'/>
    </xs:sequence>
  </xs:complexType>
```

```
</xs:schema>
```

Examples Of Element Types In XML Schema

Name Of Type	Example Values
`size`	10G, 128M, 1T, auto, max
`device`	/dev/sda, /dev/hda, /dev/cciss/c0d0
`partition`	linux, linux raid, linux swap, unspecified
`filesystem`	ext2, ext3, ext4, xfs

D.2 Example: Default Node Partitioning

The following example shows the default layout used for regular nodes:

Example

```
<?xml version="1.0" encoding="UTF-8"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <device>
    <blockdev>/dev/sda</blockdev>
    <blockdev>/dev/hda</blockdev>
    <blockdev>/dev/vda</blockdev>
    <blockdev>/dev/xvda</blockdev>
    <blockdev>/dev/cciss/c0d0</blockdev>
    <blockdev mode="cloud">/dev/sdb</blockdev>
    <blockdev mode="cloud">/dev/hdb</blockdev>
    <blockdev mode="cloud">/dev/vdb</blockdev>
    <blockdev mode="cloud">/dev/xvdb</blockdev>
    <blockdev mode="cloud">/dev/xvdf</blockdev>
    <partition id="a1">
      <size>20G</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
    <partition id="a2">
      <size>6G</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/var</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
    <partition id="a3">
      <size>2G</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/tmp</mountPoint>
      <mountOptions>defaults,noatime,nodiratime,nosuid,nodev</mountOptions>
    </partition>
    <partition id="a4">
      <size>12G</size>
      <type>linux swap</type>
    </partition>
    <partition id="a5">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/local</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
  </device>
</diskSetup>
```

The example assumes a single disk. Another disk can be added by adding another pair of `<device><device>` tags and filling in the partitioning specifications for the next disk. Because multiple `blockdev` tags are used, the node-installer first tries to use `/dev/sda`, then `/dev/hda`, then `/dev/vda` (virtio disks), then `/dev/xvda` (xen disks), and so on. Cloud devices can also be accessed using the `mode="cloud"` option. Removing block devices from the layout if they are not going to be used does no harm.

For each `partition`, a `size` tag is specified. Sizes can be specified using megabytes (e.g. `500M`), gigabytes (e.g. `50G`) or terabytes (e.g. `2T`). Alternatively, for a device, the attribute `max` for a `size` tag forces the last device in the partition to use all remaining space, and if needed, adjusts the implementation of the sequence of `size` tags in the remaining partitions accordingly. For swap partitions, a size of `auto` sets a swap partition to twice the node memory size. If there is more than one swap partition, then the `priority` tag can be set so that the partition with the higher priority is used first.

In the example, all file systems are specified as `ext3`. Valid alternatives are `ext2` and `xfs`.

The `mount` man page has more details on mount options. If the `mountOptions` tag is left empty, its value defaults to `defaults`.

D.3 Example: Software RAID

The following example shows a simple software RAID setup:

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="schema.xsd">

  <device>
    <blockdev>/dev/sda</blockdev>
    <partition id="a1">
      <size>25G</size>
      <type>linux raid</type>
    </partition>
  </device>
  <device>
    <blockdev>/dev/sdb</blockdev>
    <partition id="b1">
      <size>25G</size>
      <type>linux raid</type>
    </partition>
  </device>

  <raid id="r1">
    <member>a1</member>
    <member>b1</member>
    <level>1</level>
    <filesystem>ext3</filesystem>
    <mountPoint>/</mountPoint>
    <mountOptions>defaults,noatime,nodiratime</mountOptions>
  </raid>

</diskSetup>
```

The `level` tag specifies the RAID level used. The following are supported:

- `0` (striping without parity)
- `1` (mirroring)
- `4` (striping with dedicated parity drive)
- `5` (striping with distributed parity)
- `6` (striping with distributed double parity)

The `member` tags must refer to an `id` attribute of a `partition` tag, or an `id` attribute of a another `raid` tag. The latter can be used to create, for example, RAID 10 configurations.

The administrator must ensure that the correct RAID kernel module is loaded (section 6.3.2). Including the appropriate module from the following is usually sufficient: `raid0`, `raid1`, `raid4`, `raid5`, `raid6`.

D.4 Example: Software RAID With Swap

The `<swap></swap>` tag is used to indicate a swap partition in a RAID device specified in the XML schema of section D.1. For example, the following marks a 1GB RAID 1 partition as being used for swap, and the second partition for an ext3 filesystem:

```
<?xml version="1.0" encoding="UTF-8"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <device>
    <blockdev>/dev/sda</blockdev>
    <partition id="a1">
      <size>1G</size>
      <type>linux raid</type>
    </partition>
    <partition id="a2">
      <size>max</size>
      <type>linux raid</type>
    </partition>
  </device>
  <device>
    <blockdev>/dev/sdb</blockdev>
    <partition id="b1">
      <size>1G</size>
      <type>linux raid</type>
    </partition>
    <partition id="b2">
      <size>max</size>
      <type>linux raid</type>
    </partition>
  </device>
  <raid id="r1">
    <member>a1</member>
    <member>b1</member>
    <level>1</level>
    <swap></swap>
  </raid>
  <raid id="r2">
    <member>a2</member>
    <member>b2</member>
    <level>1</level>
    <filesystem>ext3</filesystem>
    <mountPoint>/</mountPoint>
    <mountOptions>defaults,noatime,nodiratime</mountOptions>
  </raid>
</diskSetup>
```

As in section D.3, the appropriate RAID modules must be loaded beforehand.

D.5 Example: Logical Volume Manager

This example shows a simple LVM setup:

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <device>
    <blockdev>/dev/sda</blockdev>
    <blockdev>/dev/hda</blockdev>
    <blockdev>/dev/vda</blockdev>
    <blockdev>/dev/xvda</blockdev>
    <blockdev>/dev/cciss/c0d0</blockdev>
    <blockdev mode="cloud">/dev/sdb</blockdev>
    <blockdev mode="cloud">/dev/hdb</blockdev>
    <blockdev mode="cloud">/dev/vdb</blockdev>
    <blockdev mode="cloud">/dev/xvdb</blockdev>
    <blockdev mode="cloud">/dev/xvdf</blockdev>
    <partition id="a1">
      <size>512M</size>
      <type>linux</type>
      <filesystem>ext2</filesystem>
      <mountPoint>/boot</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
    <partition id="a2">
      <size>16G</size>
      <type>linux swap</type>
    </partition>
    <partition id="a3">
      <size>max</size>
      <type>linux lvm</type>
    </partition>
  </device>
  <volumeGroup>
    <name>vg01</name>
    <extentSize>8M</extentSize>
    <physicalVolumes>
      <member>a3</member>
    </physicalVolumes>
    <logicalVolumes>
      <volume>
        <name>lv00</name>
        <size>max</size>
        <filesystem>ext3</filesystem>
        <mountPoint>/</mountPoint>
        <mountOptions>defaults,noatime,nodiratime</mountOptions>
      </volume>
      <volume>
        <name>lv01</name>
        <size>8G</size>
        <filesystem>ext3</filesystem>
        <mountPoint>/tmp</mountPoint>
        <mountOptions>defaults,noatime,nodiratime</mountOptions>
      </volume>
    </logicalVolumes>
  </volumeGroup>
</diskSetup>
```

The `member` tags must refer to an `id` attribute of a `partition` tag, or an `id` attribute of a `raid` tag.

The administrator must ensure that the `dm-mod` kernel module is loaded when LVM is used.

D.6 Example: Diskless

This example shows a node configured for diskless operation:

Example

```
<?xml version="1.0" encoding="UTF-8"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <diskless maxMemSize="90%"></diskless>
</diskSetup>
```

An example of the implementation of a diskless configuration is given in section 4.10.3.

In diskless mode the software image is transferred by the node-installer to a RAM-based filesystem on the node called `tmpfs`.

The obvious advantage of running from RAM is the elimination of the physical disk, cutting power consumption and reducing the chance of hardware failure. On the other hand, some of the RAM on the node is then no longer available for user applications.

Special considerations with diskless mode:

- **Recommended minimum RAM size:** The available RAM per node should be sufficient to run the OS and the required tasks. At least 4GB is recommended for diskless nodes.
- **The `tmpfs` size limit:** The maximum amount of RAM that can be used for a file system is set with the `maxMemSize` attribute. A value of `100%` allows all of the RAM to be used. The default value is `90%`. A value of `0`, without the % sign, removes all restrictions.

  A limit does not however necessarily prevent the node from crashing, as some processes might not deal properly with a situation when there is no more space left on the filesystem.
- **Persistence issues:** While running as a diskless node, the node is unable to retain any non-shared data each time it reboots. For example the files in `/var/log/*`, which are normally preserved by the exclude list settings for disked nodes, are lost from RAM during diskless mode reboots.
- **Leftover disk issues:** Administrators in charge of sensitive environments should be aware that the disk of a node that is now running in diskless mode still contains files from the last time the disk was used, unless the files are explicitly wiped.
- **Reducing the software image size in `tmpfs` on a diskless node:** To make more RAM available for tasks, the software image size held in RAM can be reduced:
  - by removing unnecessary software from the image.
  - by mounting parts of the filesystem in the image over NFS during normal use. This is especially worthwhile for less frequently accessed parts of the image (section 4.11.3).

D.7 Example: Semi-diskless

Diskless operation (section D.6) can also be mixed with certain parts of the file system on the local physical disk. This frees up RAM which the node can then put to other use. In this example all data in `/local` is on the physical disk, the rest in RAM.

Example

```
<?xml version="1.0" encoding="UTF-8"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="schema.xsd">
```

```
  <diskless maxMemSize="0"></diskless>
  <device>
    <blockdev>/dev/sda</blockdev>
    <partition id="a1">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/local</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
  </device>
</diskSetup>
```

When nodes operate in semi-diskless mode the node-installer always uses `excludelistfullinstall` (section 6.4.7) when synchronizing the software image to memory and disk.

An alternative to using a local disk for freeing up RAM is to use NFS storage, as is described in section 4.11.3.

D.8 Example: Preventing Accidental Data Loss

Optional tags, `vendor` and `requiredSize`, can be used to prevent accidentally repartitioning the wrong drive. Such a tag use is shown in the following example.

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="schema.xsd">
  <device>
    <blockdev>/dev/sda</blockdev>
    <vendor>Hitachi</vendor>
    <requiredSize>200G</requiredSize>

    <partition id="a1">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>

  </device>
  <device>
    <blockdev>/dev/sdb</blockdev>
    <vendor>BigRaid</vendor>
    <requiredSize>2T</requiredSize>

    <partition id="b1">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/data</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>

  </device>
</diskSetup>
```

If a vendor or a requiredSize element is specified, it is treated as an assertion which is checked by the node-installer. The node-installer reads the drive vendor string from /sys/block/<drive name>/device/vendor. The actual disk size, as seen by fdisk -l for example, should be from 85% to about 118% of the value specified by requiredSize for the assertion to succeed.

If any assertion fails, no partitioning changes will be made to any of the specified devices.

Specifying device assertions is recommended for machines that contain important data because it protects against a situation where a drive is assigned to an incorrect block device. This can happen, for example, when the first drive in a multi-drive system is not detected (e.g. due to a hardware failure) which could cause the second drive to become known as /dev/sda, potentially causing much woe.

As an aside, CMDaemon does offer another way, outside of assertions, to avoid wiping out drive data automatically. This is by setting the requirefullinstallconfirmation toggle, as discussed in section 6.4.4.

D.9 Example: Using Custom Assertions

The following example shows the use of the assert tag, which can be added to a device definition:

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="schema.xsd">
  <device>
    <blockdev>/dev/sda</blockdev>
    <assert name="modelCheck" args="WD800AAJS">
      <![CDATA[
        #!/bin/bash
        if grep -q $1 /sys/block/$ASSERT_DEV/device/model; then
          exit 0
        else
          exit 1
        fi
      ]]>
    </assert>

    <partition id="a1">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
  </device>
  <device>
    <blockdev>/dev/sdb</blockdev>
    <vendor>BigRaid</vendor>
    <requiredSize>2T</requiredSize>

    <partition id="b1">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/data</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>

  </device>
</diskSetup>
```

The `assert` tag is similar to the `vendor` and `size` tags described in section D.8.

It can be used to define custom assertions. The assertions can be implemented using any script language.

The script will have access to the environment variables `ASSERT_DEV` (i.e. sda) and `ASSERT_NODE` (i.e. /dev/sda).

Each assert needs to be assigned an arbitrary name and can be passed custom parameters. A non-zero exit code in the assertion causes the node-installer to halt.

E
Example `initialize` And `finalize` Scripts

The node-installer executes any `initialize` and `finalize` scripts at particular stages of its 13-step run during node-provisioning (section 6.4). They are sometimes useful for troubleshooting or workarounds during those stages. The scripts are stored in the CMDaemon database, rather than in the filesystem as plain text files, because they run before the node's `init` process takes over and establishes the final filesystem.

E.1 When Are They Used?

They are sometimes used as an alternative configuration option out of a choice of other possible options (section 4.16.1). Sometimes there is no reasonable alternative other than using an `initialize` or `finalize` script.

An `initialize` script: is used well before the `init process` starts, to execute custom commands before partitions and mounting devices are checked. Typically, `initialize` script commands are related to partitioning, mounting, or initializing special storage hardware. Often an `initialize` script is needed because the commands in it cannot be stored persistently anywhere else.

A `finalize` script: (also run before `init`, but shortly before `init` starts) is used to set a file configuration or to initialize special hardware, sometimes after a hardware check. It is run in order to make software or hardware work before, or during the later `init` stage of boot. Thus, often a `finalize` script is needed because its commands must be executed before `init`, and the commands cannot be stored persistently anywhere else, or it is needed because a choice between (otherwise non-persistent) configuration files must be made based on the hardware before `init` starts.

E.2 Accessing From `cmgui` And `cmsh`

The `initialize` and `finalize` scripts are accessible for viewing and editing:

- In cmgui, using the "Node Categories" or Nodes resource, under the Settings tabbed pane for the selected item.
- In cmsh, using the category or device modes. The get command is used for viewing the script, and the set command to start up the default text editor to edit the script. Output is truncated in the two following examples at the point where the editor starts up:

Example

```
[root@ddnmon61 ~]# cmsh
[ddnmon61]% category use default
[ddnmon61->category[default]]% show | grep script
Parameter                       Value
------------------------------ ------------------------------------------------
Finalize script                 <1367 bytes>
Initialize script               <0 bytes>
[ddnmon61->category[default]]% set initializescript
```

Example

```
[ddnmon61]% device use node001
[ddnmon61->device[node001]]%
[ddnmon61->device[node001]]% set finalizescript
```

E.3 Environment Variables Available To initialize And finalize Scripts

For the initialize and finalize scripts, node-specific customizations can be made from a script using environment variables. The following table shows the available variables with some example values:

Table E: Environment Variables For The initialize And Finalize Scripts

Variable	Example Value
CMD_ACTIVE_MASTER_IP	10.141.255.254
CMD_CATEGORY	default
CMD_CHASSIS	chassis01
CMD_CHASSIS_IP	10.141.1.1
CMD_CHASSIS_PASSWORD	ADMIN
CMD_CHASSIS_SLOT	1
CMD_CHASSIS_USERNAME	ADMIN

...continues

Table E: Environment Variables For The `initialize` *And* `Finalize` *Scripts...continued*

Variable	Example Value
CMD_CLUSTERNAME	DDN Cluster
CMD_DEVICE_HEIGHT	1
CMD_DEVICE_POSITION	10
CMD_DEVICE_TYPE	SlaveNode
CMD_ETHERNETSWITCH	switch01:1
CMD_FSEXPORT__SLASH_cm_SLASH\ _node-installer_ALLOWWRITE	no
CMD_FSEXPORT__SLASH_cm_SLASH\ _node-installer_HOSTS	10.141.0.0/16
CMD_FSEXPORT__SLASH_cm_SLASH\ _node-installer_PATH	/cm/node-installer
CMD_FSEXPORTS	_SLASH_cm_SLASH_node-installer
CMD_FSMOUNT__SLASH_cm_SLASH\ _shared_DEVICE	master:/cm/shared
CMD_FSMOUNT__SLASH_cm_SLASH\ _shared_FILESYSTEM	nfs
CMD_FSMOUNT__SLASH_cm_SLASH\ _shared_MOUNTPOINT	/cm/shared
CMD_FSMOUNT__SLASH_cm_SLASH\ _shared_OPTIONS	rsize=32768,wsize=32768,\ hard,intr,async
CMD_FSMOUNT__SLASH_dev_SLASH\ _pts_DEVICE	none
CMD_FSMOUNT__SLASH_dev_SLASH\ _pts_FILESYSTEM	devpts
CMD_FSMOUNT__SLASH_dev_SLASH\ _pts_MOUNTPOINT	/dev/pts
CMD_FSMOUNT__SLASH_dev_SLASH\ _pts_OPTIONS	gid=5,mode=620
CMD_FSMOUNT__SLASH_dev_SLASH\ _shm_DEVICE	none
CMD_FSMOUNT__SLASH_dev_SLASH\ _shm_FILESYSTEM	tmpfs

...continues

Table E: Environment Variables For The initialize *And* Finalize *Scripts...continued*

Variable	Example Value
CMD_FSMOUNT__SLASH_dev_SLASH\ _shm_MOUNTPOINT	/dev/shm
CMD_FSMOUNT__SLASH_dev_SLASH\ _shm_OPTIONS	defaults
CMD_FSMOUNT__SLASH\ _home_DEVICE	master:/home
CMD_FSMOUNT__SLASH_home\ _FILESYSTEM	nfs
CMD_FSMOUNT__SLASH_home\ _MOUNTPOINT	home
CMD_FSMOUNT__SLASH_home\ _OPTIONS	rsize=32768,wsize=32768,\ hard,intr,async
CMD_FSMOUNT__SLASH_proc\ _DEVICE	none
CMD_FSMOUNT__SLASH_proc\ _FILESYSTEM	proc
CMD_FSMOUNT__SLASH_proc\ _MOUNTPOINT	/proc
CMD_FSMOUNT__SLASH_proc\ _OPTIONS	defaults,nosuid
CMD_FSMOUNT__SLASH_sys\ _DEVICE	none
CMD_FSMOUNT__SLASH_sys\ _FILESYSTEM	sysfs
CMD_FSMOUNT__SLASH_sys\ _MOUNTPOINT	/sys
CMD_FSMOUNT__SLASH_sys\ _OPTIONS	defaults
CMD_FSMOUNTS *	_SLASH_dev_SLASH_pts _SLASH_proc _SLASH_sys _SLASH_dev_SLASH_shm _SLASH_cm_SLASH_shared _SLASH_home

...continues

Table E: Environment Variables For The `initialize` *And* `Finalize` *Scripts...continued*

Variable	Example Value
CMD_GATEWAY	10.141.255.254
CMD_HOSTNAME	node001
CMD_INSTALLMODE	AUTO
CMD_INTERFACE_eth0_IP **	10.141.0.1
CMD_INTERFACE_eth0_MTU **	1500
CMD_INTERFACE_eth0_NETMASK **	255.255.0.0
CMD_INTERFACE_eth0_TYPE **	physical
CMD_INTERFACES *	eth0 eth1 eth2 ipmi0
CMD_IP	10.141.0.1
CMD_MAC	00:00:00:00:00:01
CMD_PARTITION	base
CMD_PASSIVE_MASTER_IP	10.141.255.253
CMD_PDUS	
CMD_POWER_CONTROL	custom
CMD_RACK	rack01
CMD_RACK_HEIGHT	42
CMD_RACK_ROOM	serverroom
CMD_ROLES	sgeclient storage
CMD_SHARED_MASTER_IP	10.141.255.252
CMD_SOFTWAREIMAGE_PATH	/cm/images/default-image
CMD_SOFTWAREIMAGE	default-image
CMD_TAG	00000000a000
CMD_USERDEFINED1	var1
CMD_USERDEFINED2	var2

* The value for this variable is a string with spaces, not an array. Eg:
CMD_FSMOUNTS="_SLASH_dev_SLASH_pts _SLASH_proc _SLASH_sys ..."
** The name of this variable varies according to the interfaces available. So, `eth0` can be replaced by `eth1`, `eth2`, `ipmi0`, and so on.

E.4 Using Environment Variables Stored In Multiple Variables

Some data values, such as those related to interfaces (`CMD_INTERFACES_*`), mount points (`CMD_FSMOUNT__SLASH_*`) and exports (`CMD_FSEXPORT__SLASH_cm__SLASH_node-installer_*`) are stored in multiple variables. The code example below shows how they can be used:

Example

```
#!/bin/bash
```

```
for interface in $CMD_INTERFACES
do
  eval type=\$CMD_INTERFACE_${interface}_TYPE
  eval ip=\$CMD_INTERFACE_${interface}_IP
  eval mask=\$CMD_INTERFACE_${interface}_NETMASK

  echo "$interface type=$type"
  echo "$interface ip=$ip"
  echo "$interface netmask=$mask"
done
```

For remotely mounted devices, the name of the environment variables for mount entries have the following naming convention:

Description	Naming Convention
volume	CMD_FSMOUNT_<*x*>_DEVICE
mount point	CMD_FSMOUNT_<*x*>_MOUNTPOINT
filesystem type	CMD_FSMOUNT_<*x*>_FILESYSTEM
mount point options	CMD_FSMOUNT_<*x*>_OPTIONS

For the names, the entries <*x*> are substituted with the local mount point path, such as "/cm/shared", but with the "/" character replaced with the text "_SLASH_". So, for a local mount point path "/cm/shared", the name of the associated volume environment variable becomes CMD_FSMOUNT__SLASH_cm_SLASH_shared_DEVICE.

A similar naming convention is applicable to the names of the environment variables for the export entries:

Description	Naming Convention
exported system writable?	CMD_FSEXPORT_<*y*>_ALLOWWRITE
allowed hosts or networks	CMD_FSEXPORT_<*y*>_HOSTS
path on exporter	CMD_FSMOUNT_<*y*>_PATH

Here, the entry <*y*> is replaced by the file path to the exported filesystem on the exporting node. This is actually the same as the value of "CMD_FSMOUNT_<*y*>_PATH", but with the "/" character replaced with the text "_SLASH_".

The entries for the local mount values and the export values in the table in section E.3 are the default values for a newly installed cluster. If the administrator wishes to add more devices and mount entries, this is done by configuring fsexports on the head node, and fsmounts on the regular nodes, using cmgui or cmsh (section 4.11).

E.5 Storing A Configuration To A Filesystem

E.5.1 Storing With Initialize Scripts

The initialize script (section 6.4.5) runs after the install-mode type and execution have been determined (section 6.4.4), but before unloading specific drivers and before partitions are checked and filesystems mounted (section 6.4.6). Data output cannot therefore be written to a local drive. It can however be written by the script to the tmpfs, but data placed there is lost quite soon, namely during the `pivot_root` process that runs when the node-installer hands over control to the `init` process running from the local drive. However, if needed, the data can be placed on the local drive later by using the `finalize` script to copy it over from the tmpfs.

Due to this, and other reasons, a `finalize` script is easier to use for an administrator than an `initialize` script, and the use of the `finalize` script is therefore preferred.

E.5.2 Ways Of Writing A Finalize Script To Configure The Destination Nodes

Copying The File To The Image

For a `finalize` script (section 6.4.11), which runs just before switching from using the ramdrive to using the local hard drive, the local hard drive is mounted under `/localdisk`. Data can therefore be written to the local hard drive if needed, but is only persistent until a reboot, when it gets rewritten. For example, predetermined configuration files can be written from the NFS drive for a particular node, or they can be written from an image prepared earlier and now running on the node at this stage, overwriting a node-installer configuration:

Example

```
#!/bin/bash
cp /etc/myapp.conf.overwrite /localdisk/etc/myapp.conf
```

This technique is used in a `finalize` script example in section 4.16.4, except that an append operation is used instead of a copy operation, to overcome a network issue by modifying a network configuration file slightly.

Protecting The Configuration File Change With `excludelistupdate`

In the preceding example, the finalize script saves a file `/etc/myapp.conf` to the destination nodes.

To protect such a configuration file from erasure, its file path must be covered in the second sublist in the `excludelistupdate` list.

Copying A File To The Image—Decision Based On Detection

Detection within a `finalize` script is another useful technique. The `finalize` script example of section 4.16.4 does detection too, to decide if a configuration change is to be done on the node or not.

A useful variation of a `finalize` script with detection is a script selecting from a choice of possible configurations. A symlink is set to one of the possible configurations based on hardware detection or detection of an environment variable. The environment variable can be a node parameter or similar, from the table in section E.3. If it is necessary to overwrite different nodes with different configurations, then the previous `finalize` script example might become something like:

Example

```
#!/bin/bash
if [[ $CMD_HOSTNAME = node00[1-7] ]]
```

```
  then ln -s /etc/myapp.conf.first /localdisk/etc/myapp.conf
fi
if [[ $CMD_HOSTNAME = node01[5-8] ]]
  then ln -s /etc/myapp.conf.second /localdisk/etc/myapp.conf
fi
if [[ $CMD_HOSTNAME = node02[3-6] ]]
  then ln -s /etc/myapp.conf.third /localdisk/etc/myapp.conf
fi
```

In the preceding example, the configuration file in the image has several versions: `/etc/myapp.conf.<first|second|third>`. Nodes node001 to node007 are configured with the first version, nodes node015 to node018 with the second version, and nodes node023 to node026 with the third version. More versions are convenient to add to this kind of decision mechanism.

E.5.3 Restricting The Script To Nodes Or Node Categories

As mentioned in section 3.1.3, node settings can be adjusted within a category. So the configuration changes to `ifcfg-eth0` is best implemented per node by accessing and adjusting the `finalize` script per node if only a few nodes in the category are to be set up like this. If all the nodes in a category are to be set up like this, then the changes are best implemented in a `finalize` script accessed and adjusted at the category level. Accessing the scripts at the node and category levels is covered in section E.2.

People used to normal object inheritance behavior should be aware of the following when considering category level and node level finalize scripts:

With objects, a node item value overrules a category level value. On the other hand, finalize scripts, while treated in an analogous way to objects, cannot always inherit properties from each other in the precise way that true objects can. Thus, it is possible that a finalize script run at the node level may not have anything to do with what is changed by running it at the category level. However, to allow it to resemble the inheritance behavior of object properties a bit, the node-level finalize script, if it exists, is always run after the category-level script. This gives it the ability to “overrule” the category level.

F

Quickstart Installation Guide

This appendix describes a basic installation of DirectMon on a cluster as a step-by-step process. Following these steps allows cluster administrators to get a cluster up and running as quickly as possible without having to read the entire *Administrator Manual*. References to chapters and sections are provided where appropriate.

F.1 Installing The Head Node

The head node does not need to be connected to the regular nodes at this point, though it helps to have the wiring done beforehand so that how things are connected is known.

1. Set the local time in the BIOS of the head node.

2. Boot the head node from the DirectMon DVD.

3. Select `Install DirectMon` in the text boot menu, to bring up the GUI installation `Welcome` screen.

4. At the `Welcome` screen, click `Continue`. By default, this continues with a `Normal (recommended)` installation mode.

5. At the `License` screens:

   - At the `DDN Software License` screen, click the acceptance checkbox. Then click `Continue`.
   - At the Linux base distribution screen, click on the acceptance checkbox. Then click `Continue`.

6. At the `Kernel Modules` screen, click `Continue`.

7. At the `Hardware Information` screen, review the detected hardware. Go back to the `Kernel Modules` screen if additional kernel modules are required. Once all the relevant hardware (Ethernet interfaces, hard drive and DVD drive) is detected, click `Continue`.

8. At the `Nodes` screen:

   - Specify the number of racks and regular nodes

- Set the base name for the regular nodes. Accepting the default of `node` means nodes names are prefixed with `node`, for example: `node001`, `node002`...
- Set the number of digits to append to the base name. For example, accepting the default of `3` means nodes from `node001` to `node999` are possible names.
- Select the correct hardware manufacturer

Then click `Continue`.

9. At the `Network Topology` screen, choose a network layout. The default layout, `private internal network`, is the most commonly used. The rest of this appendix assumes the first layout was chosen. Click `Continue`

10. At the `Additional Network Configuration` screen, you can

    - add an InfiniBand and/or 10 Gig-E network, and
    - configure the use of IPMI/iLO BMCs on the nodes. Adding an IPMI/iLO network is needed to configure IPMI/iLO interfaces in a different IP subnet, and is recommended.

    When done, click `Continue`.

11. At the `Networks` screen, enter the network parameters for the head node for the interface facing the network named `externalnet`:

    If using DHCP on that interface, click on the `OK` button to accept the parameters for `IP Address`, `Netmask` and `Gateway` as suggested by the DHCP server on the external network.

    If not using DHCP on that interface, uncheck the `Use DHCP` checkbox, and put in static values instead. Then click the `OK` button.

    The network parameters for `externalnet` can then then be reviewed. These are the:

    - base address (also called the *network address*)
    - netmask
    - domain name
    - default gateway

    The network `externalnet` corresponds to the site network that the cluster resides in (e.g. a corporate or campus network). The IP address details are therefore the details of the head node for a type 1 externalnet network (figure 2.8). A domain name should be entered to suit the local requirements.

12. At the `Nameservers` screen, additional DNS search domains and additional external DNS name servers, if required, can be added or removed. For the default network topology (see item 9, page 622), if the external network has a DHCP lease with DNS configuration information, then nothing needs to be added to resolve external hosts. Click `Continue`

13. At the `Network Interfaces` screen:

- Review the IP addresses assigned to the network interfaces. All networks besides the `externalnet` network use private IP ranges by default and normally do not have to be changed.
- If necessary, modify the node interface properties. When IPMI/iLO interfaces reside in the same IP subnet, an IP Offset is set for the `ipmi0` interface.

Click `Continue`.

14. The `Subnet Managers` screen is displayed if an InfiniBand network was enabled. At this screen, select which nodes (if any) are to run the subnet manager for the InfiniBand network. Then click `Continue`.
15. At the `Installation sources` screen, select the DVD drive containing the DirectMon DVD, then click `Continue`.
16. At the `Workload Management` screen, select a workload manager and set the number of slots per node equal to the number of CPU cores per node. Click `Continue`.
17. At the `Disk Partitioning and Layouts` screen, select a drive on the head node. The installation will be done onto this drive, overwriting all its previous content.

    You may modify the disk layout for the head node by selecting a pre-defined layout.

    For hard drives that have less than about 500GB space, the XML file `master-one-big-partition.xml` is used by default:

Partition	Space	Mounted At	Filesystem Type
1	512M	/boot	ext2
2	16G	-	swap
3	rest	/	ext3

Default layout for up to 500GB: One big partition.

For hard drives that have about 500GB or more of space, the XML file `master-standard.xml` is used by default:

Partition	Space	Mounted At	Filesystem Type
1	512M	/boot	ext2
2	16G	-	swap
3	8G	/tmp	ext3
4	16G	/var	ext3
5	10G	/var/lib/mysql/ cmdaemon_mon	ext3
6	rest	/	ext3

Default layout for more than 500GB: Several partitions.

The layouts indicated by these tables may be fine-tuned by editing the XML partitioning definition during this stage. The "max" setting in the XML file corresponds to the "rest" entry in these tables, and means the rest of the drive space is used up for the associated partition, whatever the leftover space is.

There are also other layout templates available from a menu.

Click `Continue`, and then confirm that the data on the listed drive(s) may be erased by clicking `Yes`.

18. At the `Time Configuration` screen, select a time-zone and optionally add NTP time-servers. Then click `Continue`.

19. At the `Cluster Access` screen, some network restrictions can be set. By default there are no network-specific restrictions on access to the cluster (e.g. using `ssh` or `cmgui`). To accept the defaults, click `Continue`.

20. At the `Authentication` screen, enter a hostname for the head node. Also enter a password that is to be used for system administration twice. Then click `Continue`.

21. At the `Console` screen, a text or graphical console can be configured for the nodes in the cluster. Note that the Cluster Management GUI can still be used remotely even if the console of the head node is set to text mode. Then click `Continue`

22. At the `Summary` screen, review the network summary. Click the `Start` button to start the installation and click `Yes` to confirm that the data on the listed drive(s) may be erased.

23. Wait for the `Installation Progress` screen to complete. Then click `Reboot` and click `Yes` to confirm that the head node may be rebooted.

F.2 First Boot

1. Ensure that the head node boots from the first hard drive by removing the DVD or altering the boot-order in the BIOS configuration.

2. Once the machine is fully booted, log in as `root` with the password that was entered during installation.

3. Confirm that the machine is visible on the external network. Ensure that the second NIC (i.e. `eth1`) is physically connected to the external network.

4. Verify that the license parameters are correct:
   ```
   cmsh -c "main licenseinfo"
   ```
   If the license being used is a temporary license (see `End Time` value), a new license should be requested well before the temporary license expires. The procedure for requesting and installing a new license is described in section 4.1.

F.3 Booting Regular Nodes

1. Make sure the first NIC (i.e. `eth0`) on the head node is physically connected to the internal cluster network.

2. Configure the BIOS of regular nodes to boot from the network, then boot the regular nodes. No operating system is expected to be on the regular nodes already. If there is an operating system there already, then by default, it is overwritten by a default image provided by the head node during the next stages.

3. If everything goes well, the node-installer component starts on each regular node and a certificate request is sent to the head node.

   If a regular node does not make it to the node-installer stage, it is possible that additional kernel modules are needed. Section 6.8 contains more information on how to diagnose problems during the regular node booting process.

4. To identify the regular nodes (i.e. to assign a host name to each physical node), several options are available. Which option is most convenient depends mostly on the number of regular nodes and whether a (configured) managed Ethernet switch is present.

   Rather than identifying nodes based on their MAC address, it is often beneficial (especially in larger clusters) to identify nodes based on the Ethernet switch port that they are connected to. To allow nodes to be identified based on Ethernet switch ports, section 4.9 should be consulted.

   Any one of the following methods may be used to assign node identities when all nodes have been booted:

   a. **Identify each node on the node console**: To manually identify each node, the "`Manually select node`" option is selected for each node. The node is then identified manually by selecting a node-entry from the list, choosing the `Accept` option. This option is easiest when there are not many nodes. It requires being able to view the console of each node and keyboard entry to the console.

   b. **Identify nodes using** `cmgui`: The Node Identification Wizard (section 6.4.2) in `cmgui` automates the process of assigning identities so that nodes do not require manual identification on the console.

   c. **Identify nodes using** `cmsh`: In `cmsh` the `newnodes` command in `device` mode (section 6.4.2) can be used to assign identities to nodes from the command line. When called without parameters, the `newnodes` command can be used to verify that all nodes have booted into the node-installer and are all waiting to be assigned an identity.

      Example

      To verify that all regular nodes have booted into the node-installer:

```
[root@mycluster ~]# cmsh
[mycluster]% device newnodes
MAC                First appeared            Detected on switch port
-----------------  -------------------------  -----------------------
00:0C:29:D2:68:8D  05 Sep 2011 13:43:13 PDT  [no port detected]
00:0C:29:54:F5:94  05 Sep 2011 13:49:41 PDT  [no port detected]
..
[mycluster]% device newnodes | wc -l
MAC                First appeared      Detected on switch port
-----------------  -------------------- -----------------------
32
[mycluster]% exit
[root@mycluster ~]#
```

Example

Once all regular nodes have been booted in the proper order, we can use the order of their appearance on the network to assign node identities. To assign identities `node001` through `node032` to the first 32 nodes that were booted, the following commands may be used:

```
[root@mycluster ~]# cmsh
[mycluster]% device newnodes -s -n node001..node032
MAC                First appeared                Hostname
-----------------  ----------------------------- ---------
00:0C:29:D2:68:8D  Mon, 05 Sep 2011 13:43:13 PDT node001
00:0C:29:54:F5:94  Mon, 05 Sep 2011 13:49:41 PDT node002
..
[mycluster]% exit
[root@mycluster ~]#
```

5. Each regular node is now provisioned and eventually fully boots. In case of problems, section 6.8 should be consulted.

6. *Optional:* To configure power management, Chapter 5 should be consulted.

F.4 Running Cluster Management GUI

To run the Cluster Management GUI on the cluster from a workstation running X11:

1. From a Linux desktop PC, log in to the cluster with SSH X-forwarding:
   `ssh -X root@mycluster`

2. Start the Cluster Management GUI:
   `cmgui`

3. Click on the connect button (see figure 3.2 and enter the password that was configured during installation.)

4. *Optional:* For more information on how the Cluster Management GUI can be used to manage one or more clusters, consult section 3.4.

To run the Cluster Management GUI on a desktop PC:

1. Copy the appropriate package(s) for MS Windows, Linux, or Mac OS from `/cm/shared/apps/cmgui/dist` to the desktop PC:

   ```
   scp root@mycluster:/cm/shared/apps/cmgui/dist/* /tmp
   ```

   Note: On windows use e.g. WinSCP.

2. Install the package.
   On Windows: execute the installer and follow the steps.
   On Linux: extract using `tar -xvjf` *filename*.
   On Mac OS: execute the installer and follow the steps.

3. Start the cluster management GUI.
   On Windows: from the Start menu or by clicking the desktop icon.
   On Linux: change into the `cmgui` directory and execute:
   `./cmgui`
   On Mac OS: from the Start menu or by clicking the desktop icon.

4. Click on `Add a new cluster` and enter the following parameters:
   Host: Hostname or IP address of the cluster
   Password: Password entered during installation

5. Click on the connect button (see figure 3.2)

6. *Optional:* For more information on how the Cluster Management GUI can be used to manage one or more clusters, consult section 3.4.

Your cluster should now be ready for running compute jobs. For more information on managing the cluster, please consult the appropriate chapters in this manual.

Please consult the *User Manual* provided in:
`/cm/shared/docs/cm/user-manual.pdf`
for more information on the user environment and how to start jobs through the workload management system.

G

Workload Managers Quick Reference

G.1 Slurm

Slurm is a GPL-licensed workload management system and developed largely at Lawrence Livermore National Laboratory. The name was originally an acronym for Simple Linux Utility for Resource Management, but the acronym is deprecated because it no longer does justice to the advanced capabilities of Slurm.

The Slurm service and outputs are normally handled using the `cmgui` or `cmsh` front end tools for CMDaemon (section 9.4).

From the command line, direct Slurm commands that may sometimes come in useful include the following:

- `sacct`: used to report job or job step accounting information about active or completed jobs.
- `salloc`: used to allocate resources for a job in real time. Typically this is used to allocate resources and spawn a shell. The shell is then used to execute srun commands to launch parallel tasks.
- `sattach` used to attach standard input, output, and error plus signal capabilities to a currently running job or job step. One can attach to and detach from jobs multiple times.
- `sbatch`: used to submit a job script for later execution. The script typically contains one or more srun commands to launch parallel tasks.
- `sbcast`: used to transfer a file from local disk to local disk on the nodes allocated to a job. This can be used to effectively use diskless compute nodes or provide improved performance relative to a shared file system.
- `scancel`: used to cancel a pending or running job or job step. It can also be used to send an arbitrary signal to all processes associated with a running job or job step.
- `scontrol`: the administrative tool used to view and/or modify Slurm state. Note that many scontrol commands can only be executed as user root.

- `sinfo`: reports the state of partitions and nodes managed by Slurm. It has a wide variety of filtering, sorting, and formatting options.
- `smap`: reports state information for jobs, partitions, and nodes managed by Slurm, but graphically displays the information to reflect network topology.
- `squeue`: reports the state of jobs or job steps. It has a wide variety of filtering, sorting, and formatting options. By default, it reports the running jobs in priority order and then the pending jobs in priority order.
- `srun`: used to submit a job for execution or initiate job steps in real time. srun has a wide variety of options to specify resource requirements, including: minimum and maximum node count, processor count, specific nodes to use or not use, and specific node characteristics (so much memory, disk space, certain required features, etc.). A job can contain multiple job steps executing sequentially or in parallel on independent or shared nodes within the job's node allocation.
- `smap`: reports state information for jobs, partitions, and nodes managed by Slurm, but graphically displays the information to reflect network topology.
- `strigger`: used to set, get or view event triggers. Event triggers include things such as nodes going down or jobs approaching their time limit.
- `sview`: a graphical user interface to get and update state information for jobs, partitions, and nodes managed by Slurm.

Full documentation on Slurm is available online at: `https://computing.llnl.gov/linux/slurm/documentation.html`.

G.2 Sun Grid Engine

Sun Grid Engine (SGE) is a workload management system that was originally made available under an Open Source license by Sun Microsystems. It forked off into various versions in 2010 and its future is unclear at the time of writing, but it remains in widespread use. DirectMon™ uses version 6.2 update 5 patch 2 of Grid Scheduler, which is a bugfix-patched version of the last SGE release from Sun Microsystems, and is made available on sourceforge at `http://gridscheduler.sourceforge.net/`.

SGE services should be handled using CMDaemon, as explained in section 9.4. However SGE can break in obtuse ways when implementing changes, so the following notes are sometimes useful in getting a system going again:

- The `sge_qmaster` daemon on the head node can be started or stopped using `/etc/init.d/sgemaster.sqe1 start|stop`, or alternatively via `qconf -{s|k}m`.
- The `sge_execd` execution daemon running on each compute node accepts, manages, and returns the results of the jobs on

the compute nodes. The daemon can be started or stopped via `/etc/init.d/sgeexecd start|stop`, or alternatively deregistered from `qmaster` via `qconf -{s|k}s`.

- To see why a queue is in an error state, `qstat -explain E` shows the reason. Queues in an error state can be cleared with a `qmod -c <queue name>`.

SGE can be configured and managed generally with the command line utility `qconf`, which is what most administrators become familiar with. A GUI alternative, `qmon`, is also provided.

SGE commands are listed below. The details of these are in the `man` page of the command and the SGE documentation.

- `qalter`: modify existing batch jobs
- `qacct`: show usage information from accounting data
- `qconf`: configure SGE
- `qdel`: delete batch jobs
- `qhold`: place hold on batch jobs
- `qhost`: display compute node queues, states, jobs
- `qlogin`: start `login`-based interactive session with a node
- `qmake`: distributed, parallel make utility
- `qmod`: suspend/enable queues and jobs
- `qmon`: configure SGE with an X11 GUI interface
- `qping`: check `sge_qmaster` and `sge_execd` status
- `qquota`: list resource quotas
- `qresub`: create new jobs by copying existing jobs
- `qrdel`: cancel advance reservations
- `qrls`: release batch jobs from a held state
- `qrsh`: start `rsh`-based interactive session with node
- `qrstat`: show status of advance reservations
- `qrsub`: submit advanced reservation
- `qselect`: select queues based on argument values
- `qsh`: start `sh` interactive session with a node
- `qstat`: show status of batch jobs and queues
- `qsub`: submit new jobs (related: qalter, qresub)
- `qtcsh`: start `csh`-based interactive session with a node

G.3 Torque

The following commands can be used to manage Torque:

Torque resource manager commands:

- `qalter`: alter batch job
- `qdel`: delete batch job
- `qhold`: hold batch jobs
- `qrls`: release hold on batch jobs
- `qstat`: show status of batch jobs
- `qsub`: submit job
- `qmgr`: batch policies and configurations manager
- `qenable`: enable input to a destination
- `qdisable`: disable input to a destination
- `tracejob`: trace job actions and states

These direct commands are capable of modifying properties that CMDaemon does not manage, as well as properties that CMDaemon does manage. CMDaemon however overwrites the properties that are managed by CMDaemon with values recalled from the CMDaemon database, typically within a few minutes, if the values are changed directly via the direct commands.

Further information on these and other commands is available in the appropriate man pages and on-line documentation at `http://www.adaptivecomputing.com/resources/docs/`. The Torque administrator manual is also available from there.

G.4 PBS Pro

The following commands can be used in PBS Pro to view queues:

```
qstat             query queue status
qstat -a          alternate form
qstat -r          show only running jobs
qstat -q          show available queues
qstat -rn         only running jobs, w/ list of allocated nodes
qstat -i          only idle jobs
qstat -u username show jobs for named user
```

Other useful commands are:

```
tracejob id       show what happened today to job id
tracejob -n d id  search last d days

qmgr              administrator interface to batch system

qterm             terminates queues (but cm starts pbs_server again)

pbsnodes -a       list available worker nodes in queue
```

The commands of PBS Pro are documented in the *PBS Professional 11.0 Reference Guide*. There is further extensive documentation for PBS Pro administrators in the *PBS Professional 11.0 Administrator's Guide*. Both are available at the PBS Works website at `http://www.pbsworks.com/SupportDocuments.aspx`.

G.5 openlava

openlava is a free, GNU GPLv2 licensed workload management system based on Platform LSF. The openlava source is available on github at `https://github.com/openlava/openlava/`.

openlava is a fork of Platform Lava, released under GNU GPLv2. This in turn is based on a stripped-down version of Platform LSF 4.2.

openlava services and outputs are normally handled using the `cmgui` or `cmsh` front end tools for CMDaemon (section 9.4).

From the command line, direct openlava commands that may sometimes come in useful include the following:

- `badmin`: the administrative tool used to control and monitor openlava. It provides the following commands: `ckconfig`, `reconfig`, `mbdrestart`, `qopen`, `qclose`, `qact`, `qinact`, `hopen`, `hclose`, `hrestart`, `hshutdown`, `hstartup`, `hhist`, `mbdhist`, `hist`, `sbdtime`, `mbdtime` and `mbddebug`.
- `bjobs`: displays information about openlava jobs.
- `brun`: used to force a pending openlava job to run immediately on specified hosts.
- `bbot`: used to change the queue position of a pending openlava job, or a pending job array element, to affect the order in which jobs are considered for dispatch.
- `bchkpnt`: used to checkpoint one or more checkpointable openlava jobs.
- `bhosts`: displays information about static and dynamic resources of hosts.
- `bkill`: sends signals to kill, suspend, or resume unfinished openlava jobs.
- `bmgroup`: displays information about host openlava groups.
- `bmig`: used to migrate checkpointable or rerunnable openlava jobs.
- `bmod`: used to modify job submission options of an openlava job.
- `bparams`: displays information about configurable system parameters in lsb.params.
- `bpeek`: displays the stdout and stderr output of an unfinished openlava job.
- `bqueues`: displays information about openlava queues.
- `brequeue`: used to kill and requeue an openlava job.

- `brestart`: used to restart checkpointed openlava jobs.
- `bresume`: used to resume one or more suspended openlava jobs.
- `bstop`: used to suspend unfinished openlava jobs.
- `bsub`: used to submit a batch job to openlava.
- `bswitch`: used to switch unfinished openlava jobs from one queue to another.
- `btop`: used to move a pending openlava job relative to the first job in the queue.
- `bugroup`: displays information about user groups.
- `busers`: displays information about users and user groups.
- `lshosts`: displays information about resources defined on hosts.
- `lsload`: displays the current state of the host about its ability to run batch jobs and remote tasks.
- `lsinfo`: displays information about load indices.
- `lsid`: displays the current openlava version number, the cluster name, and the active openlava master node name.
- `lsmon`: displays load information for openlava hosts.
- `lsplace`: displays hosts available to execute openlava jobs.
- `lsrcp`: used to copy files remotely using openlava.

Full documentation on openlava is available online at: `http://openlava.org/documentation/documentation.html`.

H

Metrics, Health Checks, And Actions

This appendix describes the metrics (section H.1), health checks (section H.2), and actions (section H.3), along with their parameters, in a newly-installed cluster. Metrics, health checks, and actions can each be standalone scripts, or they can be built-ins. Standalone scripts can be those supplied with the system, or they can be custom scripts built by the administrator. Scripts can use environment variables (as described in Appendix I.5). On success scripts must exit with a status of 0, as is the normal practice.

H.1 Metrics And Their Parameters

H.1.1 Metrics

Table H.1.1: List Of Metrics

Metric	Description
`AlertLevel`	Indicates the healthiness of a device based on severity of events (section 11.2.7). The lower it is, the better.
`Average`	Cluster-wide average of metric of choice
`AvgExpFactor`	Average Expansion Factor. This is by what factor, on average, jobs took longer to run than expected. The expectation is according to heuristics based on duration in past and current job queues, as well as node availability.

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
`AvgJobDuration`	Average Job Duration of current jobs
`BufferMemory`	System memory used for buffering
`BytesRecv`*	Bytes/s received
`BytesSent`*	Bytes/s sent
`CacheMemory`	System memory used for caching
`CMDActiveSessions`	Managed active sessions count
`CMDCycleTime`	Time spent by head to process picked up data
`CMDMemUsed`	Resident memory used by CMDaemon
`CMDState`	State in which CMDaemon is running (head: 0, regular: 1, failover:2)
`CMDStoreQueryTime`	Time spent by head node to store monitoring data to database
`CMDSystime`*	Percent of time spent by CMDaemon in system mode
`CMDUsertime`*	Percent of time spent by CMDaemon in user mode
`CompletedJobs`	Jobs completed
`CPUCoresAvailable`	Cluster-wide number of CPU cores
`CPUIdle`*	Percent of node-wide core time spent in idle mode. Example: The CPUIdle time measurement on a 1-core machine over 1 second can be up to 1 second (no time in this mode = 0s, all of its time in this mode = 1s). The CPUIdle time measurement on a 2-core machine over 1 second can be up to 2 seconds at the most. The derived CPUIdle metric for a 1-core system can be up to 100% at the most. The derived CPUIdle metric for a 2-core system can be up to 200% at the most.

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
CPUIrq*	Percent of node-wide core time spent in servicing interrupts
CPUNice*	Percent of node-wide core time spent in nice user mode
CPUSoftIrq*	Percent of node-wide core time spent in servicing soft interrupts
CPUSystem*	Percent of node-wide core time spent in system mode
CPUUser*	Percent of node-wide core time spent in user mode
CPUWait*	Percent of node-wide core time spent in waiting for I/O to complete
CtxtSwitches*	Context switches/s
Current_1**	First current seen by BMC sensor, in amps (file: sample_ipmi)
Current_2**	Second current seen by BMC sensor, in amps (file: sample_ipmi)
DevicesUp	Number of devices in status UP
DropRecv*	Packets/s received and dropped
DropSent*	Packets/s sent and dropped
EC2SpotPrice	Amazon EC2 price for spot instances
EccDBitGPU**	Total number of double bit ECC errors (file: sample_gpu)
EccSBitGPU**	Total number of single bit ECC errors (file: sample_gpu)
ErrorsRecv*	Packets/s received with error
ErrorsSent*	Packets/s sent with error
EstimatedDelay	Estimated delay time to execute jobs
Exhaust_Temp	Exhaust temperature measured by BMC, in Celsius (file: sample_ipmi)
FailedJobs	Failed jobs
Fan<*number*>_RPM	RPM of Fan<*number*> as seen by BMC (file: sample_ipmi)
FanSpeedPercGPU**	Percent of full fan speed for GPU <*number*> (file: sample_gpu)

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
`Forks`*	Forks/s
`FrameErrors`*	Packet framing errors/s
`FreeSpace`	Free space for non-root. Takes mount point as a parameter
`GPUAvailable`	Cluster-wide number of GPUs
`gpu`**	GPU metrics collection (file: `sample_gpu`)
`GpuUtilGPU`**	Percent Utilization of GPU <*number*> (file: `sample_gpu`)
`ilo`**	iLO metrics collection (file: `sample_ilo`)
`Inlet_Temp`**	Inlet Temperature, in Celsius (file: `sample_ipmi`)
`IOInProgress`	I/O operations in progress
`IOTime`*	I/O operations time in milliseconds/s
`ipForwDatagrams`*	Input IP datagrams/s to be forwarded
`ipFragCreates`*	IP datagram fragments/s generated
`ipFragFails`*	IP datagrams/s which needed to be fragmented but could not
`ipFragOKs`*	IP datagrams/s successfully fragmented
`ipInAddrErrors`*	Input datagrams/s discarded because the IP address in their header was not a valid address
`ipInDelivers`*	Input IP datagrams/s successfully delivered
`ipInDiscards`*	Input IP datagrams/s discarded
`ipInHdrErrors`*	Input IP datagrams/s discarded due to errors in their IP headers
`ipInReceives`*	Input IP datagrams/s, including ones with errors, received from all interfaces
`ipInUnknownProtos`*	Input IP datagrams/s received but discarded due to an unknown or unsupported protocol
`ipmi`**	IPMI metrics collection (file: `sample_ipmi`)
`ipOutDiscards`*	Output IP datagrams/s discarded
`ipOutNoRoutes`*	Output IP datagrams/s discarded because no route could be found
`ipOutRequests`*	Output IP datagrams/s supplied to IP in requests for transmission

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
`ipReasmOKs`*	IP datagrams/s successfully re-assembled
`ipReasmReqds`*	IP fragments/s received needing re-assembly
`LoadFifteen`	Load average on 15 minutes
`LoadFive`	Load average on 5 minutes
`LoadOne`	Load average on 1 minute
`lustre`**	Lustre filesystem metrics collection (file: `sample_lustre`)
`MajorPageFaults`*	Page faults/s that require I/O
`Max`	Cluster-wide maximum for metric specified as parameter in metric configuration
`MemoryFreeGPU`**	Memory free for GPU <*number*> (file: `sample_gpu`)
`MemoryFree`	Free system memory
`MemoryUsed`	Used system memory
`MemUsedGPU`**	Memory used by GPU <*number*> (file: `sample_gpu`)
`MemUtilGPU`**	Percent of memory utilization by GPU <*number*> (file: `sample_gpu`)
`MergedReads`*	Merged reads/s
`MergedWrites`*	Merged writes/s
`mic`	MIC metrics collection (file: `sample_mic`)
`MICsUP`	Number of MICs in status UP
`Min`	Cluster-wide minimum for metric specified as parameter in metric configuration
`NetworkBytesRecv`	Cluster-wide number of bytes received on all networks
`NetworkBytesSent`	Cluster-wide number of bytes transmitted on all networks
`NetworkUtilization`	Network utilization percentage
`NodesInQueue`	Number of nodes in the queue
`NodesUp`	Number of nodes in status UP
`OccupationRate`	Cluster occupation rate—a normalized cluster load percentage. 100% means all cores on all nodes are fully loaded. The calculation is done as follows: `LoadOne` on each node is mapped to a value, calibrated so that `LoadOne`=1 corresponds to 100% per node. The maximum allowed for a node in the mapping is 100%. The average of these mappings taken over all nodes is the `OccupationRate`.

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
PacketsRecv*	Packets/s received
PacketsSent*	Packets/s sent
PageFaults*	Page faults/s
PDUBankLoad	Total PDU bank load, in amps
PDULoad	Total PDU phase load, in amps
PDUUptime*	PDU uptime per second. I.e. ideally=1, but in practice has jitter effects.
PhaseLoad	Cluster-wide phase load current, in amps (depends on PhaseLoadMetrics directive (page 595).
PowerDrawGPU	Power used by GPU <*number*>, in watts (file: sample_gpu)
price	Price per hour for this cloud node (approximation)
ProcessCount	Total number of all processes running in the OS
Pwr_Consumption	Power consumed by BMC, in watts (file: sample_ipmi)
QueuedJobs	Queued jobs
RackSensorHumidity	Rack sensor humidity
RackSensorTemp	Rack sensor Temperature
Range*	Cluster-wide range of metric of choice (difference between min and max)
Reads*	Reads/s completed successfully
ReadTime*	Read time in milliseconds/s
responsiveness**	Device I/O requests metrics collection (file: sample_responsiveness)
RunningJobs	Running jobs
RunningProcesses	Running processes
sdt**	SuperMicro metrics collection (file: sample_sdt)
SectorsRead*	Sectors/s read successfully/s
SectorsWritten*	Sectors/s written successfully
SensorFanSpeed	System or CPU fan speed detected
SensorTemp	Temperature (system and CPU) detected

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
`SensorVoltage`	Motherboard voltage detected
`SMARTHDATemp`	Temperature of a Hard Disk Assembly
`SMARTReallocSecCnt`	SMART reallocated sectors count
`SMARTSeekErrRate`	SMART seek errors/s
`SMARTSeekTimePerf`	SMART average seek time
`SMARTSoftReadErrRate`	SMART software read errors/s
`SSLCacheHitRatio`	SSL connections that are served by the SSL cache (%). If this is 100, then all connections are cached, and the cluster is serving everything from cache
`SSLCacheUsage`	SSL cache used (%). If this is 100, then there are too many SSL connections.
`Sum`	Cluster-wide sum of metric of choice
`SwapFree`	Free swap space
`SwapUsed`	Used swap space
`SwitchBroadcastPackets`*	Total number of good packets received and directed to the broadcast address/s
`SwitchCollisions`*	Collisions/s on this network segment
`SwitchCPUUsage`	Switch CPU utilization estimation (%)
`SwitchDelayDiscardFrames`*	Frames discarded/s due to excessive transit delay through the bridge
`SwitchFilterDiscardFrames`*	Valid frames received/s but discarded by the forwarding process
`SwitchMTUDiscardFrames`*	Number of frames discarded/s due to an excessive size
`SwitchMulticastPackets`*	Total number of good packets/s received and directed to a multicast address
`SwitchOverSizedPackets`*	Well-received packets/s longer than 1518 octets
`SwitchTemperature`	Switch temperature, in Celsius (for Force10 S-series)
`SwitchUnderSizedPackets`*	Packets/s received which are less than 64 octets long
`SwitchUptime`*	Switch uptime per second. Ie, ideally=1, but in practice has jitter effects

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
`tcpCurrEstab`	TCP connections that are either ESTABLISHED or CLOSE-WAIT
`tcpInErrs`*	Input IP segments/s received in error
`tcpRetransSegs`*	Total number of IP segments/s retransmitted
`TempGPU`**	Temperature of GPU <*number*>, in Celsius (file: `sample_gpu`)
`testcollection`**	An example of a metric collection script (file: `testmetriccollection`)
`testmetric`**	An example of a metric script (file: `testmetric`)
`TotalCPUIdle`	Cluster-wide core usage in idle tasks (sum of all CPUIdle metric percentages)
`TotalCPUSystem`	Cluster-wide core usage in system mode (sum of all CPUSystem metric percentages)
`TotalCPUUser`	Cluster-wide core usage in user mode (sum of all CPUUser metric percentages)
`TotalMemoryUsed`	Cluster-wide total of memory used
`TotalNodes`	Cluster-wide total number of nodes
`TotalSwapUsed`	Cluster-wide total swap used
`udpInDatagrams`*	Input UDP datagrams/s delivered to UDP users
`udpInErrors`*	Input UDP datagrams/s received that could not be delivered/s for other reasons (no port excl.)
`udpNoPorts`*	Received UDP datagrams/s for which there was no application at the destination port
`Uptime`*	System uptime per second. Ie, ideally=1, but in practice has jitter effects
`UsedSpace`	Total used space by a mount point
`Voltage_1`**	First voltage seen by BMC sensor, in Volts (file: `sample_ipmi`)
`Voltage_2`**	Second voltage seen by BMC sensor, in Volts (file: `sample_ipmi`)
`Writes`*	Writes/s completed successfully
`WriteTime`*	Write time in milliseconds/s

* Cumulative metric. I.e. the metric is derived from cumulative raw

...continues

Table H.1.1: List Of Metrics...continued

Name	Description

	measurements taken at two different times, according to: $metric_{time_2} = \frac{measurement_2 - measurement_1}{time_2 - time_1}$
	** Standalone scripts, not built-ins.
	If sampling from a head node, the script is in directory:
	`/cm/local/apps/cmd/scripts/metrics/`
	If sampling from a regular node, the script is in directory:
	`/cm/images/default-image/cm/local/apps/cmd/scripts/metrics/`

H.1.2 Parameters For Metrics

Metrics have the parameters indicated by the left column in the following example:

Example

```
[myheadnode->monitoring->metrics]% show cpuuser
Parameter               Value
----------------------  ------------------------------------------------
Class of metric         cpu
Command                 <built-in>
Cumulative              yes
Description             Percent of node-wide core time spent in user mode
Disabled                no
Extended environment    no
Measurement Unit        %
Name                    CPUUser
Only when idle          no
Parameter permissions   disallowed
Retrieval method        cmdaemon
Sampling method         samplingonnode
State flapping count    7
Timeout                 5
Valid for               node,headnode
maximum                 <range not set>
minimum                 <range not set>
[myheadnode->monitoring->metrics]%
```

The meanings of the parameters are:

`Class of metric`: A choice assigned to a metric depending on its type. The choices and what they are related to are listed below:

- `Misc` (default): miscellaneous class of metrics, used if none of the other classes are appropriate, or if none of the other classes are chosen
- `CPU`: CPU activity
- `GPU`: GPU activity
- `Disk`: Disk activity
- `Memory`: Memory activity
- `Network`: Network activity
- `Environmental`: sensor measurements of the physical environment

- `Operating System`: operating system activity
- `Internal`: DirectMon utilities
- `Workload`: workload management
- `Cluster`: clusterwide measurements
- `Prototype`: metric collections class

`Command`: For a standalone metric script, it is the full path. For a built-in, the value cannot be set, and the command is simply the name of the metric.

`Cumulative`: If set to `yes`, then the metric is cumulative. This actually means that the raw measurement instance used to calculate the metric is cumulative, like, for example, the bytes-received counter for an Ethernet interface. If set to `no` (default), then the raw value is not cumulative (for example, temperature). For a cumulative metric, the metric value is the difference in raw measurement values, divided by the time period over which they are sampled. Thus:

- The bytes-received raw measurements, which accumulate as the packets are received, and are in bytes, have a corresponding metric `BytesRecv` with a value in bytes/second.
- The system uptime raw measurements, which accumulate at the rate of 1 second per second, and are in seconds, have a corresponding metric (`Uptime`) with no units. Ideally, the metric has a value of 1, but in practice the measured value varies a little due to jitter.

`Description`: Description of the raw measurement used by the metric. Empty by default.

`Disabled`: If set to `no` (default) then the script runs.

`Extended environment`: If set to `yes`, more information about the device is made part of the environment to the script. The default is `no`.

`Metric Unit`: A unit for the metric. For example B/s (bytes/second) for `BytesRecv` metric, or unitless for the `Uptime` metric. A percent is indicated with %.

`Name`: The name given to the metric.

`Only when idle`: If `Only when idle` is set to yes, then the metric script only runs when the node is idling below an idle threshold value. It is set to `no` by default. Setting this to `yes` for resource-hungry metrics burdens the node less. The idle value is the value of `loadone`. The idle threshold is set by the CMDaemon directive `IdleThreshold`, and is set by default to `1.0`.

`Parameter permissions`: Decides if a parameter instance type or device can be passed to the metric script. There are three possible parameter permissions:

- `disallowed`: parameters are not used

- `required`: parameters are mandatory
- `optional` (default): parameters are optional

If parameter permissions are required, then a metric parameter name for a device or instance type is appended to the metric name with a ":". For example, the metric name `BytesRecv` has the device interface name `eth0` appended to it as follows:

Example

```
BytesRecv:eth0
```

A list of example metric parameter device or instance names can be viewed, for example, for the head node (`headnode`), using `cmsh` as follows:

```
[ddnmon61 ~]# cmsh -c "monitoring setup use headnode; metricconf;\
 list -d : -f metric:22,metricparam:30 | grep -v ': *:'"
```

The truncated output of this one-liner looks like:

```
metric                :metricparam                    :
----------------------:-------------------------------:
AlertLevel            :max                            :
AlertLevel            :sum                            :
AvgJobDuration        :default                        :
AvgJobDuration        :longq                          :
AvgJobDuration        :shortq                         :
BytesRecv             :eth0                           :
BytesRecv             :eth1                           :
...
```

`Retrieval method`:

- `cmdaemon` (default): Metrics retrieved internally using CMDaemon
- `snmp`: Metrics retrieved internally using SNMP

`Sampling method`:

- `samplingonmaster`: The head node samples the metric on behalf of a device. For example, the head node may do this for a PDU because the PDU does not have the capability to run the cluster management daemon at present, and so cannot itself pass on data values directly when `cmsh` or `cmgui` need them.
- `samplingonnode` (default): The non-head node samples the metric itself.

`State flapping count`: How many times the metric value must cross a threshold within the last 12 samples before it is decided that it is in a flapping state. Default value is 7.

`Timeout`: After how many seconds the command will give up retrying. Default value is 5 seconds.

`Valid for`: Which device category the metric can be used with. The choices being:

- `Node` (Default)
- `Master Node` (Also a default)
- `Power Distribution Unit`
- `Myrinet Switch`
- `Ethernet Switch`
- `IB Switch`
- `Rack Switch`
- `Generic Switch`
- `Chassis`
- `GPU Unit`

`Maximum`: the default minimum value the y-axis maximum will take in graphs plotted in `cmgui`.[1]

`Minimum`: the default maximum value the y-axis minimum will take in graphs plotted in `cmgui`.[1]

[1]To clarify the concept, if `maximum`=3, `minimum`=0, then a data-point with a y-value of 2 is plotted on a graph with the y-axis spanning from 0 to 3. However, if the data-point has a y-value of 4 instead, then it means the default y-axis maximum of 3 is resized to 4, and the y-axis will now span from 0 to 4.

H.2 Health Checks And Their Parameters

H.2.1 Health Checks

Table H.2.1: List Of Health Checks

Name	**Query** (response is PASS/FAIL/UNKNOWN)
DeviceIsUp*	Is the device up, closed or installing? Determined using TCP SYN pings to port 2 for nodes, port 7 for switches and other devices.
ManagedServicesOk*	Are CMDaemon-monitored services all OK? If the response is FAIL, then at least one of the services being monitored is failing. After correcting the problem with the service, a reset of the service is normally carried out (section 4.12, page 139).
chrootprocess	Are there daemon processes running using chroot in software images? (here: yes = FAIL). On failure, kill cron daemon processes running in the software images.
cmsh	Is cmsh available?
diskspace	Is there less disk space available to non-root users than any of the space parameters specified? *The space parameters can be specified as MB, GB, TB, or as percentages with %. The default severity of notices from this check is 10, when one space parameter is used. For more than one space parameter, the severity decreases by 10 for each space parameter, sequentially, down to 10 for the last space parameter. There must be at least one space parameter. An optional non-space parameter, the filesystem mount point parameter, can be specified after the last space parameter to track filesystem space, instead of disk space. A metric-based alternative to tracking filesystem space changes is to use the built-in metric* freespace *(page 638) instead.*

...continued

Table H.2.1: List Of Health Checks...continued

Name	**Query** (response is `PASS/FAIL/UNKNOWN`)
	Examples: • `diskspace 10%` less than 10% space = FAIL, severity 10 • `diskspace 10% 20% 30%` less than 30% space = FAIL, with severity levels as indicated: **space left** / **severity**: `10%` 30; `20%` 20; `30%` 10 • `diskspace 10GB 20GB` less than 20GB space = FAIL, severity 10 less than 10GB space = FAIL, severity 20 • `diskspace 10% 20% /var` For the file system `/var`: less than 20% space = FAIL, severity 10 less than 10% space = FAIL, severity 20

space left	**severity**
`10%`	30
`20%`	20
`30%`	10

...continued

Table H.2.1: List Of Health Checks...continued

Name	**Query** (response is PASS/FAIL/UNKNOWN)
dmesg	Is dmesg output ok? *Regexes to parse the output can be constructed in the configuration file at* */cm/local/apps/cmd/scripts/ healthchecks/configfiles/dmesg.py*
exports	Are all filesystems as defined by the cluster management system exported?
failedprejob	Are there failed prejob health checks? Here: yes = FAIL. By default, the job ID is saved under /cm/ shared/apps/<*scheduler*>/var/cm/: • On FAIL, in failedprejobs. • On PASS, in allprejobs The maximum number of IDs stored (by default, 1000), or the maximum period for which the IDs are stored (by default, 30 days). Both maxima can be set with the failedprejob health check script.
failover	Is all well with the failover system?
gpuhealth_quick	Is all well with the GPUs according to a quick check?
hpraid	Are the HP Smart Array controllers OK?
ib	Is the InfiniBand Host Channel Adapter working properly? *A configuration file for this health check is at* */cm/local/apps/cmd/scripts/ healthchecks/configfiles/ib.py*
interfaces	Are the interfaces all up and OK?
ipmihealth	Is the BMC (IPMI or iLO) health OK? Uses the script sample_ipmi.
ldap	Can the ID of the user be looked up with LDAP?
lustre	Is the Lustre filesystem running OK?
megaraid	Are the MegaRAID controllers OK? *The proprietary MegaCli software from LSI (http://www.lsi.com) is needed for this health check.*

...continued

Table H.2.1: List Of Health Checks...continued

Name	**Query** (response is `PASS/FAIL/UNKNOWN`)
`mounts`	Are all mounts defined in the fstab OK?
`mysql`	Is the status and configuration of MySQL correct?
`hardware-profile`	Is the specified node's hardware configuration during health check use unchanged? Uses the script `node-hardware-profile`. *The options to this script are described using the "-h" help option. Before this script is used for health checks, the specified hardware profile is usually first saved with the* `-s` *option. Eg: "*`node-hardware-profile -n node001 -s hardwarenode001`*"*
`ntp`	Is NTP synchronization happening?
`oomkiller`	Has the oomkiller system process run? Yes=FAIL. The configuration file `/cm/local/apps/cmd/scripts/healthchecks/configfiles/oomkiller.conf` for the oomkiller healthcheck can be configured to reset the response to PASS after one FAIL is logged, until the next oomkiller system process runs. The processes killed by the oomkiller system process are logged in `/var/spool/cmd/save-oomkilleraction`. *A consideration of the causes and consequences of the killed processes is strongly recommended. A reset of the node is generally recommended.*

...continued

Table H.2.1: List Of Health Checks...continued

Name	**Query** (response is `PASS/FAIL/UNKNOWN`)
`portchecker`	Is the specified port on the specified host open for TCP (default) or UDP connections?
`rogueprocess`	Are the processes that are running legitimate (ie, not 'rogue')? Illegitimate processes are processes that should not be running on the node. An illegitimate process is, by default: • not part of the workload manager service or its jobs • not a root- or system-owned process • in the state Z, T, W, or X. States are described in the `ps` man pages in the section on "PROCESS STATE CODES" Rogue process criteria can be configured in the file `/cm/local/apps/cmd/scripts/healthchecks/configfiles/rogueprocess.py`. To implement a changed criteria configuration, the software image used by systems on which the health check is run should be updated (section 6.6). For example, using: `cmsh -c "device; imageupdate -c default -w"` for the `default` category of nodes.
`schedulers`	Are the queue instances of all schedulers on a node healthy ?
`smart`	Is the SMART response healthy? The severities can be configured in the file `/cm/local/apps/cmd/scripts/healthchecks/configfiles/smart.conf`.
`ssh2node`	Is passwordless ssh login from head to node working?
`swraid`	Are the software RAID arrays healthy?
`testhealthcheck`	*A health check script example for creating scripts, or setting a mix of PASS/FAIL/UNKNOWN responses. The source includes examples of environment variables that can be used, as well as configuration suggestions.*

...continued

Table H.2.1: List Of Health Checks...continued

Name	**Query** (response is PASS/FAIL/UNKNOWN)

* built-ins, not standalone scripts.
If sampling from a head node, a standalone script is in directory:
`/cm/local/apps/cmd/scripts/healthchecks/`
If sampling from a regular node, a standalone script is in directory:
`/cm/images/default-image/cm/local/apps/cmd/scripts/healthchecks/`

H.2.2 Parameters For Health Checks

Health checks have the parameters indicated by the left column in the example below:

Example

```
[myheadnode->monitoring->healthchecks]% show cmsh
Parameter                Value
------------------------ -----------------------------------------------
Class of healthcheck     internal
Command                  /cm/local/apps/cmd/scripts/healthchecks/cmsh
Description              Checks whether cmsh is available, i.e. can we +
Disabled                 no
Extended environment     no
Name                     cmsh
Only when idle           no
Parameter permissions    optional
Sampling method          samplingonnode
State flapping count     7
Timeout                  10
Valid for                node,master,pdu,ethernet,myrinet,ib,racksensor+
```

The parameters have the same meaning as for metrics, with the following exceptions due to inapplicability:

Parameter	**Reason For Inapplicability**
`class: prototype`	only applies to metric collections
`cumulative`	only sensible for numeric values
`measurementunit`	only applies to numeric values
`retrievalmethod`	all health checks use CMDaemon internally for retrieval
`maximum`	only applies to numeric values
`minimum`	only applies to numeric values

The remaining parameters have meanings that can be looked up in section H.1.2.

H.3 Actions And Their Parameters

H.3.1 Actions

Table H.3.1: List Of Actions

Name	Description
`Drain node`	Allows no new processes on a compute node from the workload manager. This means that already running jobs are permitted to complete. Usage Tip: Plan for undrain from another node becoming active
`killprocess`*	Kills processes listed in STDIN with KILL (-9) signal. Format: `killprocess` <*PID1*[,<*PID2*>,...]>
`Power off`	Powers off, hard
`Power on`	Powers on, hard
`Power reset`	Power reset, hard
`Reboot`	Reboot via the system, trying to shut everything down cleanly, and then start up again
`SendEmail`**	Sends mail using the mailserver that was set up during server configuration. Format: `sendemail [`*`somebody@example.com`*`]`. Default destination is `root@localhost`. The e-mail address that it is sent to is specified by a parameter in the monitoring configuration that calls this action.
`Shutdown`	Power off via system, trying to shut everything down cleanly
`Restart service`**	Restart service. The service is specified by a parameter in the monitoring configuration that calls this action.
`Start service`**	Start service. The service is specified by a parameter in the monitoring configuration that calls this action.
`Stop service`**	Stop service. The service is specified by a parameter in the monitoring configuration that calls this action

...continued

Table H.3.1: List Of Actions...continued

Name	Description
`test action`*	An action script example for users who would like to create their own scripts. The source has helpful remarks about the environment variables that can be used as well as tips on configuring it generally
`Undrain node`	Allow processes to run on the node from the workload manager

* standalone scripts, not built-ins.
If running from a head node, the script is in directory:
`/cm/local/apps/cmd/scripts/actions/`
If running from a regular node, the script is in directory:
`/cm/images/default-image/cm/local/apps/cmd/scripts/actions/`

** the text
"is specified by a parameter in the monitoring configuration that calls this action"
means:
for `cmsh`:
-for metrics: like in `append` example for thresholds on page 447
-for health checks: like in `append` example for health configuration on page 451
for `cmgui`:
-for metrics: as illustrated by figure 11.25
-for health checks: as illustrated by figure 11.29

H.3.2 Parameters For Actions

Actions have the parameters indicated by the left column in the example below:

Example

```
[myheadnode->monitoring->actions]% show drainnode
Parameter          Value
------------------ ------------------------------------------------
Command            <built-in>
Description        Remove a node from further use by the scheduler+
Name               Drain node
Run on             headnode
Timeout            5
isCustom           no
```

The meanings of these parameters are:

`Command`: For a standalone metric script, it is the full path. For a built-in, the value cannot be set, and the command is simply the name of the metric.

`Description`: Description of the metric. Empty by default.

`Name`: The name given to the metric.

`Run on`: The node it will run on. For standalone actions it is usually a choice of head node, or the non-head node. For non-head nodes the action will run from the node that triggered it, if the node has sufficient permission to do that.

`Timeout`: After how many seconds the command will give up retrying.
Default value is 5 seconds.

`isCustom`: Is this a standalone script?

I Metric Collections

This appendix gives details on metric collections.

In section 11.4.4, metric collections are introduced, and how to add a metric collections script with `cmgui` is described.

This appendix covers how to add a metric collections script with `cmsh`. It also describes the output specification of a metric collections script, along with example outputs, so that a metric collections script can be made by the administrator.

I.1 Metric Collections Added Using `cmsh`

A metric collections script, `responsiveness`, is added in the `monitoring metrics` mode just like any other metric.

Example

```
[ddnmon61]% monitoring metrics
[ddnmon61->monitoring->metrics]% add responsiveness
[...[responsiveness]]% set command /cm/local/apps/cmd/scripts/metrics/s\
ample_responsiveness
[...*[responsiveness*]]% set classofmetric prototype; commit
```

For `classofmetric`, the value `prototype` is the class used to distinguish metric collections from normal metrics.

I.2 Metric Collections Initialization

When a metric collections script is added to the framework for the first time, it is implicitly run by CMDaemon with the `--initialize` flag. What this does is to detect and add the component metrics of the collections script to the framework.

The displayed output of a metric collections script when using the `--initialize` flag is a list of available metrics and their parameter values. The format of each line in the list is:

`metric` *<name>* *<unit>* *<class>* `"`*<description>*`"` *<cumulative>* *<min>* *<max>*

where the parameters are:

- `metric`: A bare word.

- <*name*>: The name of the metric.
- <*unit*>: A measurement unit.
- <*class*>: Any of: `misc cpu disk memory network environmental operatingsystem internal workload cluster`.
- <*description*>: This can contain spaces, but should be enclosed with quotes.
- <*cumulative*>: Either `yes` or `no`. This indicates whether the metric increases monotonically (e.g., bytes received) or not (e.g., temperature).
- <*min*> and <*max*>: Sensible values for a minimum and maximum.

Example

```
[root@myheadnode metrics]# ./sample_responsiveness --initialize
metric util_sda % internal "Percentage of CPU time during which I/O
requests were issued to device sda" no 0 100
metric await_sda ms internal "The average time (in milliseconds) for
I/O requests issued to device sda to be served" no 0 500
```

I.3 Metric Collections Output During Regular Use

The output of a metric collection script without a flag is a list of outputs from the available metrics. The format of each line in the list is:

```
metric <name> <value>
```

where the parameters are:

metric: A bare word.

name: The name of the metric.

value: The numeric value of the measurement.

Example

```
[root@myheadnode metrics]# ./sample_responsiveness
metric await_sda 0.00
metric util_sda 0.00
[root@myheadnode metrics]#
```

If the output has more metrics than that suggested by when the `--initialize` flag is used, then the extra sampled data is discarded. If the output has less metrics, then the metrics are set to `NaN` (not a number) for the sample.

A metric or health check inside a metric collection appears as a check when viewing metrics or healthcheck lists. Attempting to remove such a check specifically using `cmsh` or `cmgui` only succeeds until the node is updated or rebooted. It is the metric collection itself that should have the check removed from within it, in order to remove the check from the list of checks permanently.

Setting a node that is `UP` to a `CLOSED` state, and then bringing it out of that state with the `open` command (section 6.5.4) also has CMDaemon run the metric collections script with the `--initialize` flag. This is useful for allowing CMDaemon to re-check what metrics in the collections can be sampled, and then re-configure them.

I.4 Error Handling

As long as the exit code of the script is 0, the framework assumes that there is no error. So, with the `--initialize` flag active, despite no numeric value output, the script does not exit with an error.

If the exit code of the script is non-zero, the output of the script is assumed to be a diagnostic message and passed to the head node. This in turn will be shown as an event in `cmsh` or `cmgui`.

For example, the `sample_ipmi` script uses the `ipmi-sensors` binary internally. Calling the binary directly returns an error code if the device has no IPMI configured. However, the `sample_ipmi` script in this case simply returns 0, and no output. The rationale here being that the administrator is aware of this situation and would not expect data from that IPMI anyway, let alone an error.

I.5 Environment Variables

The following environment variables are available for a metric collection script (as well as for custom scripts) running from CMDaemon:

On all devices:

`CMD_HOSTNAME`: name of the device. For example:

```
CMD_HOSTNAME=myheadnode
```

Only on non-node devices:

`CMD_IP`: IP address of the device. For example:

```
CMD_IP=192.168.1.33
```

Only on node devices:

Because these devices generally have multiple interfaces, the single environment variable `CMD_IP` is often not enough to express these. Multiple interfaces are therefore represented by these environment variables:

- `CMD_INTERFACES`: list of names of the interfaces attached to the node. For example:

  ```
  CMD_INTERFACES=eth0 eth1 ipmi0 BOOTIF
  ```

- `CMD_INTERFACE_<interface>_IP`: IP address of the interface with the name <interface>. For example:

  ```
  CMD_INTERFACE_eth0_IP=10.141.255.254
  CMD_INTERFACE_eth1_IP=0.0.0.0
  ```

- `CMD_INTERFACE_<interface>_TYPE`: type of interface with the name <interface>. For example:

  ```
  CMD_INTERFACE_eth1_TYPE=NetworkPhysicalInterface
  CMD_INTERFACE_ipmi0_TYPE=NetworkBmcInterface
  ```

Possible values are:

- NetworkBmcInterface
- NetworkPhysicalInterface
- NetworkVLANInterface
- NetworkAliasInterface
- NetworkBondInterface

- CMD_BMCUSERNAME: username for the BMC device at this node (if available).
- CMD_BMCPASSWORD: password for the BMC device at this node (if available).

To parse the above information to get the BMC IP address of the node for which this script samples, one could use (in Perl):

```
my $ip;
my $interfaces = $ENV{"CMD_INTERFACES"};
foreach my $interface ( split( " " ,  $interfaces ) ) {
  if( $ENV{"CMD_INTERFACE_" . $interface . "_TYPE"} eq
"NetworkBmcInterface" ) {
    $ip = $ENV{"CMD_INTERFACE_" . $interface . "_IP"};
    last;
  }
}
# $ip holds the bmc ip
```

A list of environment variables available under the CMDaemon environment can be found by running a script under CMDaemon and exporting the environment variables to a file for viewing. For example, the /cm/local/apps/cmd/scripts/healthchecks/testhealthcheck script can be modified and run to sample on the head node, with the added line: set>/tmp/environment. The resulting file /tmp/environment that is generated as part of the healthcheck run then includes the CMD_* environment variables.

Example

```
CMD_BMCPASSWORD
CMD_BMCUSERNAME
CMD_CLUSTERNAME
CMD_CMDSTARTEDTIME
CMD_DEVICE_TYPE
CMD_EXPORTS
CMD_FSEXPORT__SLASH_cm_SLASH_shared_AT_internalnet_ALLOWWRITE
CMD_FSEXPORT__SLASH_cm_SLASH_shared_AT_internalnet_HOSTS
CMD_FSEXPORT__SLASH_cm_SLASH_shared_AT_internalnet_PATH
CMD_FSEXPORT__SLASH_home_AT_internalnet_ALLOWWRITE
CMD_FSEXPORT__SLASH_home_AT_internalnet_HOSTS
CMD_FSEXPORT__SLASH_home_AT_internalnet_PATH
CMD_FSEXPORT__SLASH_var_SLASH_spool_SLASH_burn_AT_internalnet_ALLOWWRITE
CMD_FSEXPORT__SLASH_var_SLASH_spool_SLASH_burn_AT_internalnet_HOSTS
CMD_FSEXPORT__SLASH_var_SLASH_spool_SLASH_burn_AT_internalnet_PATH
CMD_HOSTNAME
CMD_INTERFACES
```

```
CMD_INTERFACE_eth0_IP
CMD_INTERFACE_eth0_MTU
CMD_INTERFACE_eth0_SPEED
CMD_INTERFACE_eth0_STARTIF
CMD_INTERFACE_eth0_TYPE
CMD_INTERFACE_eth1_IP
CMD_INTERFACE_eth1_MTU
CMD_INTERFACE_eth1_SPEED
CMD_INTERFACE_eth1_STARTIF
CMD_INTERFACE_eth1_TYPE
CMD_IP
CMD_MAC
CMD_METRICNAME
CMD_METRICPARAM
CMD_MOUNTS
CMD_NODEGROUPS
CMD_PARTITION
CMD_PORT
CMD_PROTOCOL
CMD_ROLES
CMD_SCRIPTTIMEOUT
CMD_STATUS
CMD_STATUS_CLOSED
CMD_STATUS_HEALTHCHECK_FAILED
CMD_STATUS_HEALTHCHECK_UNKNOWN
CMD_STATUS_MESSAGE
CMD_STATUS_RESTART_REQUIRED
CMD_STATUS_STATEFLAPPING
CMD_STATUS_USERMESSAGE
CMD_SYSINFO_SYSTEM_MANUFACTURER
CMD_SYSINFO_SYSTEM_NAME
CMD_USERDEFINED1
CMD_USERDEFINED2
```

I.6 Metric Collections Examples

DirectMon has several scripts in the `/cm/local/apps/cmd/scripts/metrics` directory. Among them are the metric collections scripts `testmetriccollection` and `sample_responsiveness`. A glance through them while reading this appendix may be helpful.

I.7 iDataPlex And Similar Units

IBM's iDataPlex is a specially engineered dual node rack unit. When the term iDataPlex is used in the following text in this section, it also implies any other dual node units that show similar behavior.

This section gives details on configuring an iDataPlex if IPMI metrics retrieval seems to skip most IPMI values from one of the nodes in the unit.

When carrying out metrics collections on an iDataPlex unit, DirectMon should work without any issues. However, it may be that due to the special paired node design of an iDataPlex unit, most IPMI metrics of one member of the pair are undetectable by the `sample_ipmi` script sampling on that particular node. The missing IPMI metrics can instead be retrieved from the second member in the pair (along with the IPMI metrics of the second member).

The output may thus look something like:

Example

```
[root@master01 ~]# cmsh
[master01]% device latestmetricdata node181 | grep Domain
Metric                       Value
---------------------------- -----
Domain_A_FP_Temp             23
Domain_A_Temp1               39
Domain_A_Temp2               37
Domain_Avg_Power             140
Domain_B_FP_Temp             24
Domain_B_Temp1               40
Domain_B_Temp2               37
[master01]% device latestmetricdata node182 | grep Domain
Metric                       Value
---------------------------- -----
Domain_A_FP_Temp             no data
Domain_A_Temp1               no data
Domain_A_Temp2               no data
Domain_Avg_Power             170
Domain_B_FP_Temp             no data
Domain_B_Temp1               no data
Domain_B_Temp2               no data
[master01]%
```

Because there are usually many iDataplex units in the rack, the metrics retrieval response of each node pair in a unit should be checked for this behavior.

The issue can be dealt with by DirectMon by modifying the configuration file for the `sample_ipmi` script in `/cm/local/apps/cmd/scripts/metrics/configfiles/sample_ipmi.conf`. Two parameters that can be configured there are `chassisContainsLeadNode` and `chassisContainsLeadNodeRegex`.

- Setting `chassisContainsLeadNode` to `on` forces the `sample_ipmi` script to treat the unit as an iDataPlex unit.

  In particular:

  – `auto` (recommended) means the unit is checked by the IPMI metric sample collection script for whether it behaves like an iDataPlex unit.

  – `on` means the unit is treated as an iDataplex node pair, with one node being a lead node that has all the IPMI metrics.

  – `off` means the unit is treated as a non-iDataPlex node pair, with each node having normal behavior when retrieving IPMI metrics. This setting may need to be used in case the default value of `auto` ever falsely detects a node as part of an iDataPlex pair.

- The value of `chassisContainsLeadNodeRegex` can be set to a regular expression pattern that matches the system information pattern for the name, as obtained by CMDaemon for an iDataPlex unit (or similar clone unit). The pattern that it is matched against is the output of:

  ```
  cmsh -c 'device ; sysinfo master | grep "^System Name"'
  ```

  If the pattern matches, then the IPMI sample collection script assumes the unit behaves like an iDataPlex dual node pair. The missing IPMI data values are then looked for on the lead node.

  The value of `chassisContainsLeadNodeRegex` is set to `iDataPlex` by default.

J

Changing The Network Parameters Of The Head Node

J.1 Introduction

After a cluster physically arrives at its site, the administrator often has to change the network settings to suit the site. Details on this are given in section 4.3.1 of the DirectMon *Administrator Manual*. However, it relies on understanding the material leading up to that section.

This document is therefore a quickstart document explaining how to change the IPv4 network settings while assuming no prior knowledge of DirectMon and its network configuration interface.

J.2 Method

A cluster consists of a head node, say `ddnmon61` and one or more regular nodes. The head node of the cluster is assumed to face the internal network (the network of regular nodes) on one interface, say `eth0`. The external network leading to the internet is then on another interface, say `eth1`. This is referred to as a *type 1* configuration in the manual.

Typically, an administrator gives the head node a static external IP address before actually connecting it up to the external network. This requires logging into the physical head node with the vendor-supplied root password. The original network parameters of the head node can then be viewed and set. For example for `eth1`:

```
# cmsh -c "device interfaces ddnmon61; get eth1 dhcp"
yes
```

Here, `yes` means the interface accepts DHCP server-supplied values.

Disabling DHCP acceptance allows a static IP address, for example `192.168.1.176`, to be set:

```
# cmsh -c "device interfaces ddnmon61; set eth1 dhcp no"
# cmsh -c "device interfaces ddnmon61; set eth1 ip 192.168.1.176; commit"
# cmsh -c "device interfaces ddnmon61; get eth1 ip"
192.168.1.176
```

Other external network parameters can be viewed and set in a similar way, as shown in table J.1. A reboot implements the networking changes.

Table J.1: *External Network Parameters And How To Change Them On The Head Node*

Network Parameter	Description	Operation	Command Used
IP*	IP address of head node on eth1 interface	view	`cmsh -c "device interfaces ddnmon61; get eth1 ip"`
		set	`cmsh -c "device interfaces ddnmon61; set eth1 ip` *`address`*`; commit"`
baseaddress*	base IP address (network address) of network	view	`cmsh -c "network get externalnet baseaddress"`
		set	`cmsh -c "network; set externalnet baseaddress` *`address`*`; commit"`
broadcastaddress*	broadcast IP address of network	view	`cmsh -c "network get externalnet broadcastaddress"`
		set	`cmsh -c "network; set externalnet broadcastaddress` *`address`*`; commit"`
netmaskbits	netmask in CIDR notation (number after "/", or prefix length)	view	`cmsh -c "network get externalnet netmaskbits"`
		set	`cmsh -c "network; set externalnet netmaskbits` *`bitsize`*`; commit"`
gateway*	gateway (default route) IP address	view	`cmsh -c "network get externalnet gateway"`
		set	`cmsh -c "network; set externalnet gateway` *`address`*`; commit"`
nameservers*, **	nameserver IP addresses	view	`cmsh -c "partition get base nameservers"`
		set	`cmsh -c "partition; set base nameservers` *`address`*`; commit"`
searchdomains**	name of search domains	view	`cmsh -c "partition get base searchdomains"`
		set	`cmsh -c "partition; set base searchdomains` *`hostname`*`; commit"`
timeservers**	name of timeservers	view	`cmsh -c "partition get base timeservers"`
		set	`cmsh -c "partition; set base timeservers` *`hostname`*`; commit"`

* If *address* is set to 0.0.0.0 then the value offered by the DHCP server on the external network is accepted

** Space-separated multiple values are also accepted for these parameters when setting the value for *address* or *hostname*.

J.3 Terminology

A reminder about the less well-known terminology in the table:

- `netmaskbits` is the netmask size, or prefix-length, in bits. In IPv4's 32-bit addressing, this can be up to 31 bits, so it is a number between 1 and 31. For example: networks with 256 (2^8) addresses (i.e. with host addresses specified with the last 8 bits) have a netmask size of 24 bits. They are written in CIDR notation with a trailing "`/24`", and are commonly spoken of as "slash 24" networks.
- `baseaddress` is the IP address of the network the head node is on, rather than the IP address of the head node itself. The `baseaddress` is specified by taking `netmaskbits` number of bits from the IP address of the head node. Examples:
  - A network with 256 (2^8) host addresses: This implies the first 24 bits of the head node's IP address are the network address, and the remaining 8 bits are zeroed. This is specified by using "0" as the last value in the dotted-quad notation (i.e. zeroing the last 8 bits). For example: 192.168.3.0
  - A network with 128 (2^7) host addresses: Here `netmaskbits` is 25 bits in size, and only the last 7 bits are zeroed. In dotted-quad notation this implies "128" as the last quad value (i.e. zeroing the last 7 bits). For example: 192.168.3.128.

When in doubt, or if the preceding terminology is not understood, then the values to use can be calculated using the head node's `sipcalc` utility. To use it, the IP address in CIDR format for the head node must be known.

When run using a CIDR address value of 192.168.3.130/25, the output is (some output removed for clarity):

```
# sipcalc 192.168.3.130/25

Host address            - 192.168.3.130
Network address         - 192.168.3.128
Network mask            - 255.255.255.128
Network mask (bits)     - 25
Broadcast address       - 192.168.3.255
Addresses in network    - 128
Network range           - 192.168.3.128 - 192.168.3.255
```

Running it with the `-b` (binary) option may aid comprehension:

```
# sipcalc -b 192.168.3.130/25

Host address            - 11000000.10101000.00000011.10000010
Network address         - 11000000.10101000.00000011.10000000
Network mask            - 11111111.11111111.11111111.10000000
Broadcast address       - 11000000.10101000.00000011.11111111
Network range           - 11000000.10101000.00000011.10000000 -
                          11000000.10101000.00000011.11111111
```

K DirectMon Python API

This appendix introduces the Python API of DirectMon. For a head node `ddnmon61`, the API reference for all available objects is available in a default cluster from the User Portal (section 11.9) at:

`https://ddnmon61//userportal/downloads/python`

It is also available directly on the head node itself at:

`file:///cm/local/docs/cmd/python/index.html`

K.1 Installation

The Python cluster manager bindings are pre-installed on the head node.

K.1.1 Windows Clients

For windows clients, Python version 2.5.X is needed. Newer versions of Python do not work with the API.

For Windows a redistributable package is supplied in the `pythoncm-dist` package installed on the cluster. The file at `/cm/shared/apps/pythoncm/dist/windows-pythoncm.` `.r15673.zip`—the exact version number may differ—is copied to the Windows PC and unzipped.

A Windows shell (`cmd.exe`) is opened to the directory where the Python bindings are. The `headnodeinfo.py` example supplied with the unzipped files has a line that has the following format:

```
    cluster = clustermanager.addCluster(<parameters>);
```

where <*parameters*> is either:

`'<URL>', '<PEMauth1>', '<PEMauth2>'`
or
`'<URL>', '<PFXauth>', '', '<password>'`

The <*parameters*> entry is edited as follows:

- the correct hostname is set for the <*URL*> entry. By default it is set to `https://localhost:8081`
- If PEM key files are to be used for client authentication,

  - <*PEMauth1*> is set to path of `cert.pem`
  - <*PEMauth2*> is set to the path of `cert.key`

- If a PFX file is used for client authentication,
  - <*PFXauth*> is set to path of `admin.pfx`
  - <*password*> is set to the password

To verify everything is working, it can be run as follows:

```
c:\python25\python headnodeinfo.py
```

K.1.2 Linux Clients

For Linux clients, a redistributable source package is supplied in the `pythoncm-dist` package installed on the cluster. The file at `/cm/shared/apps/pythoncm/dist/pythoncm--r18836-src.tar.bz2`—the exact version number may differ—is copied and untarred to any directory.

The `build.sh` script is then run to compile the source. About 4GB of memory is usually needed for compilation, and additional packages may be required for compilation to succeed. A list of packages needed to build Python cluster manager bindings can be found in the `README` file included with the package.

The `headnodeinfo.py` example supplied with the untarred files is edited as for in the earlier windows client example, for the `clustermanager.addCluster` line.

The path to the remote cluster manager library is added:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:remotecm
```

To verify everything is working, the following can be run:

```
python ./headnodeinfo.py
```

K.2 Examples

A set of examples can be found in `/cm/local/examples/cmd/python/` on the head node of the cluster.

K.2.1 First Program

A Python script is told to use the cluster manager bindings by importing `pythoncm` at the start of the script:

```
  import pythoncm
```

If not working on the cluster, the administrator needs to set the path where the shared libraries can be found (`pythoncm.so` in Linux, or `python.pyd` in windows). This is done by adding the following to the start of the script:

```
  import sys
  sys.path.append(".")     # path to pythoncm.so/python.pyd
```

Python cluster manager bindings allow for simultaneous connections to several clusters. For this reason the first thing to do is to create a ClusterManager object:

```
clustermanager = pythoncm.ClusterManager()
```

A connection to a cluster can now be made. There are two possible ways of connecting.

The first is using the certificate and private key file that `cmsh` uses by default when it authenticates from the head node.

```
cluster = clustermanager.addCluster('https://mycluster:8081',\
'/root/.cm/cmsh/admin.pem', '/root/.cm/cmsh/admin.key');
```

The second way uses the password protected `admin.pfx` file, which is generated with the `cmd -c` command. A Python script could ask for the password and store it in a variable for increased security.

```
cluster = clustermanager.addCluster('https://mycluster:8081',\
'/root/.cm/cmgui/admin.pfx', '', '<password>');
```

Having defined the cluster, a connection can now be made to it:

```
isconnected = cluster.connect()
if !isconnected:
 print "Unable to connect"
 print cluster.getLastError()
 exit(1)
```

If a connection cannot be made, the function `cluster.connect()` returns false. The function `cluster.getLastError()` shows details about the problem. The two most likely problems are due to a wrong password setting or a firewall settings issue.

Similar to `cmgui` and `cmsh`, the cluster object contains a local cache of all objects. This cache will be filled automatically when the connection is established. All changes to properties will be done on these local copies and will be lost after the Python scripts exits, unless a `commit` operation is done.

The most common operation is finding specific objects in the cluster.

```
active = cluster.find('active')
if active == None:
 print "Unable to find active head node"
 exit(1)
else:
 print "Hostname of the active head node is %s" % active.hostname
```

If creating an automated script that runs at certain times, then it is highly recommended to check if objects can be found. During a failover, for instance, there will be a period over a few minutes in which the active head node will not be set.

It is good practice to disconnect from the cluster at the end of the script.

```
cluster.disconnect()
```

When connecting to a cluster with a failover setup, it is the shared IP address that should be connected to, and not the fixed IP address of either of the head nodes.

K.3 Methods And Properties

K.3.1 Viewing All Properties And Methods

All properties visible in cmsh and cmgui are also accessible from Python cluster manager bindings. The easiest way to get an overview of the methods and properties of an object is to define the following function:

```
import re
def dump(obj):
  print "--- DUMP ---"
  for attr in dir(obj):
    p = re.compile('^__.*__$')
    if not p.match(attr):
      print "%s = %s" % (attr, getattr(obj, attr))
```

An overview of all properties and methods for the active head node can be obtained with:

```
active = cluster.find('active')
dump(active)
```

K.3.2 Property Lists

Most properties are straightforward and their names are almost identical to the cmsh equivalent.

For instance:

```
node.mac = '00:00:00:00:00:00'
category.softwareimage = cluster.find('testimage')
```

Properties that contain lists, like node.roles, node.interfaces, category.fsmounts and several others, are trickier to deal with. While iterating over a list property is simple enough:

```
for role in node.roles:
  print role.name
```

because of an implementation restriction, adding a new role requires that a local copy of the roles list be made:

```
roles = node.roles
provisioningrole = pythoncm.ProvisioningRole()   # Create a new pro\
                                                visioning role object
roles.append(provisioningrole)
node.roles = roles            # This will update the internal\
                                roles list with the local copy
```

K.3.3 Creating New Objects

Creating a new node can be done with:

```
node = pythoncm.Node()
```

This is valid command, but fairly useless because a node has to be a MasterNode, PhysicalNode or VirtualSMPNode. So to create a normal compute or login node, the object is created as follows:

```
node = pythoncm.PhysicalNode()
```

The first thing to do after creating a new object is to add it to a cluster.

```
cluster.add(node)
```

It is impossible to add one node to more than one cluster.

After the node has been added its properties can be set. In `cmsh` and `cmgui` this is semi-automated, but in Python cluster manager bindings it has to be done by hand.

```
node.hostname = 'node001'
node.partition = cluster.find('base')
node.category = cluster.find('default')
```

Similar to the node object, a `NetworkInterface` object has several subtypes: `NetworkPhysicalInterface`, `NetworkVLANInterface`, `NetworkAliasInterface`, `NetworkBondInterface`, and `NetworkIPMIInterface`.

```
interface = pythoncm.NetworkPhysicalInterface()
interface.name = 'eth0'
interface.ip = '10.141.0.1'
interface.network = cluster.find('internalnet')
node.interfaces = [interface]
node.provisioningInterface = interface
```

Having set the properties of the new node, it can now be committed.

```
cr = node.commit()
```

If a commit fails for some reason, the reason can be found:

```
if not cr.result:
  print "Commit of %s failed:" % node.resolveName()
  for j in range(cr.count):
    print cr.getValidation(j).msg
```

K.3.4 List Of Objects

In the following lists of objects:

- Objects marked with (*) cannot be used
- Trees marked with (+) denote inheritance

Roles

```
Role (*)
+ BackupRole
+ BootRole
+ DatabaseRole
+ EthernetSwitch
+ LoginRole
+ LSFClientRole
+ LSFServerRole
+ MasterRole
+ PbsProClientRole
+ PbsProServerRole
+ ProvisioningRole
+ SGEClientRole
+ SGEServerRole
+ SlurmClientRole
+ SlurmServerRole
+ SubnetManagerRole
+ TorqueClientRole
+ TorqueServerRole
```

Devices

```
Device (*)
+ Chassis
+ GpuUnit
+ GenericDevice
+ PowerDistributionUnit
+ Switch (*)
   + EthernetSwitch
   + IBSwitch
   + MyrinetSwitch
   Node (*)
   + MasterNode
   + SlaveNode (*)
     + PhysicalNode
     + VirtualSMPNode
```

Network Interfaces

```
NetworkInterface (*)
+ NetworkAliasInterface
+ NetworkBondInterface
+ NetworkIpmiInterface
+ NetworkPhysicalInterface
+ NetworkVLANInterface
```

Information Objects

```
ClusterSetup
GuiClusterOverview
GuiGpuUnitOverview
GuiNodeOverview
GuiNodeStatus
LicenseInfo
SysInfoCollector
VersionInfo
```

Monitoring Configuration Objects

```
MonConf
ConsolidatorConf
MonHealthConf
HealthCheck
MonMetricConf
ThreshActionConf
ThreshAction
Threshold
```

LDAP Objects

```
User
Group
```

Category Objects

```
Category
FSExport
FSMount
```

Miscellaneous Objects

```
SoftwareImage
```

`KernelModule`

`Network`

`NodeGroup`

`Partition`

`Rack`

K.3.5 Useful Methods

For The Cluster Object:

Name	Description
`find(<name>)`	Find the object with a given name, <*name*>
`find(<name>, <type>)`	Because it is possible to give a category and node the same name, sometimes the type <*type*> of the object needs to be specified too
`getAll(<type>)`	Get a list of all objects of a given type: e.g. device, category
`activeMaster()`	Get the active master object
`passiveMaster()`	Get the active master object
`overview()`	Get all the data shown in the `cmgui` cluster overview
`add(<object>)`	Add a newly created object <*object*> to the cluster. Only after an object is added can it be used
`pexec(<nodes>, <command>)`	Execute a command <*command*> on one or more nodes

For Any Object:

Name	Description
`commit()`	Save changes to the cluster
`refresh()`	Undo all changes and restore the object to its last saved state
`remove()`	Remove an object from the cluster
`clone()`	Make an identical copy. The newly created object is not added to a cluster yet

For Any Device:

Name	Description
`close()`	Close a device
`open()`	Open a device
`powerOn()`	Power on a device
`powerOff()`	Power off a device
`powerReset()`	Power reset a device
`latestMonitoringData()`	Return a list of the most recent monitoring data

For Any Node:

Name	Description
`overview()`	Get the data displayed in the `cmgui` node overview tab
`sysinfo()`	Get the data displayed in the `cmgui` node system information tab
`pexec(`*<command>*`)`	Execute a command

K.3.6 Useful Example Program

In the directory `/cm/local/examples/cmd/python` are some example programs using the python API.

One of these is `printall.py`. It displays values for objects in an easily viewed way. With `all` as the argument, it displays resource objects defined in a list in the program. The objects are 'Partition', 'MasterNode', 'SlaveNode', 'Category', 'SoftwareImage', 'Network', 'NodeGroup'. The output is displayed something like (some output elided):

Example

```
[root@ddnmon61 ~]# cd /cm/local/examples/cmd/python
[root@ddnmon61 python]# ./printall all
Partition base
 +- revision .......................
 |  name .......................... base
 |  clusterName ................... DDN  Cluster
...
 |  burnConfigs
 |   +- revision ..................
 |   |  name ...................... default
 |   |  description ............... Standard burn test.
 |   |  configuration ............. < 2780 bytes >
 |   +- revision ..................
 |   |  name ...................... long-hpl
...
 |  provisioningInterface ......... None
 |  fsmounts ...................... < none >
 |  fsexports
 |   +- revision ..................
 |   |  name ...................... /cm/shared@internalnet
 |   |  path ...................... /cm/shared
 |   |  hosts ..................... !17179869185!
...
Category default
 +- revision ......................
 |  name .......................... default
 |  softwareImage ................. default-image
 |  defaultGateway ................ 10.141.255.253
 |  nameServers ................... < none >
...
```

The values of a particular resource-level object, such as `nodegroup`, can be viewed by specifying it as the argument:

Example

```
[root@ddnmon61 python]# ./printall.py nodegroup
nodegroup us-east-1-director-dependents
 +- revision ......................
 |  name .......................... us-east-1-director-dependents
 |  nodes ......................... < none >
 |  type .......................... CLOUDDIRECTOR
 |  firstSeenTime ................. 0
 |  lastSeenTime .................. 0
nodegroup us-east-1-director-dependents-2
 +- revision ......................
 |  name .......................... us-east-1-director-dependents-2
 |  nodes ......................... < none >
 |  type .......................... CLOUDDIRECTOR
 |  firstSeenTime ................. 1327593444
 |  lastSeenTime .................. 1327593444
[root@ddnmon61 python]#
```

L Workload Manager Configuration Files Updated By CMDaemon

This appendix lists workload manager configuration files changed by CMDaemon, events causing such change, and the file or property changed.

L.1 Slurm

File/Property	Updates What?	Updated During
`/etc/slurm/slurm.conf`	head node	Add/Remove/Update nodes, hostname change
`/etc/slurm/slurmdbd.conf`	head node	Add/Remove/Update nodes, hostname change
`/etc/slurm/gres.conf`	all nodes	Add/Remove/Update nodes
`/etc/slurm/topology.conf`	head node	Add/Remove/Update nodes

L.2 Grid Engine

File/Property	Updates What?	Updated During
`$ROOT/default/common/host_aliases`	head node	hostname/domain change, failover
`$ROOT/default/common/act_qmaster`	head node	hostname/domain change, failover

...continues

...continued

File/Property	Updates What?	Updated During
`queue hostlist`	head node	Add/Remove/Update nodes
`queue slots`	head node	Add/Remove/Update nodes
`queue ngpus`	head node	Add/Remove/Update nodes

L.3 Torque

File/Property	Updates What?	Updated During
`$ROOT/spool/server_name`	head node	hostname/domain change, failover
`$ROOT/spool/torque.cfg`	head node	hostname/domain change, failover
`$ROOT/server_priv/acl_svr/acl_hosts`	head node	hostname/domain change, failover
`$ROOT/spool/server_priv/acl_svr/operators`	head node	hostname change, failover
`$ROOT/spool/server_priv/nodes`	head node	Add/Remove/Update nodes
`$ROOT/mom_priv/config`	software image	hostname change, failover

L.4 PBS Pro

File/Property	Updates What?	Updated During
`/etc/pbs.conf`	head node, software image	hostname/domain change, failover
`$ROOT/server_priv/acl_svr/operators`	head node	hostname change, failover
`$ROOT/spool/server_priv/nodes`	head node	Add/Remove/Update nodes
`$ROOT/server_priv/acl_hosts`	head node	Add/Remove/Update nodes

L.5 LSF

File/Property	Updates What?	Updated During

...continues

...continued

File/Property	Updates What?	Updated During
`$LSF_TOP/conf/lsf.conf`	head node	hostname/domain change, failover
`$LSF_TOP/conf/hosts`	cloud-director	add/remove/update cloud nodes
`$LSF_TOP/conf/lsbatch/<clustername>/configdir/lsb.queues`	head node	add/remove/update nodes
`$LSF_TOP/conf/lsbatch/<clustername>/configdir/lsb.hosts`	head node	add/remove/update nodes
`$LSF_TOP/conf/lsf.cluster.<clustername>`	head node	add/remove/update nodes

The default value of $LSF_TOP in DirectMon is `/cm/shared/apps/lsf`

L.6 openlava

File/Property	Updates What?	Updated During
`$LSF_ENVDIR/lsf.conf`	head node	hostname/domain change, failover
`$LSF_ENVDIR/hosts`	cloud-director	add/remove/update cloud nodes
`$LSF_ENVDIR/lsf.cluster.<clustername>`	head node	add/remove/update nodes
`$LSF_ENVDIR/lsb.queues`	head node	add/remove/update nodes
`$LSF_ENVDIR/lsb.hosts`	head node	add/remove/update nodes

The default value of $LSF_ENVDIR in DirectMon is `/cm/shared/apps/openlava/var/etc/`

M

Linux Distributions That Use Registration

This appendix describes setting up registered access for the DirectMon with the Red Hat and SUSE distributions.

The head node and regular node images can be set up with registered access to the enterprise Linux distributions of Red Hat and SUSE so that updates from their repositories can take place on the cluster correctly. This allows the distributions to continue to provide support and security updates. Registered access can also set up in order to create an up-to-date custom software image (section 10.6) if using Red Hat or SUSE as the base distribution.

Registered access can be avoided for the head node and regular node images by moving the registration requirement to outside the cluster. This can be done by configuring registration to run from a local mirror to the enterprise Linux distributions. The head node and regular node images are then configured to access the local mirror repository. This configuration has the benefit of reducing the traffic between the local network and the internet. However it should be noted that the traffic from node updates scales according to the number of regular node images, rather than according to the number of nodes in the cluster. In most cases, therefore, the added complication of running a separate repository mirror, is unlikely to be worth implementing.

M.1 Registering A Red Hat Enterprise Linux Based Cluster

To register a Red Hat Enterprise Linux (RHEL) system, Red Hat subscriptions are needed as described at `https://www.redhat.com/`. Registration with the Red Hat Network is needed to install new RHEL packages or receive RHEL package updates, as well as carry out some other tasks.

M.1.1 Registering A Head Node With RHEL

The `rhn_register` command can be used to register an RHEL head node. If the head node has no direct connection to the internet, an HTTP proxy can be configured as a command line option. The `rhn_register` man pages give details on configuring the proxy from the command line.

The head node can be registered in text mode by running:

```
[root@ddnmon61 ~]# rhn_register --nox
```

When the `rhn_register` command is run, the following screens are gone through in a standard run:

- Connect to Red Hat Network
- Information
- Enter login information for Red Hat Network
- Register a System Profile—Hardware
- Register a System Profile—Packages
- Send Profile Information to Red Hat Network
- Review system subscription details
- Finish setup

Some of the screens may require inputs before going on to the next screen.

The Red Hat Network base software channel subscription configuration is displayed in the "`Review system subscription details`" screen.

The Red Hat Network software channels subscriptions configuration can also be viewed outside of the `rhn_register` command by using the `rhn-channel` command.

For RHEL5 it can be run as:

```
[root@ddnmon61 ~]# rhn-channel -l
```

For RHEL6 it can be run as:

```
[root@ddnmon61~]# rhn-channel -L -u <Red Hat Network username>
-p \
<Red Hat Network password>
```

For a head node based on RHEL5, the following Red Hat Network software channels subscriptions need to be configured:

- rhel-x86_64-server-5
- rhel-x86_64-server-supplementary-5

For a head node based on RHEL6 these are:

- rhel-x86_64-server-6
- rhel-x86_64-server-optional-6

Typically, on an RHEL5-based head node, the `rhel-x86_64-server-supplementary-5` channel must still be added to the configuration. To do this, the `rhn-channel` command can be used as follows:

```
[root@ddnmon61~]# rhn-channel -a -c rhel-x86_64-server-supplementary-5 \
-u <Red Hat Network username> -p <Red Hat Network password>
```

Similarly, for an RHEL6-based head node configuration the `rhel-x86_64-server-optional-6` channel is added with:

```
[root@ddnmon61~]# rhn-channel -a -c rhel-x86_64-server-optional-6 -u \
<Red Hat Network username> -p <Red Hat Network password>
```

After registering the system with Red Hat Network the `rhnsd` daemon is enabled, and it then becomes active after a reboot. The `rhnsd` daemon synchronizes information regularly with the Red Hat Network.

The following commands are useful for handling `rhnsd` options at this stage:

Command	Description
`service rhnsd start`	make it active without a reboot
`chkconfig rhnsd off`	stop it becoming active on boot
`service rhnsd status`	see if it is currently running

After registration, the `yum-rhn-plugin` is enabled, so that `yum` can be used to install and update from the Red Hat Network repositories.

M.1.2 Registering A Software Image With RHEL

The `rhn_register` command can be used to register an RHEL software image. If the head node, on which the software image resides, has no direct connection to the internet, then an HTTP proxy can be configured as a command line option. The `rhn_register` man pages give details on configuring the proxy from the command line.

The default software image, `default-image`, can be registered by running the following on the head node:

```
[root@ddnmon61 ~]# chroot /cm/images/default-image rhn_register --nox
```

When the `rhn_register` command is run, the following screens are gone through in a standard run:

- Connect to Red Hat Network
- Information
- Enter login information for Red Hat Network
- Register a System Profile—Hardware
- Register a System Profile—Packages
- Send Profile Information to Red Hat Network
- System subscription details
- Finish setup

Some of the screens may require inputs before going on to the next screen.

The Red Hat Network base software channel subscription configuration is displayed in the "`Review system subscription details`" screen.

The Red Hat Network software channels subscriptions configuration can also be viewed outside of the `rhn_register` command by using the `rhn-channel` command.

For RHEL5 it can be run from the head node as:

```
[root@ddnmon61 ~]# chroot /cm/images/default-image rhn-channel -l
```

For RHEL6 it is:

```
[root@ddnmon61~]# chroot /cm/images/default-image rhn-channel -L -u \
<Red Hat Network username> -p <Red Hat Network password>
```

For an RHEL5-based software image, the following Red Hat Network software channels subscriptions need to be configured:

- rhel-x86_64-server-5
- rhel-x86_64-server-supplementary-5

For an RHEL6-based software image these are:

- rhel-x86_64-server-6
- rhel-x86_64-server-optional-6

Typically, for an RHEL5-based software image, the `rhel-x86_64-server-supplementary-5` channel must still be added to the configuration. To do this, the `rhn-channel` command can be used as follows on the head node:

```
[root@ddnmon61~]# chroot /cm/images/default-image rhn-channel
-a -c \
rhel-x86_64-server-supplementary-5 -u <Red Hat Network username> -p \
<Red Hat Network password>
```

Similarly, for an RHEL6-based software image configuration the `rhel-x86_64-server-optional-6` channel must be added. This can be done with:

```
[root@ddnmon61~]# chroot /cm/images/default-image rhn-channel
-a -c \
rhel-x86_64-server-optional-6 -u <Red Hat Network username> -p \
<Red Hat Network password>
```

After registering the software image with the Red Hat Network, the `rhnsd` daemon is enabled, and it then becomes active when a node boots. The `rhnsd` daemon synchronizes information regularly with the Red Hat Network.

The `rhnsd` daemon can be turned off in the default `default-image` software image with:

```
[root@ddnmon61 ~]# chroot /cm/images/default-image chkconfig rhnsd off
```

After registering, the `yum-rhn-plugin` is enabled within the software image, so `yum` can be used for the software image to install and update from the Red Hat Network repositories.

M.2 Registering A SUSE Linux Enterprise Server Based Cluster

To register a SUSE Linux Enterprise Server system, SUSE Linux Enterprise Server subscriptions are needed as described at `http://www.suse.com/`. Registration with Novell helps with installing new SLES packages or receiving SLES package updates, as well as to carry out some other tasks.

M.2.1 Registering A Head Node With SUSE

The `suse_register` command can be used to register a SUSE head node. If the head node has no direct connection to the internet, then the `HTTP_PROXY` and `HTTPS_PROXY` environment variables can be set, to access the internet via a proxy. Running the command with the help option, as "`suse_register --help`", provides further information about the command and its options.

The head node can be registered as follows:

```
[root@ddnmon61~]# suse_register -a email=<e-mail address> -a regcode-\
sles=<activation code> --restore-repos
```

The e-mail address used is the address that was used to register the subscription with Novell. When logged in on the Novell site, the activation code or registration code can be found at the products overview page after selecting "`SUSE Linux Enterprise Server`".

After registering, the SLES and SLE SDK repositories are added to the repository list and enabled.

The repository list can be viewed with:

```
[root@ddnmon61 ~]# zypper lr
```

and the head node can be updated with:

```
[root@ddnmon61 ~]# zypper refresh
[root@ddnmon61 ~]# zypper update
```

M.2.2 Registering A Software Image With SUSE

The `suse_register` command can be used to register a SUSE software image. If the head node on which the software image resides has no direct connection to the internet, then the `HTTP_PROXY` and `HTTPS_PROXY` environment variables can be set to access the internet via a proxy. Running the command with the help option, as "`suse_register --help`", provides further information about the command and its options.

The default software image `default-image` can be registered by running the following on the head node:

```
[root@ddnmon61~]# chroot /cm/images/default-image suse_register -n -a \
email=<e-mail address> -a regcode-sles=<activation code> --restore-repos
```

The e-mail address is the address used to register the subscription with Novell. When logged in on the Novell site, the activation code or registration code can be found at the products overview page after selecting "`SUSE Linux Enterprise Server`".

When running the `suse_register` command, warnings about the `/sys` or `/proc` filesystems can be ignored. The command tries to query hardware information via these filesystems, but these are empty filesystems in a software image, and only fill up on the node itself after the image is provisioned to the node.

Instead of registering the software image, the SLES repositories can be enabled for the `default-image` software image with:

```
[root@ddnmon61 ~]# cp /etc/zypp/repos.d/*novell* /cm/images/default-ima\
ge/etc/zypp/repos.d/
[root@ddnmon61 ~]# cp /etc/zypp/credentials.d/NCCcredentials /cm/images\
/default-image/etc/zypp/credentials.d/
```

The repository list of the `default-image` software image can be viewed with the chroot option, `-R`, as follows:

```
[root@ddnmon61 ~]# zypper -R /cm/images/default-image lr
```

and the software image can be updated with:

```
[root@ddnmon61 ~]# export PBL_SKIP_BOOT_TEST=1
[root@ddnmon61 ~]# zypper -R /cm/images/default-image refresh
[root@ddnmon61 ~]# zypper -R /cm/images/default-image update
[root@ddnmon61 ~]# zypper -R /cm/images/default-image clean --all
```

N Burning Nodes

The *burn framework* is a component of DirectMon™ that can automatically run test scripts on specified nodes within a cluster. The framework is designed to stress test newly built machines and to detect components that may fail under load. Nodes undergoing a burn session with the default burn configuration, lose their filesystem and partition data for all attached drives, and revert to their software image on provisioning after a reboot.

N.1 Test Scripts Deployment

The framework requires power management to be running and working properly so that the node can be power cycled by the scripts used. In modern clusters power management is typically achieved by enabling a baseboard management controller such as IPMI or iLO. Details on power management are given in Chapter 5.

The framework can run any executable script. The default test scripts are mostly `bash` shell scripts and Perl scripts. Each test script has a directory in `/cm/shared/apps/cmburn` containing the script. The directory and test script must have the same name. For example: `/cm/shared/apps/cmburn/disktest/disktest` is the default script used for testing a disk. More on the contents of a test script is given in section N.3.3.

N.2 Burn Configurations

A *burn configuration* file specifies the order of the tests that are run. Within the burn configuration the tests are normally grouped into sequences, and several sequences typically make up a phase. Phases in turn are grouped in either a pre-install section or post-install section. A simple example of such a burn configuration could therefore look like:

Example

```
<?xml version="1.0"?>
<burnconfig>

  <mail>
    <address>root@master</address>
    <address>some@other.address</address>
  </mail>

  <pre-install>

    <phase name="01-hwinfo">
      <test name="hwinfo"/>
      <test name="hwdiff"/>
      <test name="sleep" args="10"/>
    </phase>

    <phase name="02-disks">
      <test name="disktest" args="30"/>
      <test name="mce_check" endless="1"/>
    </phase>

  </pre-install>

  <post-install>

    <phase name="03-hpl">
      <test name="hpl"/>
      <test name="mce_check" endless="1"/>
    </phase>

    <phase name="04-compile">
      <test name="compile" args="6"/>
      <test name="mce_check" endless="1"/>
    </phase>

  </post-install>

</burnconfig>
```

N.2.1 Mail Tag

The optional `<mail>` tag pair can add a sequence of e-mail addresses, with each address enclosed in an `<address>` tag pair. These addresses receive burn failure and warning messages, as well as a notice when the burn run has completed.

N.2.2 Pre-install And Post-install

The pre-install part of a burn configuration is configured with the `<pre-install>` tag pair, and run from inside a node-installer environment. This environment is a limited Linux environment and allows some simpler tests to run before loading up the full Linux node environment.

Similarly, the post-install part of a burn configuration uses the `<post-install>` tag pair to run from inside the full Linux node environment. This environment allows more complex tests to run.

N.2.3 Post-burn Install Mode

The optional `<post-burn-install>` tag pair allows the administrator to specify the post-burn install mode (section 6.4.4). The tag pair can enclose a setting of AUTO, FULL, MAIN, or NOSYNC. The default setting is the install mode that was set before burn started.

N.2.4 Phases

The phases sections must exist. If there is no content for the phases, the phases tags must still be in place ("must exist"). Each phase must have a unique name and must be written in the burn configuration file in alphanumerical order. By default, numbers are used as prefixes. The phases are executed in sequence.

N.2.5 Tests

Each phase consists of one or more test tags. The tests can optionally be passed arguments using the args property of the burn configuration file (section N.2). If multiple arguments are required, they should be a space separated list, with the (single) list being the args property.

Tests in the same phase are run simultaneously.

Most tests test something and then end. For example, the disk test tests the performance of all drives and then quits.

Tests which are designed to end automatically are known as *non-endless* tests.

Tests designed to monitor continuously are known as *endless tests*. Endless tests are not really endless. They end once all the non-endless tests in the same phase are ended, thus bringing an end to the phase. Endless tests typically test for errors caused by the load induced by the non-endless tests. For example the `mce_check` test continuously keeps an eye out for Machine Check Exceptions while the non-endless tests in the same phase are run.

N.3 Running A Burn Configuration

Burn configurations can be viewed and executed from `cmsh` or `cmgui`.

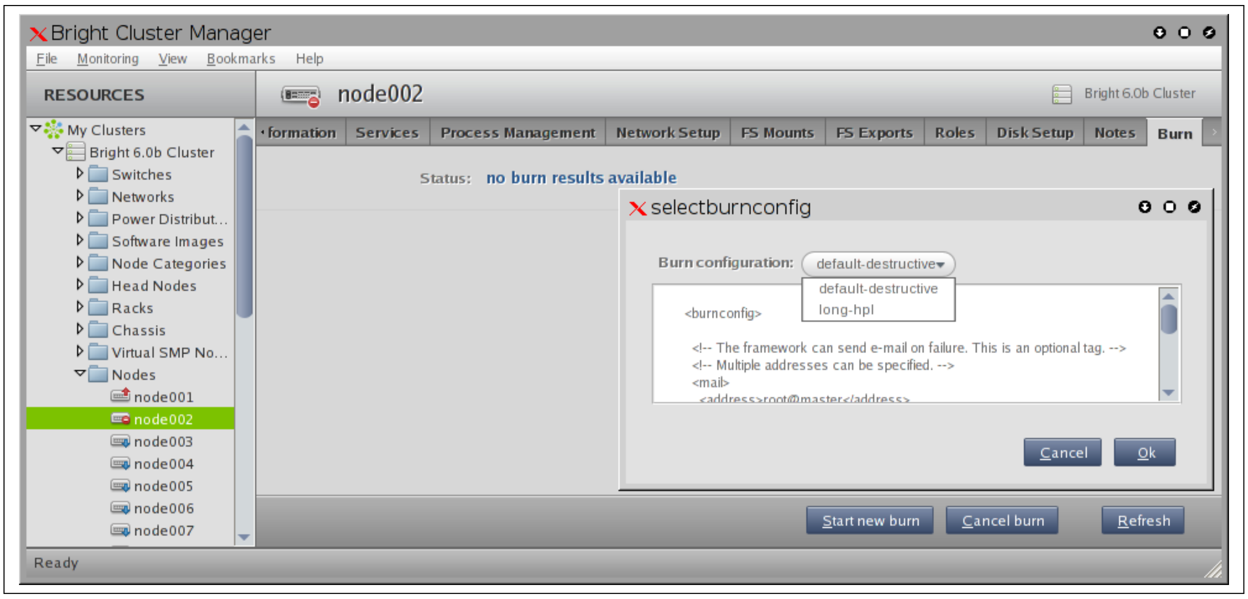
N.3.1 Burn Configuration And Execution In `cmgui`

From within `cmgui` the configuration can be selected for a particular node by selecting a node from the `Nodes` resource, then selecting the `Burn` tab. Clicking on the "`Start New Burn`" button opens up the burn configuration file selection dialog (figure N.1).

Clicking on the `OK` button then means the burn setting is turned on, as well as sending a power reset signal to the node via the Baseboard Management Controller or the APC. The burn setting being on means the node starts up in burn mode when starting up from now on (section 6.5.3). The "`Cancel burn`" button can be used to turn the burn setting off from `cmgui`, and then also sends a power reset signal.

The burn status of a node can be monitored for each individual node (figure N.2).

An overview of the burn status of nodes in a particular category can also be viewed for all nodes in the category (figure N.3). The category view is accessible when selecting the `Nodes` resource folder, and then selecting the "`Burn Overview`" tab. It can also be accessed by selecting the category item associated with the node from "`Node Categories`" resource, and then selecting the "`Burn Overview`" tab.



Status: no burn results available

Burn configuration: default-destructive

default-destructive
long-hpl

```
<burnconfig>

  <!-- The framework can send e-mail on failure. This is an optional tag. -->
  <!-- Multiple addresses can be specified. -->
  <mail>
   <address>root@master</address>
```

Figure N.1: `cmgui`: Starting A New Burn

N.3.2 Burn Configuration And Execution In `cmsh`

Burn configuration and execution can be carried out using `cmgui` (section N.3.1), or using `cmsh`. Using `cmsh` has some extra features, including the ability to create or modify a new burn configuration file, and also the ability to have the burn execution wait for a separate manual power reset.

Burn Configuration File Settings

From `cmsh`, the burn configurations can be accessed from `partition` mode as follows:

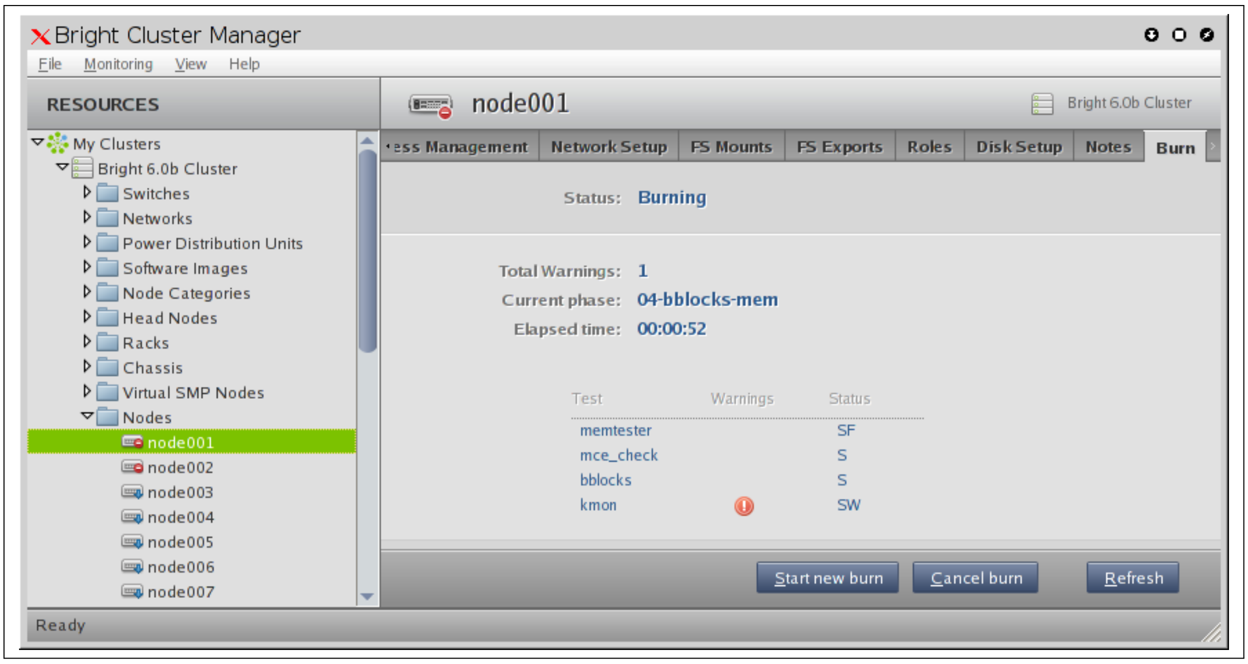
Example

```
[ddnmon61]% partition use base
[ddnmon61->partition[base]]% burnconfigs
[ddnmon61->partition[base]->burnconfigs]% list
Name (key)           Description               XML
-------------------- ------------------------- -------------
default-destructive  Standard burn test.       <2780 bytes>
long-hpl             Run HPL test for a long+  <879 bytes>
```

The values of a particular burn configuration (`default` in the following example) can be viewed as follows:

Example

```
[ddnmon61->partition[base]->burnconfigs]% use default-destructive
[ddnmon61->partition[base]->burnconfigs[default-destructive]]% show
Parameter                      Value
------------------------------ ----------------------------
Description                    Standard burn test.
Name                           default-destructive
Revision
XML                            <2780 bytes>
```

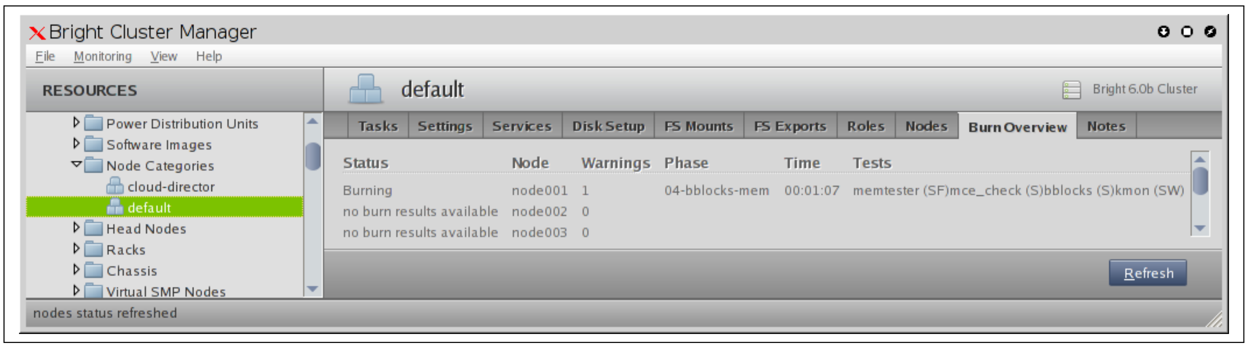


Status: Burning

Total Warnings: 1
Current phase: 04-bblocks-mem
Elapsed time: 00:00:52

Test	Warnings	Status
memtester		SF
mce_check		S
bblocks		S
kmon	!	SW

Figure N.2: `cmgui`: Status Of A Burn Run—Node View



Status	Node	Warnings	Phase	Time	Tests
Burning	node001	1	04-bblocks-mem	00:01:07	memtester (SF)mce_check (S)bblocks (S)kmon (SW)
no burn results available	node002	0			
no burn results available	node003	0			

Figure N.3: `cmgui`: Status Of A Burn Run—Category View

The `set` command can be used to modify existing values of the burn configuration, that is: `Description`, `Name` and `XML`. `XML` is the burn configuration file itself, and the default text editor opens up when using the set command on it, thus allowing the burn configuration to be modified.

A new burn configuration can also be added with the `add` command. The new burn configuration can be created from scratch with the `set` command. However, an XML file can also be imported to the new burn configuration by specifying the full path of the XML file to be imported:

Example

```
[ddnmon61->partition[base]->burnconfigs]% add boxburn
[ddnmon61->partition[base]->burnconfigs*[boxburn*]]% set xml /tmp/im.xml
```

The burn configuration can also be edited when carrying out burn execution with the `burn` command.

Executing A Burn

A burn as specified by the burn configuration file can be executed using `cmgui` (section N.3.1), or using `cmsh`. Execution using `cmsh` has some extra features, including the ability to have the burn execution wait for a separate manual power reset.

Burn-related properties: To execute a burn configuration file on a node in `cmsh`, the node object is accessed from `device` mode in `cmsh`. Among its properties, a node has

- `Burn config`: the selected burn configuration file
- `Burning`: the burn setting of the node. When its value is "`on`", and if the node has been power reset, then the node PXE boots into an image that runs burn tests according to the specifications of the burn configuration file

These properties can be viewed in `device` mode with the `show` command:

Example

```
[ddnmon61->device[node001]]% show | grep ^Burn
Burn config                             custom <2780 bytes>
Burning                                 no
```

Burn commands: The burn commands can modify these properties, as well as execute other burn-related operations.

The burn commands are executed within device mode, and are:

- `burn start`
- `burn stop`
- `burn status`
- `burn log`

The burn help text lists the detailed options (figure N.4). Next, operations with the burn commands illustrate how the options may be used along with some features.

Burn command operations: Burn commands allow the following operations, and have the following features:

- `start`, `stop`, `status`, `log`: The basic burn operations allow a burn to be started or stopped, and the status of a burn to be viewed and logged.
  - The "`burn start`" command always needs a configuration file name (here it is "`default-destructive`"), and also always needs to be given the nodes it operates on. For example:

    ```
    burn start -o default-destructive -n node001
    ```

```
[head1->device]% burn

Name:          burn - Node burn control

Usage:    burn [OPTIONS] status
          burn [OPTIONS] start
          burn [OPTIONS] stop
          burn [OPTIONS] log

Options:  -n, --nodes node(list)
               List of nodes, e.g. node001..node015,node20..node028,nod\
               e030 or ^/some/file/containing/hostnames

          -g, --group group(list)
               Include all nodes that belong to the node group, e.g. te\
               stnodes or test01,test03

          -c, --category category(list)
               Include all nodes that belong to the category, e.g. defa\
               ult or default,gpu

          -r, --rack rack(list)
               Include all nodes that are located in the given rack, e.\
               g rack01 or rack01..rack04

          -h, --chassis chassis(list)
               Include all nodes that are located in the given chassis,\
               e.g chassis01 or chassis03..chassis05

          -o, --config <name>
               Burn with the specified burn configuration. See in parti\
               tion burn configurations for a list of valid names

          -l, --later
               Do not reboot nodes now, wait until manual reboot

          -e, --edit
               Open editor for last minute changes

          -p, --path
               Show path to the burn log files. Of the form: /var/spool\
               /burn/<mac>.

Examples: burn -o default-destructive start -n node001
```

Figure N.4: Usage information for `burn`

- The "`burn stop`" command only needs to be given the nodes it operates on, for example:

  ```
  burn stop -n node001
  ```

- The "`burn status`" command:
  * may be given the nodes for which the status is to be found, for example:

```
burn status -n node001
```

  * need not have any nodes specified for it, for example:

    ```
    burn status
    ```

    in which case the burn status is shown for all nodes.
- The "`burn log`" command displays the burn log for specified node groupings. Each node with a boot MAC address of <*mac*> has an associated burn log file, by default under `/var/spool/burn/<mac>` on the head node.

- Advanced options allow the following:
  - `-n|--nodes`, `-g|--group`, `-c|--category`, `-r|--rack`, `-h|--chassis`: Burn commands can be executed over various node groupings.
  - `-o|--config`: The burn configuration file can be chosen from one of the burn configuration file names from `partition` mode.
  - `-l|--later`: This option disables the immediate power reset that occurs on running the "`burn start`" or "`burn stop`" command on a node.
  - `-e|--edit`: The burn configuration file can be edited with the `-e` option for the "`burn start`" command. This is an alternative to editing the burn configuration file in `partition` mode.
  - `-p|--path`: This shows the burn log path. The default burn log path is under `/var/spool/burn/<mac>`.

Burn command output examples: The `burn status` command has a compact one-line output per node:

Example

```
[ddnmon61->device]% burn -n node001 status
node001 (00000000a000) - W(0) phase 02-disks 00:02:58 (D:H:M) FAILED, m\
ce_check (SP), disktest (SF,61), kmon (SP)
```

The fields in the preceding output example are:

Description	Value	Meaning Here
The node name	node001	
The node tag	(00000000a000)	
Warnings since start of burn	(0)	
The current phase name	02-disks	Burn configuration phase being run is 02-disks
Time since phase started	00:02:58 (D:H:M)	2 hours 58 minutes
State of current phase	FAILED	Failed in 02-disks
burn test for MCE	mce_check (SP)	Started and Passed
burn test for disks	disktest (SF,61)	Started and Failed 61 is the speed and is custom information
burn test kernel log monitor	kmon (SP)	Started and Passed

Each test in a phase uses these letters to display its status:

Letter	Meaning
S	started
W	warning
F	failed
P	passed

The "burn log" command output looks like the following (some output elided):

```
[ddnmon61->device]% burn -n node001 log
Thu ... 2012: node001 - burn-control: burn framework initializing
Thu ... 2012: node001 - burn-control: e-mail will be sent to: root@master
Thu ... 2012: node001 - burn-control: finding next pre-install phase
Thu ... 2012: node001 - burn-control: starting phase 01-hwinfo
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/hwinfo
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/sleep
Thu ... 2012: node001 - sleep: sleeping for 10 seconds
Thu ... 2012: node001 - hwinfo: hardware information
Thu ... 2012: node001 - hwinfo: CPU1: vendor_id = AuthenticAMD
...
Thu ... 2012: node001 - burn-control: test hwinfo has ended, test passed
Thu ... 2012: node001 - burn-control: test sleep has ended, test passed
Thu ... 2012: node001 - burn-control: all non-endless test are done, terminating end\
less tests
Thu ... 2012: node001 - burn-control: phase 01-hwinfo passed
Thu ... 2012: node001 - burn-control: finding next pre-install phase
Thu ... 2012: node001 - burn-control: starting phase 02-disks
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/disktest
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/mce_check
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/kmon
Thu ... 2012: node001 - disktest: starting, threshold = 30 MB/s
Thu ... 2012: node001 - mce_check: checking for MCE's every minute
Thu ... 2012: node001 - kmon: kernel log monitor started
Thu ... 2012: node001 - disktest: detected 1 drives: sda
```

```
...
Thu ... 2012: node001 - disktest: drive sda wrote 81920 MB in 1278.13
Thu ... 2012: node001 - disktest: speed for drive sda was 64 MB/s -> disk passed
Thu ... 2012: node001 - burn-control: test disktest has ended, test FAILED
Thu ... 2012: node001 - burn-control: all non-endless test are done, terminating end\
less tests
Thu ... 2012: node001 - burn-control: asking test /cm/shared/apps/cmburn/kmon/kmon t\
o terminate
Thu ... 2012: node001 - kmon: kernel log monitor terminated
Thu ... 2012: node001 - burn-control: test kmon has ended, test passed
Thu ... 2012: node001 - burn-control: asking test /cm/shared/apps/cmburn/mce_check/m\
ce_check to terminate
Thu ... 2012: node001 - mce_check: terminating
Thu ... 2012: node001 - mce_check: waiting for mce_check to stop
Thu ... 2012: node001 - mce_check: no MCE's found
Thu ... 2012: node001 - mce_check: terminated
Thu ... 2012: node001 - burn-control: test mce_check has ended, test passed
Thu ... 2012: node001 - burn-control: phase 02-disks FAILED
Thu ... 2012: node001 - burn-control: burn will terminate
```

The output of the `burn log` command is actually the `messages` file in the burn directory, for the node associated with a MAC-address directory <*mac*>. The burn directory is at `/var/spool/burn/` and the `messages` file is thus located at:

/var/spool/burn/<*mac*>/messages

The tests have their log files in their own directories under the MAC-address directory, using their phase name. For example, the pre-install section has a phase named `01-hwinfo`. The output logs of this test are then stored under:

/var/spool/burn/<*mac*>/01-hwinfo/

N.3.3 Writing A Test Script

This section describes a sample test script for use within the burn framework. The script is typically a shell or Perl script. The sample that follows is a Bash script, while the `hpl` script is an example in Perl.

Section N.1 describes how to deploy the script.

Non-endless Tests

The following example test script is not a working test script, but can be used as a template for a non-endless test:

Example

```
#!/bin/bash

# We need to know our own test name, amongst other things for logging.
me=`basename $0`

# This first argument passed to a test script by the burn framework is a path
# to a spool directory. The directory is created by the framework. Inside the
# spool directory a sub-directory with the same name as the test is also created.
# This directory ($spooldir/$me) should be used for any output files etc. Note
# that the script should possibly remove any previous output files before starting.
spooldir=$1

# In case of success, the script should touch $passedfile before exiting.
passedfile=$spooldir/$me.passed

# In case of failure, the script should touch $failedfile before exiting.
# Note that the framework will create this file if a script exits without
# creating $passedfile. The file should contain a summary of the failure.
failedfile=$spooldir/$me.failed

# In case a test detects trouble but does not want the entire burn to be halted
# $warningfile _and_ $passedfile should be created. Any warnings should be written to this file.
warningfile=$spooldir/$me.warning

# Some short status info can be written to this file. For instance, the stresscpu
# test outputs something like 13/60 to this file to indicate time remaining.
# Keep the content on one line and as short as possible!
statusfile=$spooldir/$me.status

# A test script can be passed arguments from the burn configuration. It is
# recommended to supply default values and test if any values have been overriden
# from the config file. Set some defaults:
option1=40
option2=some_other_value

# Test if option1 and/or option2 was specified (note that $1 was to spooldir parameter):
if [ ! x$2 = "x" ]; then
  option1=$2
fi
if [ ! x$3 = "x" ]; then
  option2=$3
fi

# Some scripts may require some cleanup. For instance a test might fail and be
# restarted after hardware fixes.
rm -f $spooldir/$me/*.out &>/dev/null

# Send a message to the burn log file, syslog and the screen.
# Always prefix with $me!
blog "$me: starting, option1 = $option1 option2 = $option2"

# Run your test here:
run-my-test
if [ its_all_good ]; then
  blog "$me: wOOt, it's all good! my-test passed."
  touch $passedfile
  exit 0
elif [ was_a_problem ]; then
  blog "$me: WARNING, it did not make sense to run this test. You don't have special device X."
  echo "some warning" >> $warningfile  # note the append!
  touch $passedfile
  exit 0
else
```

```
  blog "$me: Aiii, we're all gonna die! my-test FAILED!"
  echo "Failure message." > $failedfile
  exit 0
fi
```

Endless Tests

The following example test script is not a working test, but can be used as a template for an endless test.

Example

```
#!/bin/bash

# We need to know our own test name, amongst other things for logging.
me=`basename $0`

# This first argument passed to a test script by the burn framework is a path
# to a spool directory. The directory is created by the framework. Inside the
# spool directory a sub-directory with the same name as the test is also created.
# This directory ($spooldir/$me) should be used for any output files etc. Note
# that the script should possibly remove any previous output files before starting.
spooldir=$1

# In case of success, the script should touch $passedfile before exiting.
passedfile=$spooldir/$me.passed

# In case of failure, the script should touch $failedfile before exiting.
# Note that the framework will create this file if a script exits without
# creating $passedfile. The file should contain a summary of the failure.
failedfile=$spooldir/$me.failed

# In case a test detects trouble but does not want the entire burn to be halted
# $warningfile _and_ $passedfile should be created. Any warnings should be written to this file.
warningfile=$spooldir/$me.warning

# Some short status info can be written to this file. For instance, the stresscpu
# test outputs something like 13/60 to this file to indicate time remaining.
# Keep the content on one line and as short as possible!
statusfile=$spooldir/$me.status

# Since this in an endless test the framework needs a way of stopping it once
# all non-endless test in the same phase are done. It does this by calling
# the script once more and passing a "-terminate" argument.
if [ "$2" == "-terminate" ]; then
  blog "$me: terminating"

  # remove the lock file the main loop is checking for
  rm $spooldir/$me/running

  blog "$me: waiting for $me to stop"
  # wait for the main loop to die
  while [ -d /proc/`cat $spooldir/$me/pid` ]
  do
    sleep 1
  done
  blog "$me: terminated"
```

```
else
  blog "$me: starting test, checking every minute"

  # Some scripts may require some cleanup. For instance a test might fail and be
  # restarted after hardware fixes.
  rm -f $spooldir/$me/*.out &>/dev/null

  # create internal lock file, the script will remove this if it is requested to end
  touch $spooldir/$me/running

  # save our process id
  echo $$ > "$spooldir/$me/pid"

  while [ -e "$spooldir/$me/running" ]
  do

    run-some-check
    if [ was_a_problem ]; then
      blog "$me: WARNING, something unexpected happened."
      echo "some warning" >> $warningfile  # note the append!
    elif [ failure ]; then
      blog "$me: Aiii, we're all gonna die! my-test FAILED!"
      echo "Failure message." > $failedfile
    fi
    sleep 60

  done

  # This part is only reached when the test is terminating.
  if [ ! -e "$failedfile" ]; then
    blog "$me: no problem detected"
    touch $passedfile
  else
    blog "$me: test ended with a failure"
  fi

fi
```

N.3.4 Burn Failures

Whenever the burn process fails, the output of the `burn log` command shows the phase that has failed and that the burn terminates.

Example

```
Thu ... 2012: node001 - burn-control: phase 02-disks FAILED
Thu ... 2012: node001 - burn-control: burn will terminate
```

Here, `burn-control`, which is the parent of the disk testing process, keeps track of the tests that pass and fail. On failure of a test, `burn-control` terminates all tests.

The node that has failed then requires intervention from the administrator in order to change state. The node does not restart by default. The administrator should be aware that the state reported by the node to CMDaemon remains `burning` at this point, even though it is not actually doing anything.

To change the state, the burn must be stopped (the `burn stop` command in `cmsh`, and the `Cancel burn` button in `cmgui`). If the node is restarted without explicitly stopping the burn, then it simply retries the phase at which it failed.

Under the burn log directory, the log of the particular test that failed for a particular node can sometimes suggest a reason for the failure. For retries, old logs are not overwritten, but moved to a directory with the same name, and a number appended indicating the try number. Thus:

Example

First try, and failing at `02-disks` tests:

```
cd /var/spool/burn/48:5b:39:19:ff:b3
ls -ld 02-disks*/
drwxr-xr-x 6 root root 4096 Jan 10 16:26 02-disks
```

2nd try, after failing again:

```
ls -ld 02-disks*/
drwxr-xr-x 6 root root 4096 Jan 10 16:49 02-disks
drwxr-xr-x 6 root root 4096 Jan 10 16:26 02-disks.1
```

3rd try, after failing again:

```
ls -ld 02-disks*/
drwxr-xr-x 6 root root 4096 Jan 10 16:59 02-disks
drwxr-xr-x 6 root root 4096 Jan 10 16:49 02-disks.1
drwxr-xr-x 6 root root 4096 Jan 10 16:26 02-disks.2
```

N.4 Relocating The Burn Logs

A burn run can append substantial amounts of log data to the default burn spool at `/var/spool/burn`. To avoid filling up the head node with such logs, they can be appended elsewhere.

N.4.1 Configuring The Relocation

The 3-part procedure that can be followed is:

1. The `BurnSpoolDir` setting can be set in the CMDaemon configuration file on the head node, at `/cm/local/apps/cmd/etc/cmd.conf`. The `BurnSpoolDir` setting tells CMDaemon where to look for burn data when the burn status is requested through `cmsh` and `cmgui`. It has the following value by default:

   - `BurnSpoolDir="/var/spool/burn"`

   CMDaemon should be restarted after the configuration has been set. This can be done with:

   ```
   service cmd restart
   ```

2. The `burnSpoolHost` setting, which matches the host, and `burnSpoolPath` setting, which matches the location, can be changed in the node-installer configuration file on the head node, at `/cm/node-installer/scripts/node-installer.conf`. These have the following values by default:

- `burnSpoolHost = master`
- `burnSpoolPath = /var/spool/burn`

These values define the NFS-mounted spool directory.

The `burnSpoolHost` value should be set to the new DNS host name, or to an IP address. The `burnSpoolPath` value should be set to the new path for the data.

3. Part 3 of the procedure adds a new location to export the burn log. This is only relevant if the spool directory is being relocated within the head node. If the spool is on an external fileserver, the existing burn log export may as well be removed.

   The new location can be added to the head node as a path value, from a writable filesystem export name. The writable filesystem export name can most easily be added using `cmgui`, under `FS Exports` for the host in the `Head Nodes`. Adding a new name like this is recommended, instead of just modifying the path value in an existing `FS Exports` name, just so that changing things back if the configuration is done incorrectly is easy. By default, the existing `FS Exports` for the burn directory has the name:

   - `/var/spool/burn@internalnet`

   and has a path associated with it with a default value of:

   - `/var/spool/burn`

   A new name can be set in `FS Exports`, and the associated path value can be set in agreement with the values set earlier in parts 1 and 2.

   In `cmgui` the configuration change is done via the `FS Exports` tab. In `cmsh` this can be set from within the `fsexports` submode. Section 4.11.1 gives more detail on similar examples of how to add such filesystem exports.

N.4.2 Testing The Relocation

To test the changes, it is wise to first try a single node with a short burn configuration. This allows the administrator to check install and post-install tests can access the spool directories. Otherwise there is a risk of waiting hours for the pre-install tests to complete, only to have the burn abort on the post-install tests. The following short burn configuration can be used:

Example

```
<burnconfig>
<pre-install>
<phase name="01-hwinfo">
<test name="hwinfo"/>
<test name="sleep" args="10"/>
</phase>
</pre-install>
<post-install>
<phase name="02-mprime">
```

```
<test name="mprime" args="2"/>
<test name="mce_check" endless="1"/>
<test name="kmon" endless="1"/>
</phase>
</post-install>
</burnconfig>
```

To burn a single node with this configuration, the following could be run from the device mode of `cmsh`:

Example

```
[ddnmon61->device]% burn start -o default-destructive -e -n node001
```

This makes an editor pop up containing the default burn configuration. The content can be replaced with the short burn configuration. Saving and quitting the editor causes the node to power cycle and start its burn.

The example burn configuration typically completes in less then 10 minutes or so, depending mostly on how fast the node can be provisioned. It runs the `mprime` test for about two minutes.

O

Changing The LDAP Password

The administrator may wish to change the LDAP root password. This procedure has two steps:

- setting a new password for the LDAP server (section O.1), and
- setting the new password in `cmd.conf` (section O.2).

It is also a good idea to do some checking afterwards (section O.3).

O.1 Setting A New Password For The LDAP Server

An encrypted password string can be generated as follows:

```
[root@ddnmon61 ~]# module load openldap
[root@ddnmon61 ~]# slappasswd
New password:
Re-enter new password:
SSHAJ/3wyO+IqyAwhh8Q4obL8489CWJlHpLg
```

The input is the plain text password, and the output is the encrypted password. The encrypted password is set as a value for the `rootpw` tag in the `slapd.conf` file on the head node:

```
[root@ddnmon61 ~]# grep ^rootpw /cm/local/apps/openldap/etc/slapd.conf
rootpw SSHAJ/3wyO+IqyAwhh8Q4obL8489CWJlHpLg
```

The password can also be saved in plain text instead of as an SSHA hash generated with `slappasswd`, but this is considered insecure.

After setting the value for `rootpw`, the LDAP server is restarted:

```
[root@ddnmon61 ~]# /etc/init.d/ldap restart
```

O.2 Setting The New Password In `cmd.conf`

The new LDAP password (the plain text password that generated the encrypted password after entering the `slappasswd` command in section O.1) is set in `cmd.conf`. It is kept as clear text for the entry for the LDAPPass directive (Appendix C):

```
[root@ddnmon61 ~]# grep LDAPPass /cm/local/apps/cmd/etc/cmd.conf
LDAPPass = "Mysecret1dappassw0rd"
```

CMDaemon is then restarted:

```
[root@ddnmon61 ~]# /etc/init.d/cmd restart
```

O.3 Checking LDAP Access

For a default configuration with user `cmsupport` and domain `cm.cluster`, the following checks can be run from the head node (some output truncated):

- anonymous access:

```
[root@ddnmon61 ~]# ldapsearch -x
# extended LDIF
#
# LDAPv3
# base <dc=cm,dc=cluster> (default) with scope subtree
...
```

- root cn without a password (this should fail):

```
[root@ddnmon61 ~]# ldapsearch -x -D 'cn=root,dc=cm,dc=cluster'
ldap_bind: Server is unwilling to perform (53)
 additional info: unauthenticated bind (DN with no password) disallowed
[root@ddnmon61 ~]#
```

- root cn with a password (this should work):

```
[root@ddnmon61 ~]# ldapsearch -x -D 'cn=root,dc=cm,dc=cluster' -w Mysec\
ret1dappassw0rd
# extended LDIF
#
# LDAPv3
# base <dc=cm,dc=cluster> (default) with scope subtree
...
```

P

Configuring SELinux

P.1 Introduction

Security-Enhanced Linux (SELinux) can be enabled on selected nodes. On a standard Linux operating system where SELinux is enabled, it is typically initialized in the kernel during the execution of the `init` script inside the initrd when booting from a hard drive. However, in the case of nodes provisioned by DirectMon, via PXE boot, the SELinux initialization occurs at the very end of the node installer phase.

SELinux is disabled by default because its security policies are typically customized to the needs of the organization using it. The administrator is therefore the one who must decide on appropriate access control security policies. When creating such custom policies special care should be taken that the `cmd` process is executed in, ideally, an unconfined context.

Before enabling SELinux on a cluster, the administrator is advised to first check that the Linux distribution used offers enterprise support for SELinux-enabled systems. This is because support for SELinux should be provided by the distribution in case of issues.

Enabling SELinux is only advised by DirectMon if the internal security policies of the organization absolutely require it. This is because it requires custom changes from the administrator. If something is not working right, then the effect of these custom changes on the installation must also be taken into consideration, which can sometimes be difficult.

P.2 Enabling SELinux On SLES11SP2 Systems

The default kernel provided with SLES11SP2 contains SELinux functionality. This functionality is, however, disabled by default. Furthermore, by default, the system does not have some important SELinux-related packages installed. SUSE openly states that it leaves it up to the user to configure the system to run properly with SELinux. It is, therefore, advised to use a Linux distribution based on Red Hat 6 with SELinux, if possible, as these have superior SELinux integration.

Some of the manual configuration changes required for SLES11SP2-specific systems are described in the following subsection. After these changes are in place, the process of enabling SELinux is analogous to the one for RHEL6-based systems, described in Section P.3.

P.2.1 Regular Nodes

DirectMon supports PXE booting of SLES11SP2 software images which have the SELinux functionality enabled. During the node installer phase SELinux is enabled by means of loading the initial policy, and applying the file security contexts to the filesystems.

Following subsections describe the minimal steps which have to be taken in order to prepare the SLES11SP2 software image to be run with SELinux.

Installing Missing SELinux Packages

SLES11SP2 by default comes with only some of the required SELinux-related packages installed. The missing packages have to be installed.

SELinux core utilities are installed under <*image*>, the software image directory name, for regular node use:

```
chroot /cm/images/<image>/ zypper in policycoreutils
```

It is also worth checking if are any other SELinux-related packages available in the public repositories. The SLES11SP2 release notes state that such packages might be released in the future.

```
chroot /cm/images/<image>/ zypper se --search-descriptions SELinux policy
```

Enabling SELinux In The kernel

In order to enable SELinux support in the kernel, two kernel parameters have to be added to the software image for regular node use:

```
[ddnmon61->softwareimage[default-image]]% set kernelparameters "security\
=selinux selinux=1"
[ddnmon61->softwareimage*[default-image*]]% commit
```

Creating The SELinux Configuration File

If the `/cm/images/<image>/etc/selinux/config` file is missing after installing additional packages, the file must be created. An example of a valid SELinux configuration file is:

Example

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
SELINUXTYPE=targeted
```

Providing A Security Policy

SLES11SP2 release notes state that a default reference security policy might be provided at some point. If there is no such policy in the public repositories, then a policy has to be provided by the user and installed in the directory: `/cm/images/<image>/etc/selinux/<policyname>`. The `<policyname>` file must match the value of the `SELINUXTYPE` directive in the SELinux configuration file.

Organizations intending to use SELinux typically already have such a security policy prepared. If not, such a policy must be created. The security policy provided as part of the OpenSUSE project could be used as a reference policy. Such a policy may yet require some changes in order to work properly in an SLES11SP2 environment.

Additional Steps

Depending on the security policy used and on the current state of the SELinux integration into SLES11SP2, some additional steps may still need to be taken by the user to make an SLES11SP2 system boot properly with SELinux enabled.

After the software image has been prepared, the remaining steps to enabling SELinux on SLES11SP2 are the same as for RHEL6 regular nodes (section P.3.1).

P.2.2 Head Node

Head nodes have SELinux enabled on them by following the same steps as regular nodes. The most obvious difference being that the operations apply to the `/` directory instead of the `/cm/images/<image>/` directory.

Another difference is that the kernel parameters (`"security=selinux selinux=1"`) must be manually added to the `/boot/grub/menu.lst` file for head nodes, instead of via the `softwareimage` mode of `cmsh`.

For head nodes, the user may need to perform filesystem security context relabeling, e.g. by using the `fixfiles restore` command. Such a relabeling should be done while running SELinux in `permissive` mode, and should be followed by a system restart.

P.3 Enabling SELinux on RHEL6

RedHat-based systems come with a default `targeted` policy which confines only some selected ("targeted") system services.

P.3.1 Regular Nodes

There are two ways to enable SELinux on regular nodes for RHEL6:

1. by configuring the node to boot a local disk.

2. using the node installer to set up SELinux during PXE boot

In both cases, before the regular node is provisioned, the `/cm/images/<image>/etc/selinux/config` file must be properly configured. This means configuring appropriate values for the `SELINUX` and `SELINUXTYPE` directives.

SELinux, With Booting Off A Local Disk

SELinux can be enabled on a regular node by first provisioning it via PXE, and then setting the `installbootrecord` property (section 6.4.10). The node will subsequently boot via the local hard drive, instead of from the network.

The downside to this method is that if the software image is updated, the filesystem of the node is not updated after a reboot.

SELinux with PXE booting

The other, and recommended, way to enable SELinux on a regular node is to have the node installer initialize the SELinux environment after provisioning the node. That is, the node installer loads the initial policy and applies proper security contexts to the filesystem.

To make the node installer initialize SELinux, the content of the `/cm/node-installer/scripts/node-installer.conf` file (located on the head node) must be edited. The value of the `SELinuxInitialize` directive should be changed from `false` to `true`. When the node is rebooted with this setting, SELinux initializes via the node installer after provisioning has been completed, and before the node installer finishes its execution.

P.3.2 Head Node

In order to enable SELinux on the head node in RHEL6:

- The `/etc/selinux/config` file must be edited (according to the organizational requirements) in the same way as for regular nodes.
- An `/.autorelabel` file should be created on the head node's filesystem.
- The kernel parameters (`"security=selinux selinux=1"`) must be manually added to the `/boot/grub/menu.lst` file.
- The head node must then be restarted.

P.4 Additional Considerations

It is advised that the `/cm/images/<`*`image`*`>/.autorelabel` file only gets transferred to the regular nodes during a FULL node installation. I.e., it should not be transferred during the AUTO node installation. This can be achieved by appending the following entry to the `excludelistsyncinstall` property of the node category:

```
no-new-files: - /.autorelabel
```

Why this is done is explained in section P.5.

P.5 Filesystem Security Context Checks

Ensuring that the files present on the node have correct security contexts applied to them is an important part of enforcing a security policy.

In the case of the head nodes, the filesystem security context check is performed by the default system startup scripts. This is, by default, performed only if the presence of the `/.autorelabel` file on the root filesystem is detected.

In the case of the regular nodes the process is significantly different. For these, by default, it is the node installer that is responsible for the correctness of security contexts on the filesystem of the nodes.

If the regular node has undergone full provisioning, then if the `/.autorelabel` file exists on the node's local filesystem, the security contexts of all eligible filesystems of that node are restored by the node installer. This behavior is analogous to what the startup subsystem scripts would normally do. However, since the node installer removes the

`/.autorelabel` file after performing the context restore, the operating system startup script does not detect it once the system boot continues.

If the node has undergone a sync provisioning (e.g. installed in AUTO mode), then after enabling SELinux, the node installer will only restore the security context on the files which were modified during provisioning and on files which were generated by the node installer itself. This is typically significantly faster than performing a full filesystem security context restore on all eligible filesystems.

The behavior described above can be altered using the `/cm/node-installer/script/node-installer.conf` configuration file. For example, it is always possible to force a full filesystem security context restore in the AUTO install mode, or to leave the context checking to the operating system's startup scripts.