TAS Connector Administrator Guide (S-2534-20)

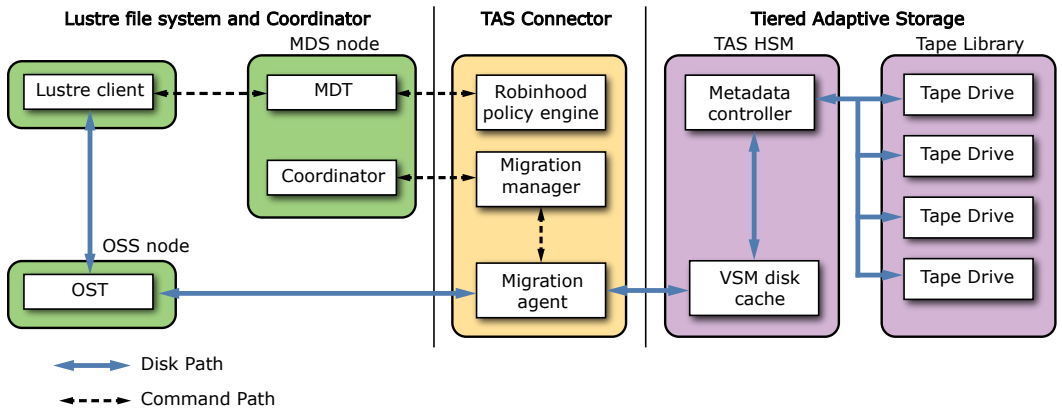
Contents

TAS Connector System Introduction

Tiered Adaptive Storage (TAS) and TAS Connector

Cray Tiered Adaptive Storage (TAS) is a hierarchical storage management (HSM) system that uses multiple storage tiers to manage and archive large file systems. The TAS Connector system interconnects a Lustre® file system with the tape archive backend.

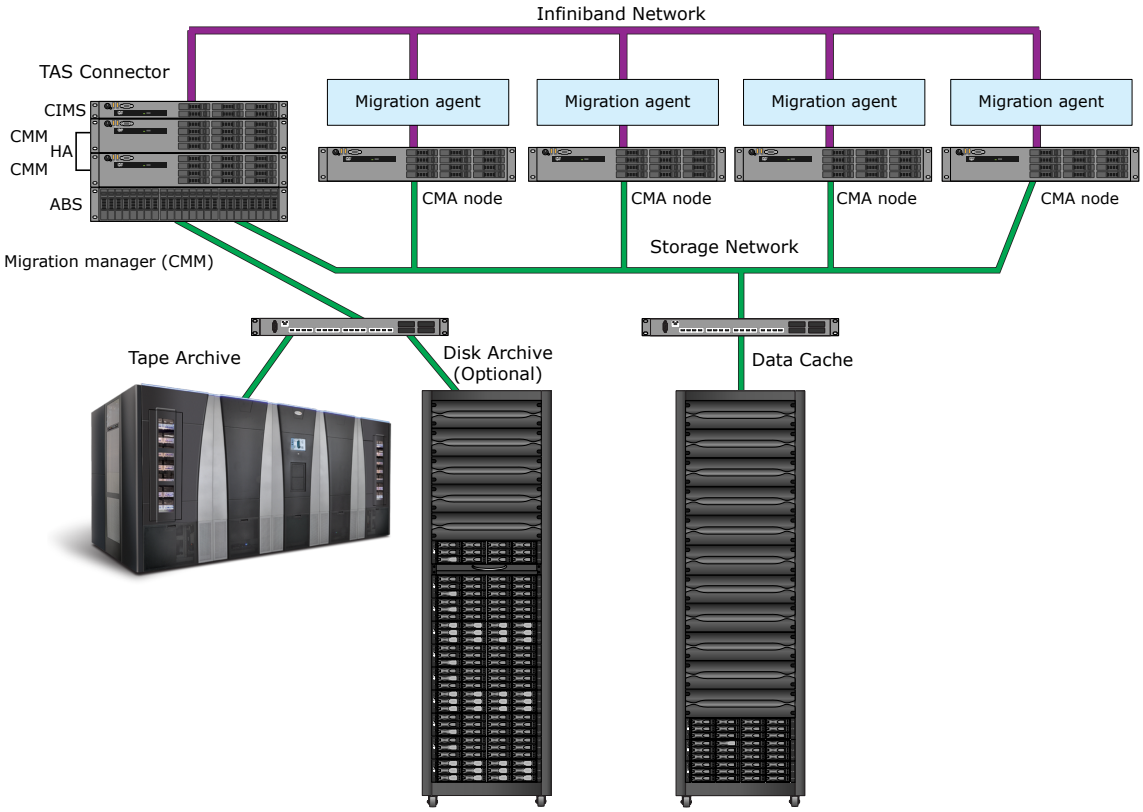
Figure 1. TAS HSM and Connector Block Diagram



System Description

The TAS Connector is a hardware and software platform that connects a Lustre® file system to Cray's hierarchical storage management backend (TAS HSM), which implements a Versity Storage Manager (VSM) file system. A Robinhood policy engine and MariaDB SQL database communicate with the Lustre® HSM coordinator to automatically move data between the VSM file system on the TAS and Lustre® file systems based on customer policies.

Figure 2. TAS Connector System Hardware Components



Migration manager nodes (CMM nodes) operate in a failover configuration and run a Robinhood policy engine and MariaDB database. CMM nodes are connected to an external administrative block storage (ABS) device. The migration agent nodes (CMA nodes) service requests from the CMM nodes and move data between the Lustre® file system, and backend TAS HSM system.

The system uses a Cray Integrated Management Services (CIMS) node running Bright Cluster Manager® 6.1 software (Bright) to manage service nodes, store software images, and monitor the system.

Note that Cray's implementation of Bright Cluster Manager (Bright) 6.1 software leverages its capability to manage Cray's specialized management services node (CIMS node) and the service nodes in the TAS Connector system. Bright is not used to manage a TAS system as a traditional compute cluster. It follows that some Bright features such as cloudbursting are not implemented in Cray TAS Connector or TAS systems. This content provides an introduction to Bright, and shows examples of how to use Bright to manage the system.

Scope of this Content

This content assumes the administrator has previous experience with operating tape archive systems, SAM-QFS and Lustre® file systems, the Robinhood policy engine, and MariaDB data software.

Note that Cray's implementation of Bright Cluster Manager (Bright) software leverages its capability to manage Cray's specialized management services node (CIMS node) and the service nodes in the TAS system. Bright is not used to manage a TAS system as traditional compute cluster. It follows that some Bright features such as cloudbursting are not implemented in Cray TAS systems.

This content provides only an overview of Bright, and shows some examples of how to use Bright to manage the system. Refer to the *Data Management Practices (DMP) Administrator Guide, S-2327* for additional administration procedures and examples for using Bright to manage the CIMS node and service nodes.

Related Content

Table 1. Related Content

Document	Description
Data Management Practices (DMP) Administrator Guide, S-2327	Provides software administration procedures for the Cray TAS CIMS node and Cray service nodes.
Cray Integrated Management Services (CIMS) Software Installation Guide, S-2522	Provides software installation and configuration procedures for the Cray TAS CIMS node.
Lustre File System by Cray (CLFS) Software Installation Guide, S-2521	Describes the initial software installation and configuration procedures for TAS service nodes.
Bright Cluster Manager 6.1 Administrator Manual	Includes information about the Bright software, and describes how to use the user interface (`cmgui`) and management shell (`cmsh`) to perform common administrative tasks. A PDF of this document is stored on the CIMS node in the `/cm/shared/docs/cm` directory.
Tiered Adaptive Storage Hardware Guide, HR85-8500	Provides an high-level introduction, architectural overview, site planning and system specifications, hardware implementation, and details for the Dell servers and NetApp block storage devices for the administrative block storage (ABS) and data-cache block storage DCBS) devices.
DELL™ R630 and R730 Hardware Owners Guides, Dell Ethernet Switch and KVM software documentation, available from *www.dell.com/support*.	Dell equipment documentation.
NetApp storage device software documentation is available from *www.netapp.com*	NetApp storage equipment documentation.

Revisions to this Content

Table 2. Revisions to this Content

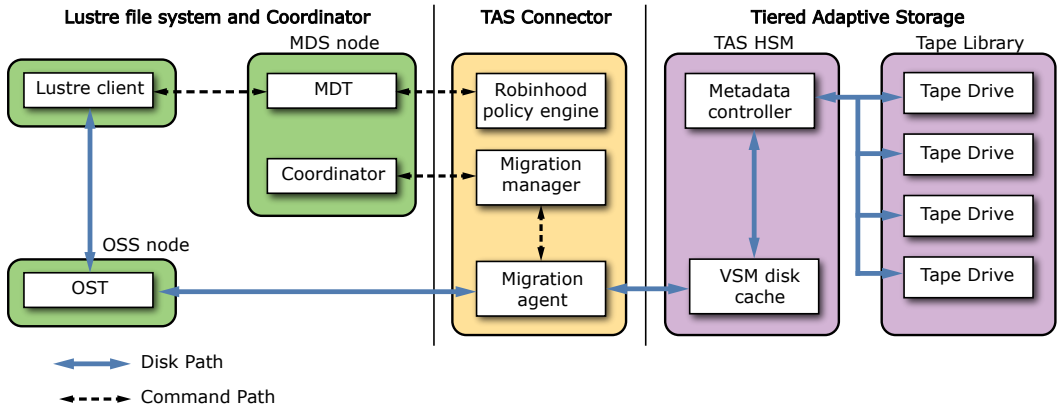
Revision	Description	Supported Products	Date
2.0	Original printing.	Cray TAS Connector systems.	March 2015

TAS HSM System Description

Cray Tiered Adaptive Storage (TAS) implements hierarchical storage management (HSM) features that use multiple storage tiers to manage and archive large file systems. TAS systems automatically move data between high-cost high-bandwidth storage devices and low-cost low-bandwidth storage devices based on customer policies.

TAS Connector systems interconnect a Lustre® file system with a tape archive backend.

Figure 3. TAS HSM and Connector Block Diagram



TAS systems run Cray proprietary TAS software and Versity® Storage Manager (VSM) software. VSM is a proprietary Linux version of SAM-QFS® developed by Versity, Inc. The TAS system uses an open architecture and Versity's open file format.

The Cray TAS system has two major functional capabilities:

- Provides a POSIX shared file system with good support for small files and support for hundreds of native file system clients, native interfaces to storage tiers, and integrated volume management.
- Provides an archive capability that manages data between disk and/or tape storage tiers via a policy engine that can manage up to 4 copies of each file.

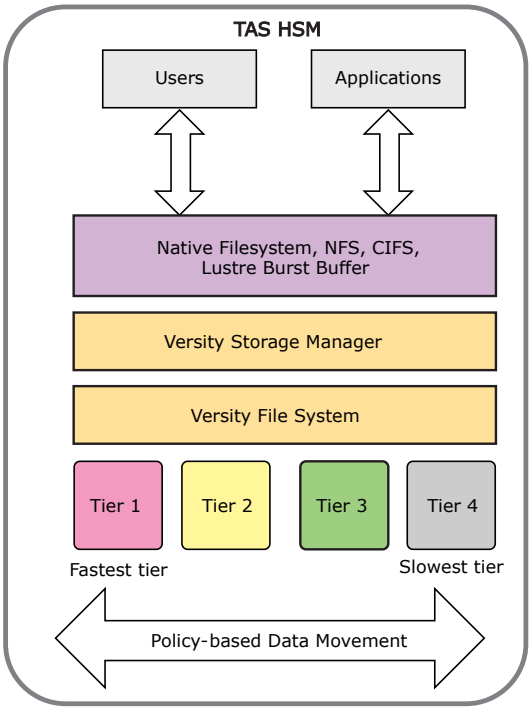
Each storage system is defined as a tier in VSM, based on storage capacity and/or I/O bandwidth. The TAS system stores data on slower devices (lower tiers) and copies the data to faster storage devices (higher tiers) based on site policies. TAS systems are built on the foundation of Data Management Practices (DMP) management and server node hardware and software, high-performance storage systems, and tape archive systems.

Up to four physical tiers are supported:

- Tier 0 – Performance-optimized for high I/O and throughout (disk or SSD)
- Tier 1 – Primary storage where live data lives most of time (disk or SSD)
- Tier 2 – Capacity-optimized nearline storage (disk or tape)

- Tier 3 – Extreme capacity- and cost-optimized for deep archives (usually tape)

Figure 4. TAS System Architecture



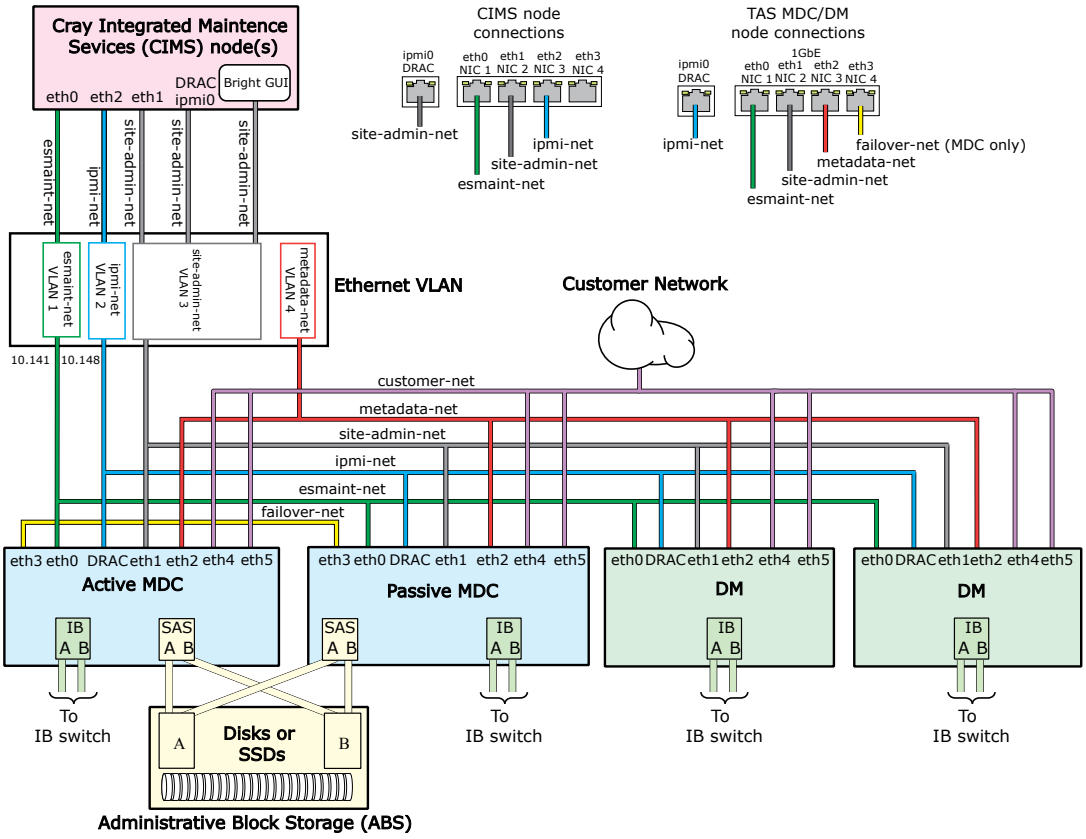
TAS HSM major components are:

- A Cray integrated management services node (CIMS)
- A metadata management system that includes two metadata controller (MDC) nodes
- A fast data disk cache block storage (DCBS) device
- An optional capability to support one or more disk or tape archive tiers
- A GigE management network (which is usually a separate VLAN on the metadata network)
- A Gigabit Ethernet (GigE) or 10 Gigabit Ethernet (10GbE) metadata network VLAN
- An InfiniBand (IB) quad-data rate (QDR) storage network
- An optional FC tape archive network
- Connectivity to a customer network
- A policy engine controls movement of data between storage tiers

There are 2 types of switches, storage area network (SAN) switches and network switches. The SAN switches may be fibre channel (QLogic) or InfiniBand (Mellanox). These switches provide a data storage fabric to connect the MDC and DM nodes to the cache block storage and to connect the MDCs to the robotic tape library.

The network VLAN Ethernet switches provide administrative communication between the CIMS node, and the MDC and DM nodes. The VLAN provides access for system administrators and all the servers (CIMS, MDC, and DM). In addition, the VLAN provides metadata communication between the MDC and DM nodes.

Figure 5. Cray TAS System Components and Networks



Lustre® HSM Features

Coordinator

The coordinator is a service that coordinates the migration of data and dispatches requests to agents (CMA nodes) to restore <FIDlist>, archive <FIDlist>, unlink <FIDlist> or abort an action. The coordinator consolidates and requeues requests to a new agent if an agent becomes unresponsive. Requests to the coordinator from the migration manager node (CMM) are stored in a log, and cancelled when the request has completed or has aborted. Agents send status updates to the coordinator and the coordinator checks for stuck threads periodically. The coordinator is integrated into the Lustre MDT and communicates with agents through the Lustre MDC.

Copytool

TAS Connector Copytool copies data between a Lustre file system and the TAS HSM file system, Versity Storage Manager (VSM).

The TAS Connector Copytool copies data between a Lustre file system and the TAS HSM file system, Versity Storage Manager (VSM) and deletes the HSM object if necessary. The Copytool opens objects by FID and uses `ioctl` calls to report progress to the Lustre MDT node. MDT nodes pass some messages to the coordinator to

update the progress while waiting for the HSM (for example, every *n* seconds), reports error conditions, and current extent.

Policy Engine

A policy engine makes decisions about **which** files should be moved between the Lustre file system and backend HSM, and **when**. The TAS Connector uses Robinhood Policy Engine software and the MariaDB SQL database software as a policy engine. The Robinhood Policy Engine is Open-Source software typically used to monitor and purge large file temporary systems. Robinhood performs all its tasks in parallel, so it also works well for managing large file systems with millions of entries and petabytes of data. It also can monitor usage per Lustre OST and also purge files per OST and/or OST pools. Robinhood refreshes its list of file system entries by scanning the file system it manages, and populating a MariaDB database with this information. It continuously reads MDT changelogs to update its database in soft real-time.

Agent

The TAS Connector migration agent nodes (CMA nodes) manage local TAS HSM requests on a client.

The TAS Connector agent (CMA node) executes file transfers sent by the manager (CMM node). It is usually run as a daemon, started by an `init` script, on a different host than the manager.

TAS Connector Policy Engine

The TAS Connector policy engine makes policy decisions about which files to archive or release on the Lustre file system based on a set of rules, using file attributes, sizes, ownership, etc. Rules can be combined and entries can be whitelisted. The policy engine used on TAS is called Robinhood. It is uses the MariaDB SQL database software, and runs on the CMM nodes in a high-availability configuration.

Manage a TAS System with Bright

Cray Tiered Adaptive Storage (TAS) systems are managed using Bright Cluster Manager (Bright) software.

Refer to the *Bright Cluster Manager 6.1 Administrator Manual* for detailed information about Bright software management. PDF files for the Bright manuals are stored on the CIMS node in the `/cm/shared/docs/cm` directory, and linked to from the `/root` directory.

A CIMS node runs Bright software and provides a system management interface for all the service nodes, RAID controllers, and switches in the system. Bright communicates with each device over the `esmaint-net` maintenance network and the `ipmi-net` IPMI network. The Bright management interfaces are:

- `cmsh` — The Cluster management shell or command line interface.
- `cmgui` — A graphical user interface (GUI)
- `cmd` or CMDaemon — A process that runs on all nodes in the TAS system. CMDaemon on the CIMS node responds to requests from cmsh or cmgui, and communicates with the CMDaemon processes running on each service node. The CMDaemon processes on each service node communicate only with the CMDaemon processes running on the CIMS node.

Either the `cmgui` or `cmsh` can be used to manage the system. There may be certain tasks that are more easily visualized using `cmgui`, and other tasks that are more efficient using `cmsh`.

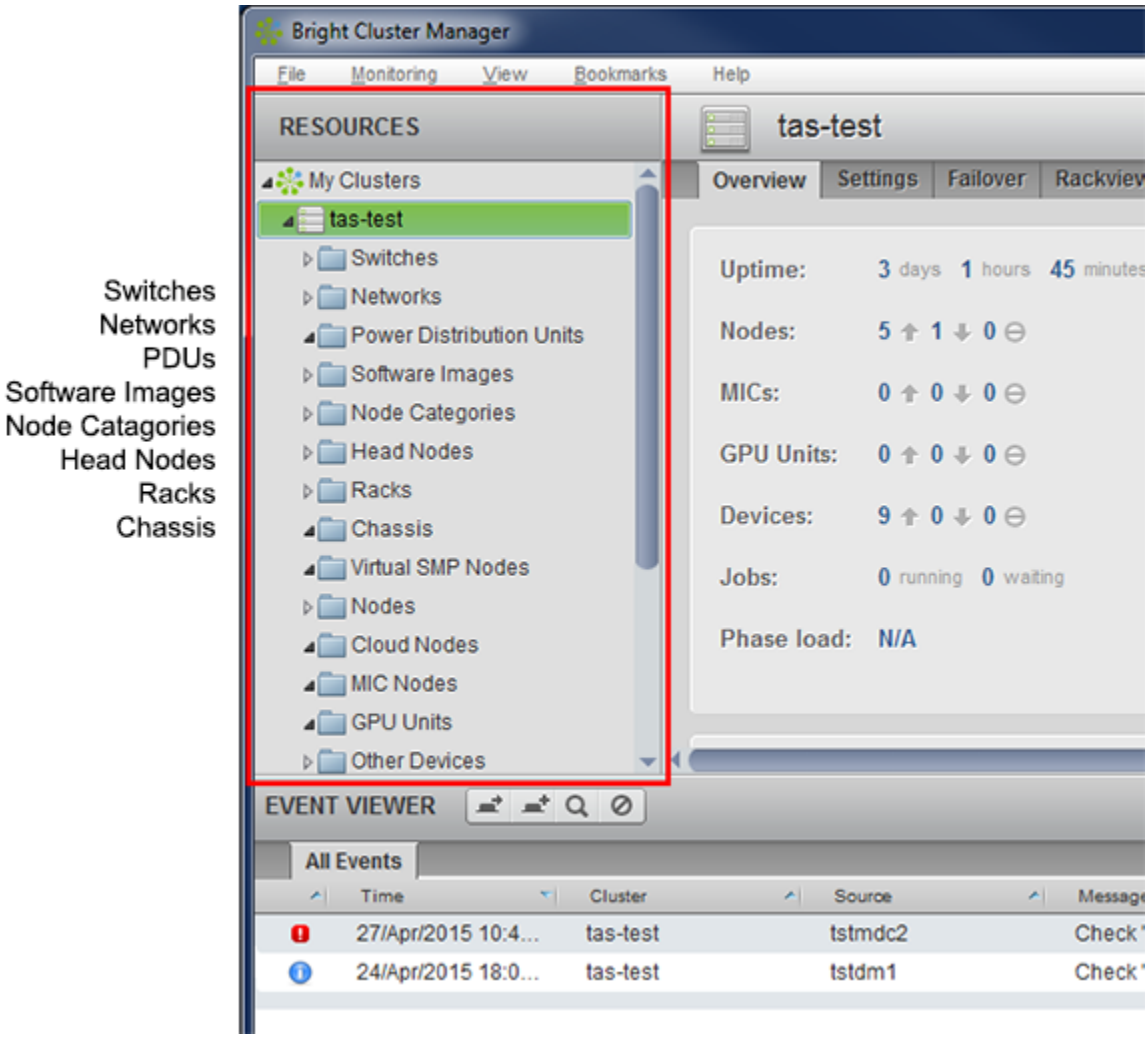
Be aware that Bright software runs a database to manage the system, therefore, modifications to the system are not invoked until they are committed in `cmsh` or saved in `cmgui`. All service nodes PXE boot from software images stored in the `/cm/images` directory on the CIMS node.

> **NOTE:** Cray recommends that software configuration changes to a service node software are made to the software image on the CIMS node using the `chroot` environment, and pushed out to the running node (updating a node). Alternatively, the software image on the running node can be captured or "grabbed" by Bright, and stored on the CIMS node, but only if the Bright exclude lists are configured properly. Grabbing a software image from a running node, could inadvertently add user file systems or other unwanted data to the software image.

Use `cmgui` to Manage TAS

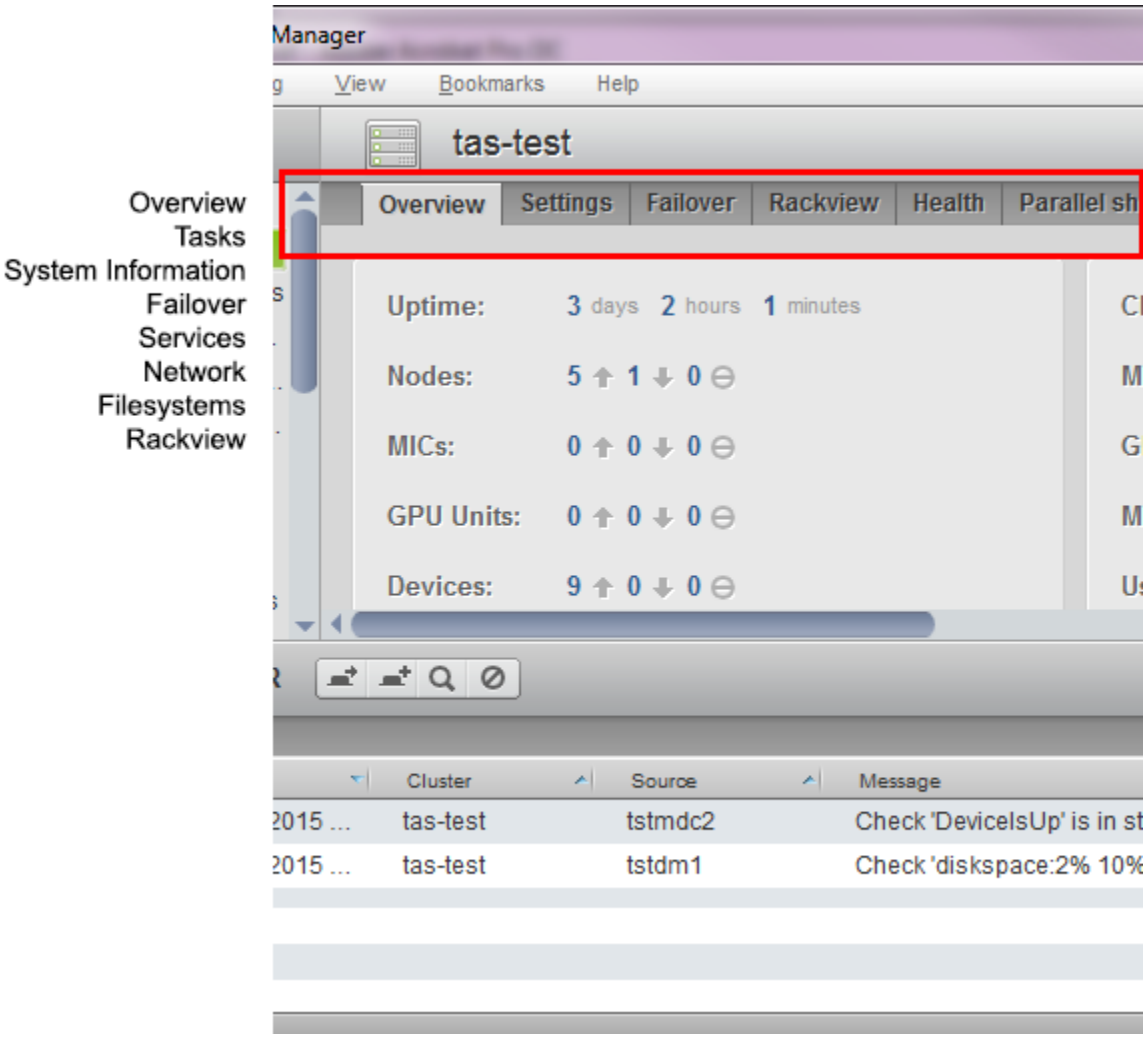
The `cmgui` RESOURCES pane lists all the devices managed by Bright and Bright categories, software images, node groups, and networks. Each of these items can be selected and managed using the `cmgui`. The `cmgui` enables the **Save** button to commit changes to the system.

Figure 6. `cmgui` RESOURCES Pane



When an object is selected in the RESOURCES pane, the tabs at the top of the GUI change. For more information about using `cmgui` and how to perform common administrative tasks for Cray system.

Figure 7. cmgui Tabs



Run CMGUI from the CIMS Node

The `cmgui` program may be run on the CIMS node and displayed to a remote X Window System running on a Linux®, Windows®, or Mac OS® desktop or other platform.

1. On a remote system such as a Linux desktop or PC, start an X-server application such as Xming or Cygwin/X.

2. Enter the following command to log in to the CIMS (in this example, `cims`) with SSH X forwarding.

```
remote% ssh -X root@cims
cims#
```

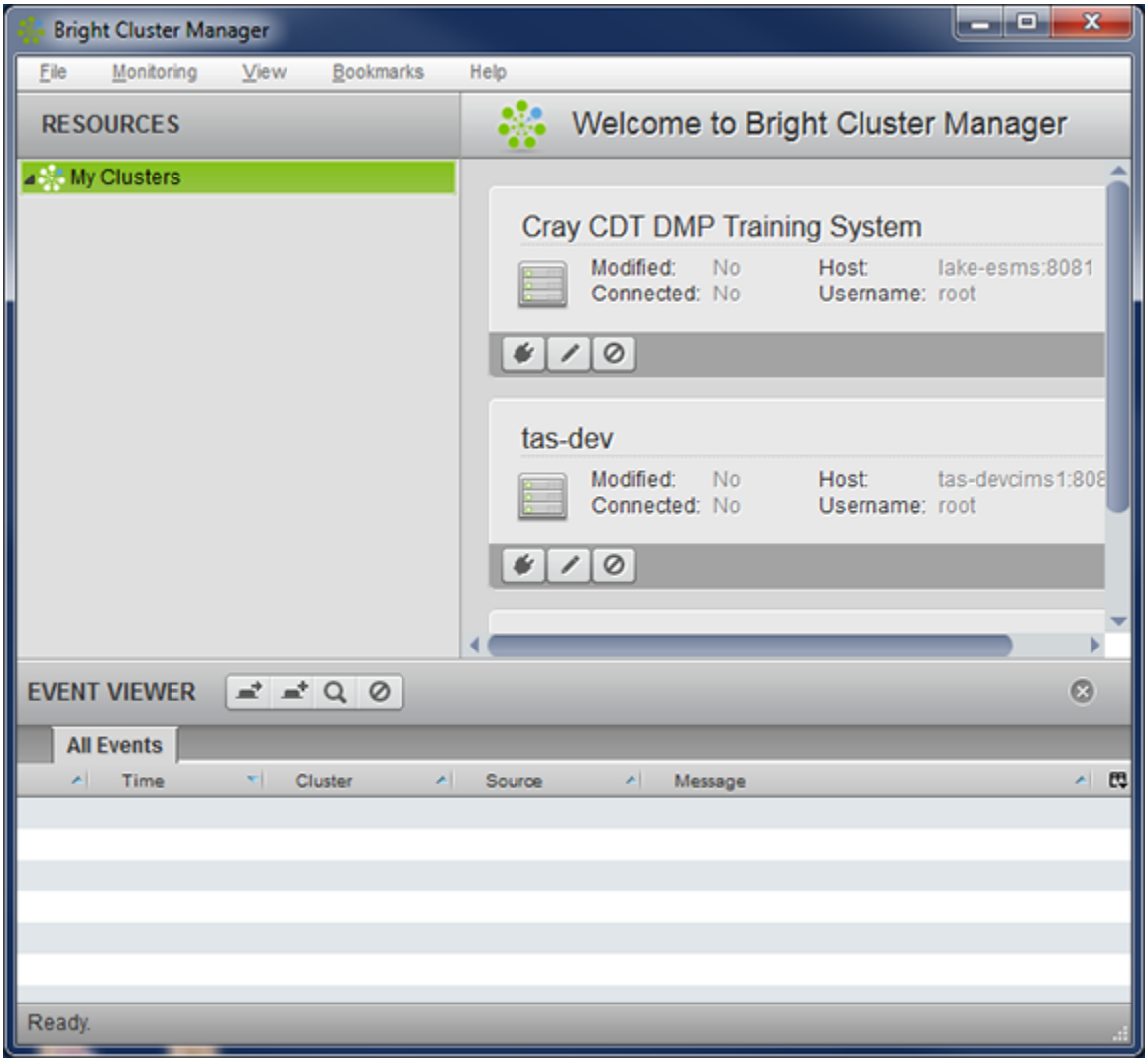
3. Start the `cmgui` program.

```
cims# /cm/shared/apps/cmgui/cmgui &
```

4. Select **Add a new cluster**. Enter the CIMS node hostname, username, and password and click OK.

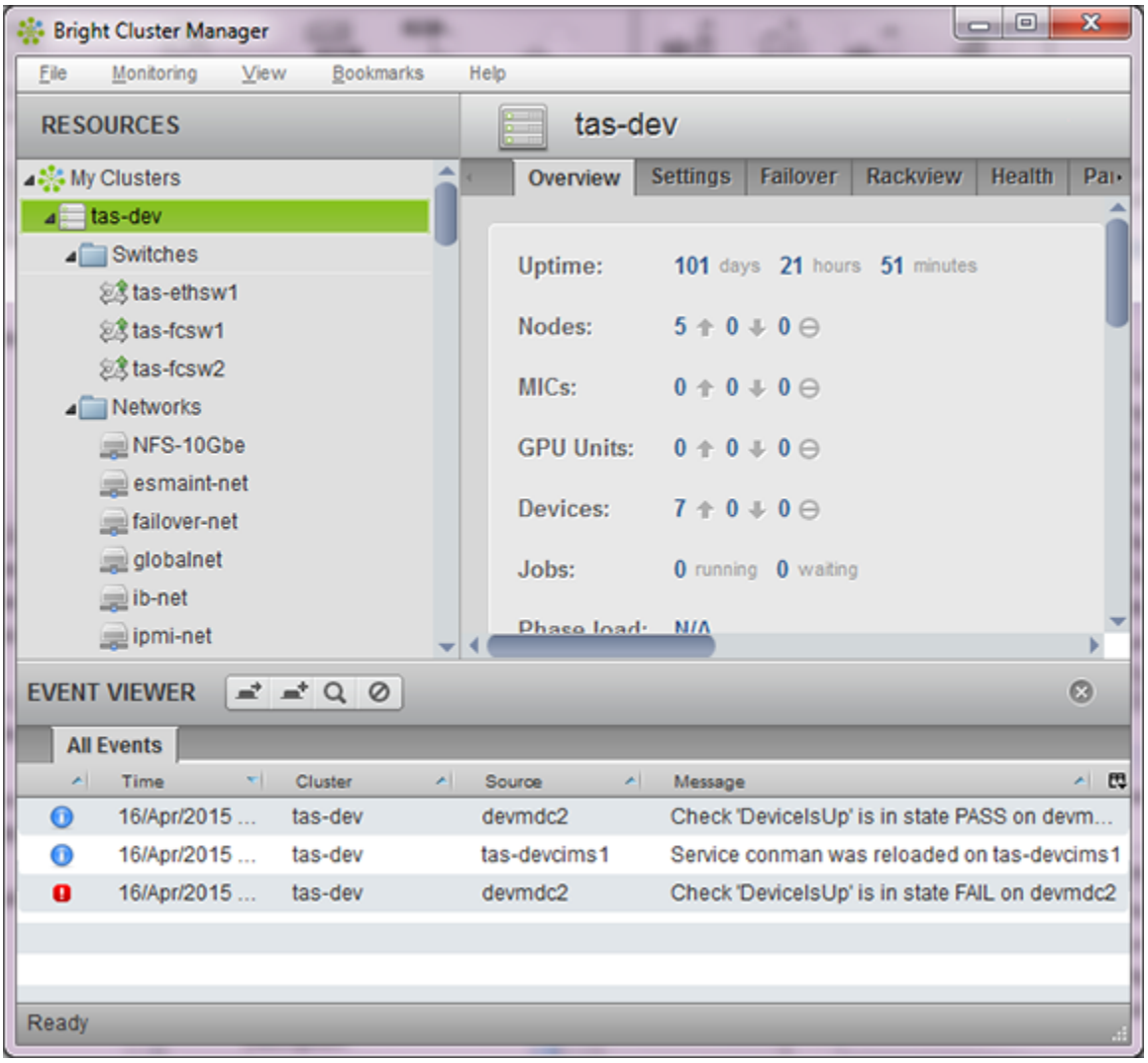
5. Select the power plug icon and enter the system password to connect to the system.

Figure 8. `cmgui` Window



The `cmgui` window displays the system configuration.

Figure 9. cmgui Connect-to-Cluster



Install and Run cmgui on a Remote System

The Bright GUI (cmgui) can be installed on a Linux®, Windows®, or Mac OS® platform and supports a virtual network computing (VNC) server for remote connections.

IMPORTANT: Communication between the remote computer and the CIMS node should be encrypted.

Cray recommends SSH port forwarding or SSH tunneling. When running the cmgui program from the remote computer, cmgui connects to the CIMS node using SSL. Cray also recommends using SSH port forwarding when using VNC. The Linux, Windows, or Mac OS installation software for the cluster management GUI (cmgui) is located in /cm/shared/apps/cmgui/dist on the CIMS node.

IMPORTANT: Whenever you update the CIMS node with the latest ESM software, always reinstall the updated cmgui software on the remote systems.

1. Copy the Windows `.exe` file, (`install.cmgui.6.1.`*`revision`*`.exe`), Linux compressed `TAR` file (`cmgui.6.1.`*`revision`*`.tar.bz2`, or Mac OS package file (`install.cmgui.macosx.6.1.`*`revision`*`.pkg` from the `/cm/shared/apps/cmgui/dist` directory on the CIMS node to a `tmp` directory on the remote system.

   ```
   remote% scp root@tas-cims1:/cm/shared/apps/cmgui/dist/* /tmp
   ```

2. Copy the PFX certificate file (`admin.pfx`) from the `root` directory of the CIMS node to a secure location on the remote system so that it can be used for authentication purposes. Rename the file so that you can identify the system it authorizes (*`systemname`*`-admin.pfx` for example).

   ```
   remote%  scp root@tas-cims1:/root/admin.pfx /securelocation/systemname-admin.pfx
   ```

3. Install the software.

   a. On Windows systems, execute the installer `.exe` file and follow the instructions.

   b. On Linux systems, extract the files using the `tar` command:

      ```
      remote%  tar -xvjf cmgui.6.1-revision.tar.bz2
      ```

   c. On Mac OS systems, click on the `.pkg` file and follow the instructions.

4. Start `cmgui` and select the power plug icon and enter the PFX certificate password to connect to the system.

Bright Node Categories for TAS

Bright software implements the concept of service *node categories* and *node groups* to manage groups of nodes.

Categories specify a number of parameters that are common to all members of the node category such as software image, finalize script, and disk setup. The node installer configures each node's image during provisioning from the CIMS node. A service node must be associated with a single node category. Category parameters can be overridden on a per node basis, if desired, by setting configuration parameters for the node, instead of the node category.

Bright software configures a separate interface for each node because the IP addresses that Bright uses are specific to each node. Software images are common across multiple nodes, so the Bright interface files must reside in the Bright database and be placed on service nodes at boot time.

Service nodes can belong to several different node groups, and there are no parameters associated with node groups. Node groups are typically used to invoke commands across several nodes simultaneously.

Table 3. Bright Node Categories for TAS

Category	Description
`default`	Default category configured by installation software for service nodes. **Do not delete the `default` category**.
`tas-dm`	Default category for data mover (DM) nodes.
`tas-mdc`	Default category for metadata controller (MDC) nodes.
`tas-cmm`	Default category for migration manager (CMM) nodes.
`tas-cma`	Default category for migration agent (CMA) nodes.

Node categories provide control over several node parameters such as:

revision	Object revision.
bmcpassword	Password used to send `ipmi`/`ilo` commands to nodes. The baseboard management controller (BMC or iDRAC) password is inherited from the `base` partition and is not set for the node category in Cray TAS systems.
bmcusername	User name used to send `ipmi`/`ilo` commands to nodes. Inherited from the `base` partition and is not set for the node category in Cray TAS systems.
defaultgateway	Default gateway for the category.
filesystemexports	Configure the entries placed in `/etc/exports`.
filesystemmounts	Configure the entries placed in `/etc/fstab`.
installbootrecord	Install boot record on service node local disk to enable booting without a CIMS node.
installmode	Specifies software install mode. Typically set to `auto`. Can be `auto`, `full`, `main`, or `nosync`.
ipmipowerresetdelay	Delay used for `ipmi`/`ilo` power reset, default is `0`.
managementnetwork	Specifies the network used for management traffic. Always set to `esmaint-net`.
name	Name of category.
nameservers	List of name servers the category will use.
newnodeinstallmode	Default install mode for new nodes. Typically set to `full`.
roles	Assign the roles the node should play.
searchdomain	Search domains for the category.
services	Manage operating system services.
softwareimage	Software image the category will use.
timeservers	List of time servers the category will use.
usernodelogin	`ALWAYS` or `NEVER` allow a user to log in to the node.
disksetup	Disk setup for nodes.
excludelistfullinstall	Exclude list for full install.
excludelistgrab	Exclude list for grabbing the image running on the node to an existing image.
excludelistgrabnew	Exclude list for grabbing to a new image.

excludelistsyncinstall	Exclude list for a sync install. Specifies what files and directories to exclude from consideration when copying parts of the filesystem from a known good software image to the node.
excludelistupdate	Exclude list for updating a running node.
finalizescript	Finalize script for category.
initializescript	Initialize script for category.
notes	Administrator notes.

Bright Node Groups for TAS

Node groups simplify management and control activities and enable administrators to perform commands on a group of nodes simultaneously. Typical node groups are listed below:

Table 4. TAS Node Groups

Node Group	Description
`DM`	All DM service nodes
`MDC`	All MDC service nodes
`CMA`	All CMA service nodes
`CMM`	All CMM service nodes

Bright Device Names for TAS Systems

A *device* in a TAS system represents a physical hardware component. A device can be any of the following types:

- CIMS nodes - Typically named `tas-cims1`, `tas-cims2`.
- Service nodes - Typically named according to their function in the system, such as `tas-mdc1`, `tas-mdc2` (metadata controllers), `tas-dm1`, `tas-dm2` (data movers). TAS Connector migration manager nodes may be named `tas-cmm`, and migration agent nodes, `tas-cma`.
- Ethernet switches - Typically named according to their function in the system such as `tas-ethsw1`.
- InfiniBand™ switches - Named similar to `tas-ibsw1` and `tas-ibsw2`.
- Fibre Channel switches - Named similar to `tas-fcsw1` and `tas-fcsw2`.
- Storage array RAID controllers - Storage controllers are added as a generic device in `cmsh` or under the Other resource in the `cmgui`. Typically named by manufacturer model number, controller, rack location, or purpose such as `tas-E5560a`, `tas-E5560b`, `tas-E2700a`, `tas-E2700b`.

Bright License Management

Cray systems using Bright are configured with a Bright Cluster Manager (Bright) license file installed on the CIMS node. The license file includes:

- A `licensee` attribute or the name of the organization, the condition in which the specified organization may use the software; a licensed nodes attribute specifies the maximum number of nodes that the cluster manager may manage. CIMS nodes are also regarded as nodes too for this attribute.
- `licensed nodes` attribute specifies the maximum number of nodes that the cluster manager may manage. CIMS nodes are included.
- An `expiration date` for the license.

  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. The license file is the X509v3 certificate for the CIMS node and is used throughout system operations.

Use the `cmsh` mode, `main licenseinfo` command, to display the Bright license information from the CIMS node.

```
cims# cmsh
[cims]% main licenseinfo
License Information
---------------------- ------------------------------------------
Licensee /C=US/ST=California/L=San Jose/O=Bright
Computing/OU=Temporary Licensing/CN=003040
Serial Number 4316
Start Time Tue Mar 24 00:00:00 2015
End Time Mon Mar 31 23:59:59 2015
Version 6.1
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
```

Bright License Verification

The `verify-license` utility determines whether a license is valid even when the Bright management daemon (CMDaemon) is not running.

Verify a license:

```
cims# /etc/init.d/cmd start
Waiting for CMDaemon to start...
CMDaemon failed to start please see log file.

cims# tail -1 /var/log/cmdaemon
Mar 30 15:57:02 cims CMDaemon: Fatal: License has expired
```

Install the Bright License on a CIMS Node

Prerequisites

Obtain a product key from Cray which enables an administrator to obtain and activate a license. The product key looks like the following string:

```
XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX
```

The initial installation of Bright is licensed only for three nodes, including the CIMS node. A permanent license must be installed to configure the system.

- Install the Bright license on a CIMS node to install the license.
- If your existing license has expired, then reinstate your license.

Certificate Sign Request (CSR) data is displayed and saved in the file `/cm/local/apps/cmd/etc/cert.csr.new`. The `cert.csr.new` file may be used with an internet-connected browser. After using a product key with `request-license`, reboot the system using the `pexec reboot` command from the CIMS node. A reboot is not required when relicensing an existing system.

1. Log in to the CIMS node as `root`.

```
remote% ssh root@cims1
```

2. Get the MAC address (`HWaddr`) for `eth0` (`BOOTIF`) interface.

```
cims1# /sbin/ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 78:2B:CB:40:CE:CA
          inet addr:10.141.255.254  Bcast:10.141.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11944661 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11018308 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2589377379 (2469.4 Mb)  TX bytes:2060383201 (1964.9 Mb)
          Interrupt:36 Memory:d6000000-d6012800
```

3. Run the `request-license` command on the CIMS node. A prompt to reuse the private key and settings from the existing license displays if the existing license is valid.

> **IMPORTANT:** When configuring an HA system, enter license information for the primary CIMS node only. Do not enter the MAC address for the secondary CIMS node. Run this procedure again if configuring the secondary CIMS node and enter **`yes`** when prompted.

```
cims1# request-license
Product Key (XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX):ProductKey
```

4. Enter the product key, then press Enter. Enter `no` at the prompt.

```
Re-use private key and settings from existing license? [Y/n] yes
```

a. Enter the site information for the Bright license when prompted:

```
Country Name (2 letter code): CountryName
State or Province Name (full name): StateProvince
Locality Name (e.g. city): City
Organization Name (e.g. company): Company
Organizational Unit Name (e.g. department): Department
Cluster Name: ClusterName
```

b. Enter the MAC address of the CIMS node for `eth0` on (`esmaint-net`).

```
MAC Address of primary head node (cims1) for eth0  []: FF:AE:FF:B5:E2:64
```

Enter `no` at the following prompt.

```
Will this cluster use a high-availability setup with 2 head nodes? [y/N]: no
```

c. Answer **`no`** when prompted to submit the certificate request:

```
Certificate request data saved to /cm/local/apps/cmd/etc/cert.csr.new
Submit certificate request to http://support.brightcomputing.com/licensing/index.cgi ? [Y/n] no
```

The CSR (Certificate Sign Request) data is displayed and saved in the file `/cm/local/apps/cmd/etc/cert.csr.new` on the CIMS node. The `cert.csr.new` file may be used to obtain a license with an Internet-connected web browser. (The license string below is fictitious.)

```
-----BEGIN CERTIFICATE REQUEST-----
MIICBjCCAW8CAQAwgcUxCzAJBgNVBAYTAlVTMRIwEAYDVQQIEwlXaXNjb25zaW4x
FzAVBgNVBAcTDkNoaXBwZXdhIEZhbGxzMRIwEAYDVQQKEwlDcmF5IEluYy4xDDAK
BgNVBAsTA0JJUzESMBAGA1UEAxMJaHVzayBlc01TMSAwHgYJKoZIhvcNAQkCExFF
MDpEQjo1NTowODpGRjpDMDExMC8GCSqGSIb3DQEJBhMiMDExNDI2LTUxMTQ3Mi0w
MDI3NDYtODU3MzM2LTQxNTYyNDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA
scCs7/hIZF5ehPq0ZhGn/bVY8cO+e9KF8psJHu1cYVC1WCcFj04LMQztUVvftigI
HWo+YZVJbuMHphvAc4BfXDhYxjLPVw+yxU9FBBBDyFZxuMJpCIhr8YAKxABVX0fS
zKK6eE7Pj1G6Ho9vW6+sHOgzCF3jm4xG52NTTma+BQUCAwEAAaAAMA0GCSqGSIb3
DQEBBQUAA4GBAG6VBE0HRqSKP8CFAaJ3AwewtXEL7gotOYBAhfe2rMvl6/NWzFGD
uCCju5psN5LpsgyhKQTWPDQwS7EbxRQ+jerHVcsI/ZEgnzVBozjvVgESVML8+yA0
6Dtba8hrqBFFtLXmm3KE+qQCt+vqGMUFs8g4D0GYOkThlg6auJFXFU3N
-----END CERTIFICATE REQUEST-----
```

5. On a system with Internet access, use a web browser to open *http://support.brightcomputing.com/licensing/*.

6. Copy CSR data obtained in to obtain a license certificate.

7. Paste the CSR data (contents of `cert.csr.new`) into the web form, then select **Submit**.

   A signed license certificate is displayed (the following example is fictitious).

```
-----BEGIN CERTIFICATE-----
MIIDNzCCAh+gAwIBAgICFkwwDQYJKoZIhvcNAQEFBQAwbDELMAkGA1UEBhMCVVM
CzAJBgNVBAgTAkNBMREwDwYDVQQHEwhTYW4gSm9zZTEfMB0GA1UEChMWQnJpZ2h
IENvbXB1dGluZywgSW5jLjEcMBoGA1UEAxMTQnJpZ2h0IENvbXB1dGluZyBDQTA
Fw0xMjA5MTIwMDAwMDBaFw0zODEyMzEyMzU5MDBaMGYxCzAJBgNVBAYTAlVTMRQ
EgYDVQQIEwtNaXNzaXNzaXBwaTESMBAGA1UEBxMJVmlja3NidXJnMREwDwYDVQQ
EwhDcmF5IEluYzENMAsGA1UECxMEc3RjbzELMAkGA1UEAxMCbmEwgZ8wDQYJKoZ
hvcNAQEBBQADgY0AMIGJAoGBAN8lCM52TEnZ63yvxPvpe4WbBTPsFKUWOpIOHT8
tbctYf54E4K4A1A0ahX48OYdhafhTb7AO0gGSv/Vp+QxZQrkrSi6A8zwlNkrz4j
yDDYFNb4sBkJUbexHmEdR5Bhp/xfEx4X7EkFb5vdnhIyiwvn6Zs5tZ/Sgt+CJcH
KJFtAgMBAAGjbTBrMA8GA1UdEwEB/wQFMAMBAf8wHgYHKwYBBKFkIQQTFhE4NDo
Rjo2OTpFMzo1Qjo1ODAPBgcrBgEEoWQiBAQWAjE2MBAGBysGAQShZCMEBRYDNS4
MBUGBysGAQShZCQEChYIQWR2YW5jZWQwDQYJKoZIhvcNAQEFBQADggEBAF1pRtg
vjMr9TlihmEcO23raTLp308zkVFpW7vJ0T8KqEirwkzzrD83igtJNd2q6jtrpRL
3kxQ2sB0gGmuptHkrYecwZtVm5FhGOuPeDUG3ww4W+GyCkczCRtPkVTXul250Z9
LZqrK0zzPRKhMNEraTXPDEgHSEgEeykro30EGHpcuCoGCBJNvhi0bCIxgJoW5DV
ykGbeE4DKBlyW0r4NRqMBR+0BH2d1gCXBJtYhHMfDUWw6JOMmsCcoAY7Yhp4N9k
AJb++bi/pO2fQeDJopfxvlU2WsEEMcEItNklknaHlOYfQ3IcloH9w464MtGFakt
xGjtfqxcU
-----END CERTIFICATE-----
```

8. Copy the license text received and save it to a plain text file named *signedlicensefile* on the CIMS node.

9. Enter the following command to install the license and answer each prompt.

```
cims1# install-license signedlicensefile
========== Certificate Information ========
Version:                 6.1
Edition:                 Advanced
Common name:             na
Organization:            ACME
Organizational unit:     Training
Locality:                City
State:                   State
Country:                 US
Serial:                  3728
Starting date:           12 Mar 2015
Expiration date:         31 Dec 2038
MAC address:             FF:AE:FF:B5:E2:64
Pre-paid nodes:          16
Max Pay-as-you-go Nodes: N/A
============================================

Is the license information correct ? [Y/n] Y
Backup directory of old license:
/var/spool/cmd/backup/certificates/2013-01-28_08.56.22Installed new license
```

```
Waiting for CMDaemon to stop: OK
Installing admin certificates

Waiting for CMDaemon to start: OK

New license was installed. In order to allow nodes to obtain a new
node certificate, all nodes must be rebooted.

Please issue the following command to reboot all nodes:
        pexec reboot
```

Protect the `admin.pfx` file in the `/root` directory. This file contains the administrator certificate which grants full access to the system. Copy the `admin.pfx` file to the remote system to runs the `cmgui` and gain access to the system.

```
Please provide a password that will be used to password-protect the PFX file
holding the administrator certificate (/root/.cm/cmgui/admin.pfx).

Password:
Verify password:
```

10. If the license process fails, check `/var/log/cmdaemon` for failure information.

Bright License Renewal

1. Log in to the CIMS as `root`.

```
# ssh root@cims1
```

2. Run the `request-license` command on the CIMS node.

```
cims1# request-license
Product Key (XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX):
```

3. Enter the product key, then press Enter (the example is not a valid key).

```
714354-916786-132324-207440-186713
```

4. Answer each prompt to reinstall the license.

```
Existing license was found:
  ...
Re-use private key and settings from existing license? [Y/n] y

Will this cluster use a high-availability setup with 2 head nodes? [y/N] n
MAC Address of primary head node for eth0 []: 7A:2B:3B:40:0E:CA

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

Contacting http://support.brightcomputing.com/licensing/index.cgi...

License granted.
License data was saved to /cm/local/apps/cmd/etc/cert.pem.new
Install license? [Y/n] y
========= Certificate Information ========
Version:                6.0
Edition:                Advanced
Common name:            Training
Organization:           ACME Training
Organizational unit:    Training and Doc
Locality:               Chippewa Falls
State:                  Wisconsin
Country:                US
Serial:                 5846
```

```
Starting date:            01 May 2014
Expiration date:          01 May 2015
MAC address:              78:2B:CB:40:CE:CA
Pre-paid nodes:           512
Max Pay-as-you-go Nodes: 1000
========================================

Is the license information correct ? [Y/n] y directory of old license: /var/spool/cmd/backup/
certificates/2013-02-10_11.34.53
Is this host the cluster's head node? [Y/n] y
Installed new license

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

Bright Default Objects and Scripts

CAUTION:

- Default software objects.
- Do not delete or modify the default objects in Bright.

Initial software images, the default node, and initial Bright categories must not be deleted or modified. These objects are cloned to make other customized objects in Bright:

node001 The default node (`node001`). The default node is cloned to create other customized TAS nodes.

/opt/cray/esms Default administrative scripts. The default scripts in this directory are copied from `./default` to `./etc` so that software updates do not overwrite site customizations.

TAS Connector Software Image Management

A TAS connector software image is a blueprint for the contents of the local file systems on service nodes. Software images reside in `/cm/images` on the CIMS and contain a full Linux™ file system and other customizations. Software images are typically named with prefixes such as `ESF` or `TAS` and include the release version and date.

The software images shipped with the system (for example, `TAS-XX-2.0.0-201501070415-cmm` for CMM nodes or `TAS-XX-2.0.0-201501070415-cma` for CMA), have been configured with a base CentOS 6.5 operating system, VSM software, and TAS software tools and utilities by Cray manufacturing and should not be modified.

The software images shipped with the system, `TAS-XX-2.0.0-201501070415-cmm` for CMM nodes or `TAS-XX-2.0.0-201501070415-cma` for CMA nodes, for example, have been configured with a base CentOS 6.5 operating system, VSM software, and TAS software tools and utilities by Cray manufacturing and should not be modified.

Always backup the factory configured software images. Always clone the factory configured software image to a new software image and make modifications to the new image. All software images in the `/cm/images` directory on the CIMS node should be backed-up regularly and clearly named for the site. Note that `default-image` and `default-image.previous` (created when the CIMS node software is updated) are default service node

images created by Cray CIMS node installer software (`ESMinstall`) and use the SLES11SP3 operating system. Do not clone these images to create TAS service node images.

When a node boots, the node provisioning system configures the node with a copy of the software image specific to the node category. Software images are assigned to a node `category` in Bright. Nodes are then assigned to a node `category` to configure their operation and behavior. This enables administrators to have flexibility in controlling specific software images for various node types. Use the Bright `cmsh` `clone` command to create a copy of the software image and configure it for testing using the Bright.

Software images can be modified in the `/cm/images` directory using the `chroot` environment on the CIMS node. This is the recommended method, because it avoids the problem of copying unwanted parts of the software image.

Grabbing a running software image from a node to a software image file can be problematic if the Bright exclude lists are not configured properly.

The Bright `excludeslistgrab` and `excludelistgrabnew` exclude lists must be configured so that user, or other shared files systems are not copied to the software image.

Cray recommends that administrators first clone an image using Bright, then modify the cloned software image and test it on a single node. If the configuration changes are satisfactory, then update the node category or node group to minimize user interruptions.

Grabbing a softwareimage from a running node may be a preferable method, as long as the Bright exclude lists are configured properly.

TAS CMM Node Category

The TAS Connector CMM node category (`tas-cmm` for example) is configured by Cray manufacturing to boot and configure CMM nodes. The `tas-cmm` category in Bright is configured with the CMM node software image (`TAS-XX-2.0.0-201501070415-cmm` for example).

This software image contains the CentOS 6.5 operating system and specific Cray customizations for TAS Connector CMM nodes. A customized category *exclude list* is configured for this category to control which files are updated or left untouched when this software image on the node is sync'd with the CIMS node software image.

The CMM node category configures file system mount points such as `/vsm/tasfs1` and the administration file system `/tas_admin`). Other shared file systems are mounted from the CIMS node.

Use the CMM node category to make configuration changes to all the CMM nodes at once. After configuration changes made to a category, commit the changes and reboot or update the node to invoke the change.

TAS CMA Node Category

The TAS Connector CMA node category (`tas-cma` for example) is configured by Cray manufacturing to boot and configure CMA nodes. The `tas-cma` category in Bright is configured with the CMA node software image (`TAS-XX-2.0.0-201501070415-cma` for example).

This software image contains the CentOS 6.5 operating system, and specific Cray customizations for TAS Connector CMA nodes. A customize category *exclude list* is configured for this category, which determines which files are updated or left untouched when sync'd with the software image on the CIMS node.

The CMA node category configures file system mount points such as `/vsm/tasfs1` and the Lustre® file system mount point). Other shared file systems are mounted from the CIMS node.

Use the CMA node category to make configuration changes to all the CMA nodes at once. After configuration changes are made to a category, commit the changes, and reboot or update the node to invoke the change.

Change TAS Node Category Settings

Prerequisites

Bright software is operational.

Node categories group similar nodes together so that they can load a specific software image when the nodes PXE boot. Bright enables administrators to make changes to either the software image or to the node category.

1. Log in to the CIMS node as `root` and start `cmsh`.

```
cims#  cmsh
[cims]%
```

2. Switch to `category` mode and list the Bright categories configured for the system.

```
cims#  category
[cims->category]% list
Type   Name (key)               Software image
------ ------------------------ ------------------------
Node   default                  default-image
Node   tas-cmm                  TAS-XX-2.0.0-201501070415-CMM
Node   tas-cmm-6.5              TAS-XX-2.0.0-rc1
Node   tas-cma                  TAS-XX-2.0.0-201501070415-CMA
```

   The `tas-cmm` category loads the `TAS-XX-2.0.0-201501070415-CMM` software image from the CIMS node `/cm/images` directory.

3. To display the settings for a category, use the `category show` subcommand from `category` mode.

```
[cims->category]% show tas-cmm

[tas-devcims1->category]% show tas-mdc
Parameter                        Value
-------------------------------- ------------------------------------------------
BMC Password                     < not set >
BMC User ID                      -1
BMC User name
Default gateway                  172.30.86.1
Disk setup                       <2436 bytes>
Exclude list full install        <312 bytes>
Exclude list grab                <1024 bytes>
Exclude list grab new            <1024 bytes>
Exclude list sync install        <1441 bytes>
Exclude list update              <2892 bytes>
Filesystem exports               <0 in submode>
Filesystem mounts                <8 in submode>
Finalize script                  <4162 bytes>
GPU Settings                     <0 in submode>
Initialize script                <0 bytes>
Install boot record              yes
Install mode                     AUTO
Ipmi power reset delay           0
Management network               esmaint-net
Name                             tas-mdc
Name servers
New node install mode            FULL
Notes                            <0 bytes>
Provisioning associations        <1 internally used>
Require FULL Install Confirmation  no
Revision
Roles                            <0 in submode>
```

```
Scaling governor
Search domain
Services                          <0 in submode>
Software image                    TAS-XX-2.0.0-201501070415-CMM
Time servers
Type                              Node
User node login                   ALWAYS
```

4. Enter the use command to set the category. Then use the `get defaultgateway` subcommand from `category` mode to list all the parameters configured for that category.

```
[cims->category]% use tas-cmm
[cims->category[tas-cmm]% get softwareimage
TAS-XX-2.0.0-201501070415-CMM
```

TIP: Press the `Tab` key to display a list of valid subcommands whenever working in `cmsh`.

5. To configure a new software image for the `tas-cmm` category for example, use the `set` command.

```
[cims->category[tas-cmm]% set softwareimage TAS-XX-2.0.0-201501070415-CMM_Test
[cims->category*[tas-cmm*]%
```

IMPORTANT: The asterisk (`*`) symbols on the `cmsh` command line indicates that the current changes have not been saved (committed) into the Bright database.

6. Commit the changes to the Bright database so that the `tas-cmm` category loads new software image.

```
[cims->category*[tas-cmm*]% commit
[cims->category[tas-cmm]%
```

7. Reboot the node to load the new software image.

TAS Software Image Properties

The `cmgui` lists the system software images in the **Software Images** section of the **RESOURCES** pane. Use `cmsh softwareimage` mode, and enter `list` to list the software images on the system.

To select an image and show its properties, first `use` the software image, then enter the `show` command to display its properties from `cmsh softwareimage` mode.

To display a specific property such as the software image kernel parameters, enter `get` from `cmsh softwareimage` mode, (press the `Tab` key to display a list of valid options) and provide the `kernelparameters` option as shown.

```
tas-cims1# cmsh
[tas-cims1]% softwareimage
[tas-cims1->softwareimage]% list
Name (key)                       Path                                        Kernel version
-------------------------------  ------------------------------------------  ----------------------
ESF-XX-2.2.0-201510252041        /cm/images/ESF-XX-2.2.0-201510252041        2.6.32-358.6.1.el6.x86_64
TAS-XX-2.0.0-201501070415-dm     /cm/images/TAS-XX-2.0.0-201501070415-dm     2.6.32-358.6.1.el6.x86_64
TAS-XX-2.0.0-201501070415-mdc    /cm/images/TAS-XX-2.0.0-201501070415-mdc    2.6.32-358.6.1.el6.x86_64
TAS-XX-2.0.0-201501070415-rev1-dm  /cm/images/TAS-XX-2.0.0-201501070415-rev1-dm  2.6.32-358.6.1.el6.x86_64
TAS-XX-2.0.0-201501070415-rev1-mdc  /cm/images/TAS-XX-2.0.0-201501070415-rev1-mdc  2.6.32-358.6.1.el6.x86_64
default-image                    /cm/images/default-image                    3.0.80-0.5-default
```

Show TAS Software Image Properties

```
[tas-cims1->softwareimage]% use TAS-XX-2.0.0-201501070415-rev1-dm
[tas-cims1->softwareimage[TAS-XX-2.0.0-201501070415-rev1-dm]]% show
Parameter                      Value
------------------------------ ------------------------------------------------
Boot FSPart                    98784247918
Creation time                  Mon, 05 March 2015 09:15:51 CDT
Enable SOL                     yes
FSPart                         98784247918
```

```
Kernel modules                  <34 in submode>
Kernel parameters               rdloaddriver=scsi_dh_rdac, pci=bfsort
Kernel version                  2.6.32-358.6.1.el6.x86_64
Locked                          no
Name                            TAS-XX-2.0.0-201501070415-rev1-dm
Notes                           <19 bytes>
Path                            /cm/images/TAS-XX-2.0.0-201501070415-rev1-dm
Revision
SOL Flow Control                no
SOL Port                        ttyS1
SOL Speed                       115200
```

Show TAS Software Image Kernel Parameters

```
[tas-cims1->softwareimage[TAS-XX-2.0.0-201501070415-rev1-dm]]% get kernelparameters
rdloaddriver=scsi_dh_rdac, pci=bfsort%
```

Clone a TAS Connector Software Image

Always clone software images to a test image and make modifications or install updates on the test software image before you deploy it on a larger scale. This method avoids corrupting a production software image, initial CMM or CMA node software images provided with the system, or software image updates created by update software.

This procedure clones a service node software image (`TAS-XX-2.0.0-201501070415-CMM`) to a new software image named `TAS-XX-2.0.0-201501070415-CMM_Test`.

1. Log into the CIMS as `root` and run the `cmsh` command.

```
remote% ssh root@esms1
cims# cmsh
[cims]%
```

2. Enter `softwareimage` mode and list the available images.

```
[cims]% softwareimage
[cims1->softwareimage]% list
Name (key)                      Path                                          Kernel version
------------------------------  --------------------------------------------  --------------------------
ESF-XX-2.2.0-201404151111       /cm/images/ESF-XX-2.1.0-201310252041          2.6.32-358.18.1.el6.x86_64
TAS-XX-2.0.0-201501070415-dm    /cm/images/TAS-XX-2.0.0-201501070415-dm       2.6.32-358.18.1.el6.x86_64
TAS-XX-2.0.0-201501070415-CMM   /cm/images/TAS-XX-2.0.0-201501070415-CMM      2.6.32-358.18.1.el6.x86_64
TAS-XX-2.0.0-201501070415-rev1-dm  /cm/images/TAS-XX-2.0.0-201501070415-rev1-dm  2.6.32-358.18.1.el6.x86_64
TAS-XX-2.0.0-201501070415-mdc   /cm/images/TAS-XX-2.0.0-201501070415-rev1-mdc  2.6.32-358.18.1.el6.x86_64
default-image                   /cm/images/default-image                      3.0.80-0.5-default
```

3. Create a new software image named `TAS-XX-2.0.0-201501070415-CMM_Test` by cloning `TAS-XX-2.0.0-201501070415-CMM`.

   IMPORTANT: The time to clone a software image using Bright depends on the image size. Cloning a minimal image (operating system only) completes in 5 to 10 minutes. A fully configured image can take longer. The Bright clone operation spawns a background process that does not prevent you from rebooting a node or performing other configuration changes to software image before the image is fully cloned. Cray recommends that you copy the software image from the UNIX prompt in the `/cm/images` directory on the CIMS node to a new image name, wait for the prompt to return, then clone the image using Bright. Because the new image structure already exists, the clone operation in Bright occurs instantly (only the image attributes in the database are cloned).

4. Exit `cmsh` and copy `TAS-XX-2.0.0-201501070415-CMM` to `TAS-XX-2.0.0-201501070415-CMM_Test` from the UNIX prompt.

```
[cims1->softwareimage]% quit
cims# cp -pr /cm/images/TAS-XX-2.0.0-201501070415-CMM /cm/images/TAS-XX-2.0.0-201501070415-CMM_Test
cims# cmsh
cims% softwareimage
[cims->softwareimage]% clone TAS-XX-2.0.0-201501070415-dm TAS-XX-2.0.0-201501070415-CMM_Test
[cims->softwareimage*[TAS-XX-2.0.0-201501070415-CMM_Test*]]% commit
[cims->softwareimage[TAS-XX-2.0.0-201501070415-CMM_Test
```

5. Display all the software images.

```
[cims->softwareimage[TAS-XX-2.0.0-201501070415-CMM_Test]]% list
Name (key)                          Path                                          Kernel version
----------------------------------  --------------------------------------------  -------------------------
ESF-XX-2.2.0-201502252041           /cm/images/ESF-XX-2.2.0-201502252041          2.6.32-358.6.1.el6.x86_64
TAS-XX-2.0.0-201501070415-dm        /cm/images/TAS-XX-2.0.0-201501070415-dm       2.6.32-504.3.3.el6.x86_64
TAS-XX-2.0.0-201501070415-CMM_Test  /cm/images/TAS-XX-2.0.0-201501070415-CMM_Test 2.6.32-504.3.3.el6.x86_64
TAS-XX-2.0.0-201501070415-mdc       /cm/images/TAS-XX-2.0.0-201501070415-mdc      2.6.32-504.3.3.el6.x86_64
TAS-XX-2.0.0-201501070415-rev1-dm   /cm/images/TAS-XX-2.0.0-201501070415-rev1-dm  2.6.32-504.3.3.el6.x86_64
TAS-XX-2.0.0-201501070415-rev1-mdc  /cm/images/TAS-XX-2.0.0-201501070415-rev1-mdc 2.6.32-504.3.3.el6.x86_64
default-image                       /cm/images/default-image                       3.0.80-0.5-default
```

6. Exit `cmsh`.

```
[cims->softwareimage[TAS-XX-2.0.0-201501070415-CMM_Test]]% quit
cims#
```

Enable Boot Record in Software Image

Prerequisites

The service node must be fully configured in Bright.

Enable the boot record setting for software images so that service nodes can boot from their local hard drive if the CIMS node is not available.

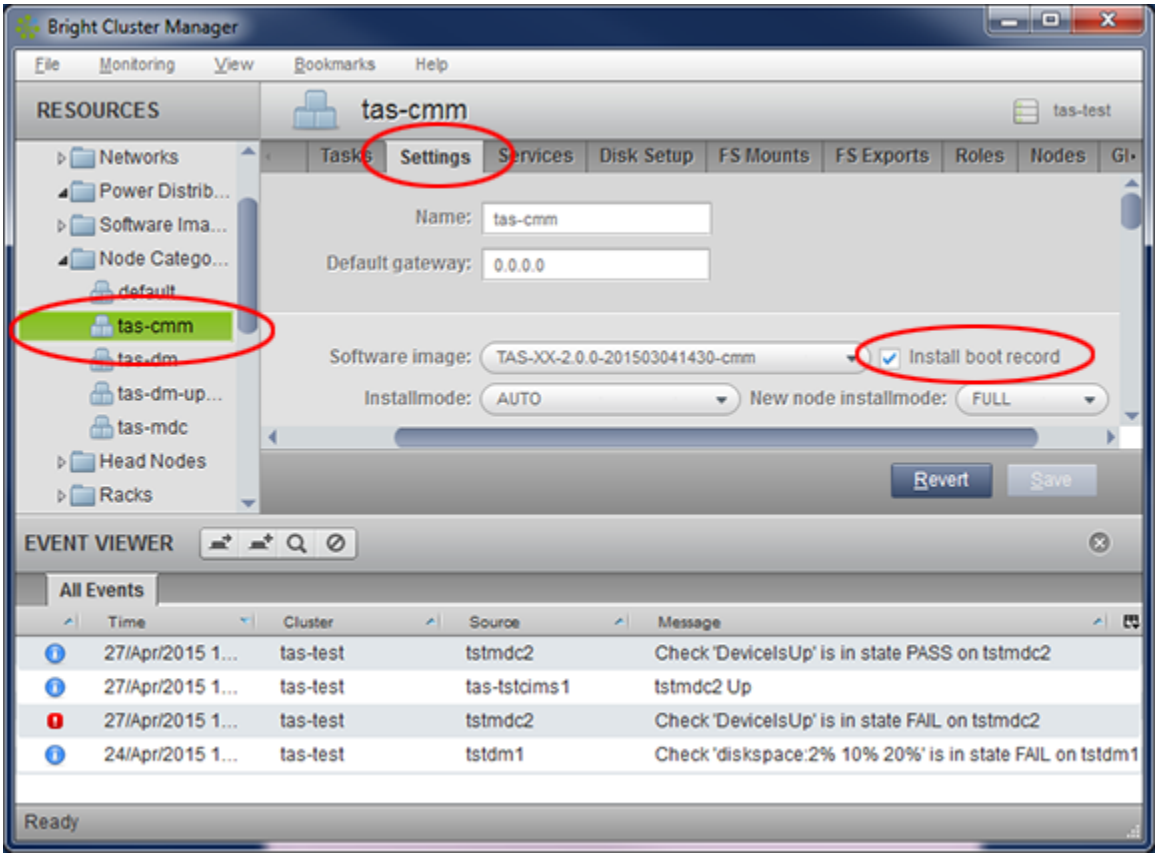
Set service node software images configured in Bright so that the node-installer installs a boot record on the local drive. This enables service nodes to reboot without access to the CIMS node.

Set the `installbootrecord` property for the service node settings and service node category to `on` so that the node can boot from the local drive. Booting from the hard drive must be set to a higher priority than network (PXE) booting in the BIOS settings for the node. Otherwise PXE booting occurs, despite the value set for `installbootrecord`.

Using `cmgui`:

1. Using `cmgui`, select the service node category or device in the **RESOURCES** pane.
2. Select the **Settings** tab for the category or device.
3. Set the GRUB bootloader `Install boot record` checkbox and save the setting in the service node device or category as shown in the figure.
4. Unplug the cable for the `esmaint-net` network (the provisioning network or BOOTIF network) on the service node and reboot the node to determine if it can boot successfully without access to the CIMS node.

Figure 10. Install Boot Record



Using `cmsh`:

5. Log in to the CIMS node as `root`.

6. Enable the `installbootrecord` setting for either the device, or category:

   - To set the `installbootrecord` setting for a single service node, enter:

     ```
     cims1 # cmsh -c "device use device_name; set installbootrecord yes; commit"
     ```

   - To set the `installbootrecord` setting for a service node category, enter:

     ```
     cims1 # cmsh -c "category use catagory_name; set installbootrecord yes; commit"
     ```

7. Unplug the cable for the `esmaint-net` network (the provisioning network or BOOTIF network) on the service node and reboot the node to determine if it can boot successfully without access to the CIMS node.

TAS Password Administration

There are several administrative passwords for a Cray TAS system. Each password is described below:

- CIMS password. The `root` password for the CIMS node.

- Software images. The `root` password for software images: This allows a `root` log in to a service node, and is stored in the software image.
- The node installer. The `root` password for the node-installer enables a `root` user to log in to the node when the node-installer minimal operating system is running. The node-installer stage prepares the node for the final operating system when the node boots.
- SQL™ password. The `root` password for MySQL enable the `root` user to log in to the MySQL server.
- The administrator certificate password: This password decrypts the `/root/admin.pfx` file on the CIMS node so that the administrator certificate can be submitted to CMDaemon for administrative tasks.
- The baseboard management controller (iDRAC) password. The iDRAC password for the CIMS node allows a root log in to the iDRAC to manage the system and start a remote console. The iDRAC password is changed using the `cmsh` shell, and not the `cm-change-passwd` script.
- Network switch administrative passwords should be managed by connecting a console to the switch and enter the switch configuration commands.

Change TAS System Passwords

There are several administrative passwords for a Cray TAS system. Each password is described below:

- CIMS node password. The `root` password for the CIMS node.
- Software images. The `root` password for software images: This allows a `root` user to log in to a service node, and is stored in the software image.
- The node-installer. The `root` password for the node-installer allows a `root` user log in to the node when the node-installer minimal operating system is running. The node-installer stage prepares the node for the final operating system when the node boots.
- MySQL® password. The `root` password for MySQL enables a `root` user to log in to the MySQL server.
- The administrator certificate password: This password decrypts the `/root/admin.pfx` file on the CIMS node so that the administrator certificate can be submitted to CMDaemon for administrative tasks.
- The baseboard management controller BMC (iDRAC) password. The iDRAC password for the CIMS node enables a root user to log in to the iDRAC to manage the system and start a remote console. The iDRAC password is changed using `cmsh`, and not the `cm-change-passwd` script.
- Network switch administrative passwords should be managed by connecting a console to the switch (usually through a web browser) and entering the switch configuration commands for the device.

1. Log in to the CIMS node as `root`.
2. Enter `cm-change-passwd` and follow the prompts to change each password on the system.

```
cims1# 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
* administrator certificate for use with cmgui (/root/admin.pfx)

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: newrootpassword
Retype new UNIX password: newrootpassword
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: newdefaultimagepassword
Retype new UNIX password: newdefaultimagepassword
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: newnode-installerpassword
Retype new UNIX password: newnode-installerpassword
passwd: all authentication tokens updated successfully.
Change password for MYSQL root user? [y/N]: y
Changing password for MYSQL root user.
Old password: oldMYSQLpassword
New password: newMYSQLpassword
Re-enter new password: newMYSQLpassword
Change password for admin certificate file? [y/N]: y
Enter old password: oldcertificatepassword
Enter new password: newcertificatepassword
Verify new password: newcertificatepassword
Password updated
```

3. Use `cmsh` to change the CIMS node BMC (iDRAC port) password.

```
cims1# cmsh
[cims1]% partition use base
[cims1->partition[base]]% show
Parameter                      Value
------------------------------ --------------------------------
Administrator e-mail
BMC Password                   *********
BMC User ID                    2
BMC User name                  root
Burn configs                   <2 in submode>
Cluster name                   Training
Default burn configuration
Default category               default
Default software image         default-image
External network               site-admin-net
Externally visible IP
Failover                       not defined
Management network             esmaint-net
Masternode                     cims1
Name                           base
Name servers                   aaa.bbb.ccc.ddd  aaa.bbb.ccc.ddd
Node basename                  node
Node digits                    3
Notes                          <0 bytes>
Revision
Search domains                 your.domain.com
Time servers                   timeserver1.com timerserver2.com
Time zone                      America/Chicago

[cims1->partition[base]]% set bmcpassword newbmcpassword
[cims1->partition*[base*]]% commit
```

Bright Administration Certificate

The `/root/admin.pfx` file on the CIMS node must be copied to a remote system in order to run the `cmgui`.

IMPORTANT:

The Bright Cluster Manager™ (Bright) infrastructure (CMDaemon or `cmd`) requires public key authentication using X.509v3. X.509 is an ITU-T standard for a public key infrastructure (PKI) for single sign-on (SSO) and Privilege Management Infrastructure (PMI). The X.509 standard specifies, amongst

other things, standard formats for public key certificates, certificate revocation lists, attribute certificates, and a certification path validation algorithm. This means in practice, a person authenticating to the cluster management infrastructure must present his/her certificate (i.e. the public key) and in addition must have access to the private key that corresponds to the certificate. A certificate includes a profile that determines which cluster management operations the holder of the certificate may perform.

The administrator password provided during Bright installation encrypts the `admin.pfx` file generated as part of the installation. The same password is also used as the initial `root` password for all nodes. the password protected `admin.pfx` file is generated with the `cmd -c` command.

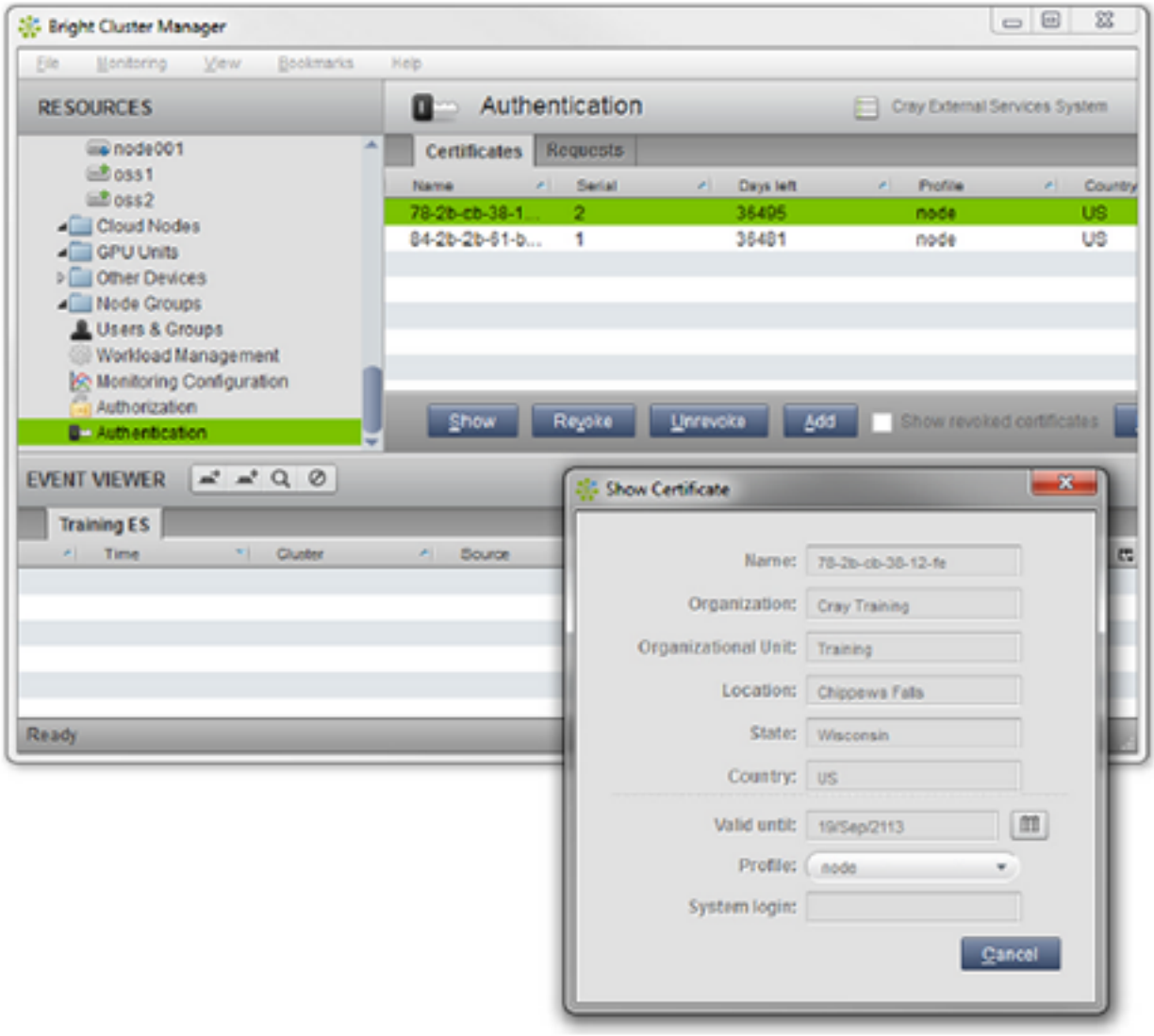
The administrator certificate is required to enable the CMDaemon, `cmgui`, and `cmsh` shell. Typically, administrators copy the `admin.pfx` file to their local laptop or workstation and use the Bright GUI (`cmgui`) to manage the system.

When the `/root/admin.pfx` file is updated with a new licence or password, the previous copy of the `admin.pfx` file continues to enable administrators to access the CMDaemon until the old version is revoked.

The password defined for the administrator certificate is used to decrypt the `admin.pfx` file, so that the administrator certificate can be presented to CMDaemon.

When the password for the `admin.pfx` file changes, the administrator must distribute the `admin.pfx` file and password to other administrators, and revoke older certificates to prevent administrators access with the old system certificate.

Figure 11. Bright Certificate Authentication Menu



Revoke Bright Administration Certificates

1. Log in to the CIMS as `root` and start `cmsh`.

```
cims1# cmsh
```

2. Switch to `cert` mode.

```
[cims1]% cert
[cims1->cert]% listcertificates
```

3. List the certificates. The Name column shows the MAC address of the node's `esmaint-net` network adapter.

```
[cims1->cert]% listcertificates
Serial num   Days left  Profile          Country          Name               Revoked
------------ ---------- ---------------- ---------------- ------------------ --------
1            36481      node             US               84-2b-2b-61-b0-04  No
2            36495      node             US               78-2b-cb-38-12-fe  No
```

4. To revoke a certificate, specify the serial number.

```
[cims1->cert]% revokecertificate 1
Certificate revoked.

[cims1->cert]% listcertificates
Serial num   Days left  Profile          Country          Name               Revoked
------------ ---------- ---------------- ---------------- ------------------ --------
1            36481      node             US               84-2b-2b-61-b0-04  Yes
2            36495      node             US               78-2b-cb-38-12-fe  No
```

Change the BMC Password

1. Log in to the CIMS node as `root` and start `cmsh`.

```
cims1# cmsh
```

2. Switch to `partition` mode. Use the `base` partition to change the BMC password.

```
[cims1]% partition use base
[cims1->partition[base]]%
```

3. Get the BMC user name.

```
[cims1->partition[base]]% get bmcusername
root
```

4. Get the BMC password (set during ESM software installation).

```
[cims1->partition[base]]% get bmcpassword
bmcpassword
```

5. Change and commit the BMC password.

```
[cims1->partition[base]]% set bmcpassword
enter new password: NewPassWord
retype new password: NewPassWord
```

6. Commit the BMC password change.

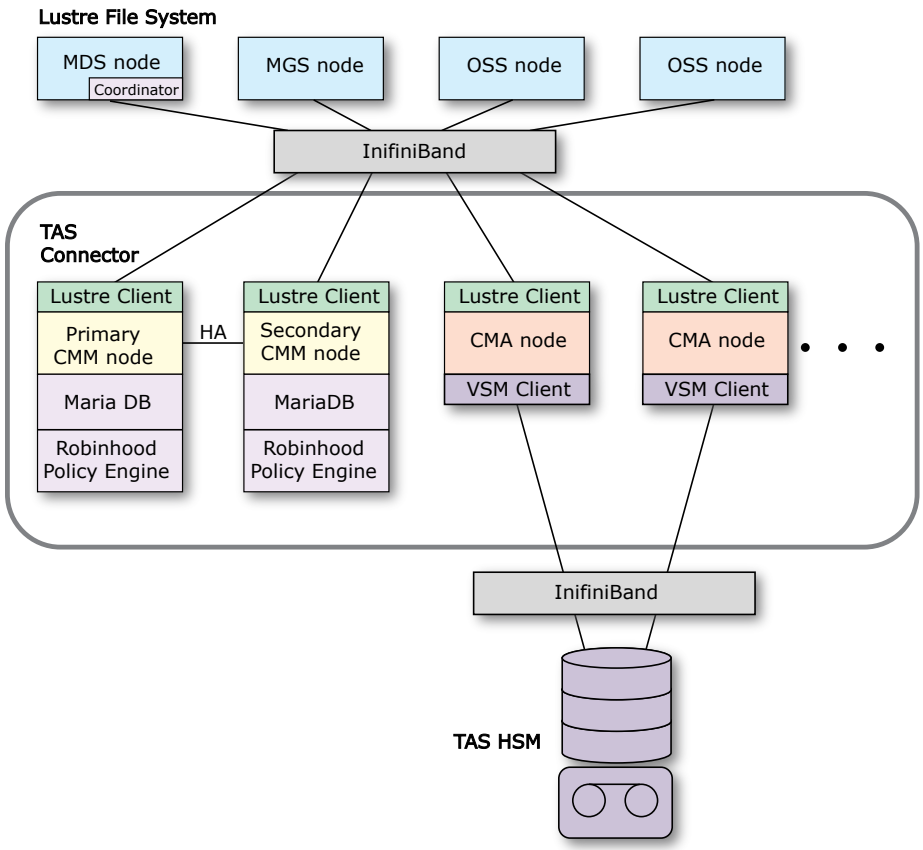
```
[cims1->partition[base*]]% commit
[cims1->partition[base]]%
```

TAS Connector Software Components

The TAS Connector uses HSM features in Lustre® 2.5 and a Policy Engine to coordinate the movement of files between a TAS HSM (*backend*) system and Lustre file system. TAS Connector software contains three primary software services:

- Robinhood policy engine and MariaDB database
- Migration manager
- Migration agent

Figure 12. TAS Connector Block Diagram



Lustre server 2.5 HSM features require a userspace application, or *Copytool*, to perform data movement between a Lustre file system and the TAS VSM file system. The Lustre HSM features are integrated directly within the Lustre file system MDS node, and the TAS Connector platform enables Cray's TAS HSM to act as another storage tier in the storage hierarchy by moving the data between the Lustre file system and the TAS VSM file system.

Cray's implementation of the Copytool can distribute the load across many physical nodes by running on a migration manager node (CMM) and several migration agent nodes (CMAs). If a migration agent fails, the migration manager restarts the work on another agent. Both the CMM and CMA nodes run Cray ESF-XX-2.2.0 software images, built from the CentOS 6.5 operating system customized specifically for TAS Connector.

Robinhood Policy Engine

The Robinhood policy engine is open-source software developed for monitoring and purging large temporary file systems. It is designed to perform all tasks in parallel, so it is particularly adapted for managing large file systems with millions of entries and exabytes of data.

Robinhood can also monitor Lustre usage per OST and also purge files per OST and/or OST pools. The Robinhood policy engine and migration manager software are co-located on migration manager (CMM) node(s). The CMM node connect to an external administrative block storage (ABS) device in the TAS Connector.

Migration Manager

The CMM nodes are the interface between Lustre HSM and the migration agent CMA nodes. High-availability (HA) failover architecture supports two CMM nodes. The TAS Connector HA failover architecture is initiated manually.

The CMM connects to the Lustre MDS coordinator and registers as a Copytool, which allows it to receive the Lustre HSM requests. The migration agent (CMA) is a daemon that performs the actual file copies between Lustre and the archives. CMA nodes have no specific knowledge of Lustre or the backend TAS HSM. CMA nodes execute requests sent by a CMM.

The active CMM can be shadowed by a passive CMM on a different host in a failover configuration for redundancy. A Robinhood policy engine maintains file system metadata in a MariaDB database and manages data migration and purge policies. The Robinhood policy engine does not communicate with the Copytool, although the Copytool executes orders coming from Robinhood policy engine through Lustre HSM.

Migration Agent

The migration agent nodes (CMA) move data between the Lustre file system and the TAS HSM over an InfiniBand or SAS connection (CMM and CMA nodes can only connect to the TAS HSM by InfiniBand or SAS).

The CMA nodes connect the TAS HSM Versity Storage Manager file system (VSM) to a Lustre client through an InfiniBand connection.

All services on Robinhood and the migration manager must be shut down before initiating a failover. If a system failure occurs, a secondary CMM node must be brought online so that the data stores can be checked.

CIMS Node Software Introduction

Cray Integrated Management Services (CIMS) software release (ESM) includes CIMS software, the SUSE Linux Enterprise Server (SLES™) Service Pack 3 (SP3) operating system, Bright Cluster Manager 6.1 software (Bright), Cray Lustre control and Cray tools and utilities. CIMS software is provided with the Cray ESM XX-3.1.0 software release.

Bright software provides:

- A management shell (`cmsh`)

- A graphical user interface (`cmgui`)
- A cluster management daemon (`cmd` command, or CMDaemon). CMDaemon runs on all nodes in the system. CMDaemon on a service node responds to requests from `cmsh` or `cmgui` on the CIMS node and communicates with the CMDaemon processes running on each service node. The CMDaemon processes on each service node communicate only with the CMDaemon processes running on the node.

Bright manages the hardware and software for all the devices and nodes in a system through the Bright CMDaemon process (`cmd`). Bright supports a GUI (`cmgui`) and command line shell (`cmsh`) interface. Either the `cmgui` or `cmsh` can be used to manage the system and there may be certain tasks are more easily visualized using `cmgui`, and other tasks are more efficient using the `cmsh`.

System administration may also be performed using the Bright GUI (`cmgui`). The `cmsh` command prompt displays an asterisk (*) when changes have not been committed. Configuration changes are queued until they are committed (saved). Be sure to commit changes using the commit command before exiting `cmsh`, or configuration changes are not saved to the Bright database.

Refer to the *Bright Cluster Manager 6.1 Administrator Manual* for detailed information about Bright software management. PDF files for the Bright manuals are stored on the CIMS node in the `/cm/shared/docs/cm` directory, and linked to from the `/root` directory.

TAS CMM Node Software Introduction

CMM nodes are installed with Cray ESF-XX-2.2.0 software which is built on CentOS 6.5 and includes Lustre® client software, InfiniBand software, `esfsprogs`, and other storage device tools and utilities. The TAS Connector `tas_cmm` command (and daemon) reads HSM commands from Lustre® and acts on them. When needed, it can offload some work to an Agent. It is usually run as a daemon, started by an `init` script.

The migration manager node (CMM) in a TAS Connector is the interface between the Lustre file system and the migration agent nodes (CMA). The CMM node receives requests from the Lustre HSM file system coordinator, then queues, sorts, prioritizes, and schedules the requests across one or more CMA nodes as required. Files are distributed across multiple CMA nodes based on a size threshold.

The CMA node sends updates to the Lustre file system during the copy operation and discards the request once a copy has completed. Two CMM nodes typically run in a high-availability configuration to provide fault tolerance. Both CMM nodes listen for events from the Lustre file system coordinator, but only the primary CMM node acts on the requests.

The CMM nodes run a Robinhood policy engine and MariaDB database software. Robinhood does not communicate with the Copytool, although the Copytool executes orders coming from Robinhood through the Lustre HSM coordinator.

CMM queues represent different priorities. If a higher priority request is sent to a CMA node, lower-priority requests are put on hold, and the higher-priority request is processed instead. If a file is very large, the CMM node can allocate more than one CMA node to initiate the transfer for either an archive or restore operation. This behaviour is controlled by the two configuration settings `min_filesize_striping` and `stripe_size`. When stripping is used the file is copied in parallel by 2 or more CMA nodes.

TAS CMA Node Software Introduction

CMA nodes are installed with Cray ESF-XX-2.2.0 software, which is built on CentOS 6.5 and includes Lustre® client software, InfiniBand software, `esfsprogs`, and other storage device tools and utilities. The `tas_cma` command executes file transfers sent by the migration manager (CMM) node. It is a daemon that runs as a system service.

The CMA node is responsible for performing the data movement from the Lustre file system to Cray TAS VSM file system; therefore, the CMA node runs a native client of both the Lustre file system and the VSM file system used as the cache for the archive. A Migration Agent (CMA) is a daemon that performs the actual file copies between Lustre and the archives.

The CMA node does not require any specific knowledge of the file system because it is a simple data transfer mechanism that responds to requests from the CMM node. Each CMA node is independent from any one CMM, and can accept requests from multiple CMM nodes that have registered with the CMA node. This provides the capability to manage multiple Lustre file systems from the same TAS Connector configuration.

Each CMA node can simultaneously copy multiple files on a single node. Copy requests from the CMM node are queued as they are received. When a copy slot is available, the oldest request is removed from the queue and the file is copied. Partial reports are sent to the CMM node regularly during the copy operation.

A userspace process copies data between the Lustre file system and the TAS HSM Versity Storage Manager file system (VSM) using a special FID directory `/`*`lustre_filesystem`*`/.lustre/fid/`*`XXXX`*.

TAS Connector Redundancy

The active CMM monitors the CMA nodes. If no traffic occurs on a CMA, then the active CMM sends a ping. If the active CMM does not get a reply, the CMA will be considered lost, and all the jobs submitted to that CMA will be reset and submitted to other functioning CMA nodes. The CMM will periodically attempt to reconnect to the lost CMA node.

Each CMA will also monitor the traffic from the CMM nodes. It is the responsibility of the CMM to monitor the CMAs, but if no traffic has been received from a CMM node, the node is considered lost, and all pending requests from this CMM are canceled and discarded, and the CMM automatically unregistered. If the CMM comes back up, it must be re-registed with the copytool.

TAS Connector Commands and Utilities

The following commands and utilities pages support the TAS Connector.

- `tascmctl` - TAS Connector for Lustre® control utility
- `tascm_import` - TAS Connector for Lustre import utility
- `tas_cmm` - TAS Connector Migration Manager
- `tas_cma` - TAS Connector Migration Agent
- `tas_connector.conf` - TAS Connector configuration file

TAS Connector `tascmctl` Command

Synopsis

```
tascmctl [options] [command]
```

Description

The `tascmctl` command sends a command to a TAS connector CMM node using the control socket and retrieves the status of the various connected processes. This utility can be used to add or remove a CMA node to a running CMM node.

The `tascmctl` command can also be used as a command line tool or as a shell:

This control utility can query the TAS Connector manager (CMM) and agent (CMA) nodes to retrieve their status and statistics.

The following commands are available:

- `ping` - ping the manager or an agent, given their name
- `stats` - retrieve the statistics from a manager or agent
- `status` - retrieve the status of a manager or agent

For `ping`, `stats` and `status`, if the name given is `cmm`, then the manager is pinged. If the name is `cma` then all the agents are pinged. If the name is `all` then the managers and all the agents are pinged. Each node has 5 seconds to reply.

In interactive mode, the commands `q` and `quit` are also supported, both used to quit the utility.

Options

`-c, --conf=FILE`

Location of the configuration file. Defaults to `/etc/tas_connector.conf`.

`-h, --help`

Display the help.

Files

`/etc/tas_connector.conf`

The TAS Connector configuration file for both the Manager and the Agent.

Ping the Manager

```
cmm#  tascmctl "ping cmm"
```

To display the status:

```
cmm#   tascmctl "status"
    Status for cmm1 (managers)
       TAS CONNECTOR MANAGER
       =====================
    | Name         |   Hostname  | Ports      |
    |--------------+-------------+------------|
    | tasclient01 | tasclient01 | 5957, 5958 |

       TAS CONNECTOR AGENTS
       ====================
    | Name    | State    | Hostname    |   Port |  Credits  | Last req. sent   | Last reply      |
    |---------+----------+-------------+--------+-----------+------------------+-----------------|
```

```
| dm1    | present | tasclient01 |   5658 |      1000 | Mar 09 14:43:44  | Mar 09 14:43:44 |
| dm3    | lost    | tasclient02 |   5558 |       999 | Mar 09 14:43:41  | never           |
| dm4    | lost    | tasclient03 |   5858 |       999 | Mar 09 14:43:41  | never           |

   LUSTRE FRONTENDS
   ================
|   max_hsm_req |   hsm_sem value | mountpoint   |   copytool registered |
|---------------+-----------------+--------------+-----------------------|
|           650 |             650 | /mnt/tas01   |                     1 |

   BACKENDS
   ========

| Name   | UID              | URI               |
|--------+------------------+-------------------|
| bk1    | 2e95f95455ae3bb7 | qfs:///vsm/tasfs1 |

   QUEUED WORK
   ===========

   UPKEEP WORK
   ===========

| Agent   | Operation   | Backend   | Request ID          | Length   | Filename   |
|---------+-------------+-----------+---------------------+----------+------------|
| dm3     | declare_cmm |     -     | 0x404a9c532e33b5a8 | -        | -          |
| dm4     | declare_cmm | -         | 0x404a9c532e33b5a9 | -        | -          |

Status for dm1 (agents)
   TAS CONNECTOR MANAGERS
  ======================
| Name        | URL                     | Registered storage   |
|-------------+-------------------------+----------------------|
| tasclient01 | tcp://tasclient01:5958 | fe1  /mnt/tas01      |
|             |                         | bk1  /vsm/tasfs1     |
```

Use tascmctl to Display Statistics

```
$ tascmctl "stats"
   Statistics for cmm1 (managers)
      Manager statistics:
     cmm1
     ========================
      Frontend: fe1
      -------------
|   Messages from HSM |   Generic failures |
|---------------------+--------------------|
|                     |
 1 |                   0 |
   | Operation   |   Requests |  Success  |   Failed |   Canceled |  Reset  |   Striped failure |
   |-------------+------------+-----------+----------+------------+---------+-------------------|
   | archive     |         34 |        34 |        0 |          0 |       0 |                 0 |
   | cancel      |          0 |         0 |        0 |          0 |       0 |                 0 |
   | other       |          0 |         0 |        0 |          0 |       0 |                 0 |
   | remove      |          0 |         0 |        0 |          0 |       0 |                 0 |
   | restore     |          0 |         0 |        0 |          0 |       0 |                 0 |
   | =Total=     |         34 |        34 |        0 |          0 |       0 |                 0 |
     Statistics for all frontends
     ----------------------------
   | Operation   |   Requests |   Success |   Failed |   Canceled |   Reset |   Striped failure |
   |-------------+------------+-----------+----------+------------+---------+-------------------|
   | archive     |         34 |        34 |        0 |          0 |       0 |                 0 |
   | cancel      |          0 |         0 |        0 |          0 |       0 |                 0 |
   | other       |          0 |         0 |        0 |          0 |       0 |                 0 |
   | remove      |          0 |         0 |        0 |          0 |       0 |                 0 |
   | restore     |          0 |         0 |        0 |          0 |       0 |                 0 |
   | =Total=     |         34 |        34 |        0 |          0 |       0 |                 0 |
     Agents statistics
     =================

   | Agent   | Hostname    |   #files |   #bytes |   #t-outs |   #fails |   #cancel |
   |---------+-------------+----------+----------+-----------+----------+-----------|
   | dm1     | tasclient01 |        0 |  4299944 |         0 |        0 |         0 |
   | dm3     | tasclient02 |        0 |        0 |         0 |        0 |         0 |
   | dm4     | tasclient03 |        0 |        0 |         0 |        0 |         0 |
   Statistics for dm1 (agents)

    Agent statistics: dm1
    =====================
   | Name of message   |   Count |
   |-------------------+---------|
   | invalid           |       0 |
   | req_cancel        |       0 |
   | req_done_copytool |       0 |
   | req_done_target   |       0 |
   | req_init_copytool |       2 |
   | req_init_target   |       4 |
   | req_ping_pong     |     201 |
   | req_remove        |       0 |
   | req_stats         |       1 |
   | req_status        |      19 |
   | req_transfer      |      34 |
```

Use Shell to Display Statistics

```
$ tascmctl
> stats
    Statistics for cmm1 (managers)
       Manager statistics:
      cmm1
. . .
```

Use Shell to Display Status

```
$ tascmctl
> status
    Status for cmm1 (managers)
       TAS CONNECTOR MANAGER
      ====================
    | Name         |   Hostname  | Ports      |
    |--------------+-------------+------------|
    | tasclient01 | tasclient01 | 5957, 5958 |
. . .
```

Files

/etc/tas_connector.conf

The TAS Connector configuration file for the Manager and Agent.

See Also

tas_cmm(1), tas_cma(1), tas_connector.conf(5)

TAS Connector `tascm_import` Utility

Synopsis

`tascm_import [options] --restore-path=`*`PATH`* `--backend=`*`PATH`* `<directory|file|FID> [...]`

Description

The `tascm_import` command imports a deleted file, or a set of deleted files, from the backend archive by recreating the entries in Lustre®, but does not restore the content. Content is restored by the CMM node when the file is accessed or when a restore operation is invoked.

The last parameters are a combination of:

- A FID to restore, whether surrounded by brackets or not, or
- The file in the archive to restore, or
- A directory containing many files to restore, such as a whole backend archive.

When a file is imported, its FID will change. The file on the archive will be moved and renamed accordingly.

Options

-r, --restore-path

Indicates the mountpoint of the Lustre® file system to restore the files on. This is not necessarily the file system from which the file originated.

-n, --backend=PATH

Points to the root path of the backend archive owning the files to be restored.

-l, --logging=TARGET

Set the logging target. The valid values are `stdout` for the standard output, `stderr` for the error output, `syslog` for the `syslog` facility or a file name. The default is `stderr`.

-L, --loglevel=LEVEL

Set the logging level. The valid values, in increasing verbosity, are: `ALERT`, `CRIT`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, `DEBUG` and `DEBUG2`. The default value is `ERROR`. The values can also be written in lower case.

-a, --archive-id=NUMBER

Archive id to use when importing. Must be 1 to 32. Defaults to 1 if not specified.

-h, --help

Display the help.

Import a Single File by FID

```
cmm_node# tascm_import --restore-path=/mnt/tas01 --backend=/vsm/tasfs1/
0x200001b70:0x118:0x0
```

Import a Single File by Archive Name

```
cmm_node# tascm_import --restore-path=/mnt/tas01 --backend=/vsm/tasfs1/ /vsm/
tasfs1/1/0/4/b/0/0x200001b70:0x118:0x0
```

Import a Whole Archive

```
cmm_node# tascm_import --restore-path=/mnt/tas01 --backend=/vsm/tasfs1/ /vsm/tasfs1/
```

See Also

tas_cmm(1), tas_cma(1), tas_connector.conf(5)

TAS Connector `tas_cmm` Command

Synopsis

```
tas_cmm [options]
```

Description

The TAS Connector `tas_cmm` command reads HSM commands from Lustre® and acts on them. When needed, it can offload some work to an Agent. It is usually run as a daemon, started by an `init` script.

Options

-c, --conf=FILE

Location of the configuration file. Defaults to `/etc/tas_connector.conf`.

-l, --logging=TARGET

Set the logging target. The valid values are `stdout` for the standard output, `stderr` for the error output, `syslog` for the `syslog` facility or a file name. The default is `stderr`.

-L, --loglevel=LEVEL

Set the logging level. The valid values, in increasing verbosity, are: `ALERT`, `CRIT`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, `DEBUG` and `DEBUG2`. The default value is `DEBUG`. The values can also be specified in lower case.

-d, --daemon

Run as a daemon. The default is to not run as a daemon.

-h, --help

Display the help.

Files

/etc/tas_connector.conf

The TAS Connector configuration file for the Manager and Agent.

See Also

tas_cma(1),tas_connector.conf(5)

TAS Connector `tas_cma` Command

Synopsis

```
tas_cma [options]
```

Description

The TAS Connector `tas_cma` command executes file transfers sent by the Manager. It is usually run as a daemon, started by an `init` script on a different host than the Manager.

Options

-c, --conf=FILE

Location of the configuration file. Defaults to `/etc/tas_connector.conf`.

-i, --id=ID

This Agent's name in the configuration file.

-l, --logging=TARGET

Set the logging target. The valid values are `stdout` for the standard output, `stderr` for the error output, `syslog` for the `syslog` facility or a file name. The default is `stderr`.

-L, --loglevel=LEVEL

Set the logging level. The valid values, in increasing verbosity, are: `ALERT`, `CRIT`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, `DEBUG` and `DEBUG2`. The default value is `DEBUG`. The values can also be specified in lower case.

-d, --daemon

Run as a daemon. The default is to not run as a daemon.

-h, --help

Display the help.

Files

/etc/tas_connector.conf

The TAS Connector configuration file for the Manager and Agent.

See Also

tas_cmm(1),tas_connector.conf(5)

TAS Connector Configuration File `tas_connector.conf`

Synopsis

`/etc/tas_connector.conf`

Description

The CMM and the CMA nodes take their configuration from a default file `/etc/tas_connector.conf`. They both share the same file, although some values are intended only for a CMM or CMA nodes. This file defines which CMAs and which backend archives are available to a CMM.

Each configuration file must define `lustre_fs`, which is the mount point of the Lustre® file system. The `archive_id` property is optional, and is either a number or an array of numbers ([1,2,3,4]). If there is an overlap in the range of archive IDs used, several CMM nodes will attempt to archive the file. This is a legitimate use as long as the backends are different systems.

At least one migration agent must be defined with a host and a port number. The agents may or may not be available during setup and are probed continuously until they appear.

There must be at least one backend defined. There can be a generic definition for the backend, such as `default`. In Robinhood, the destination backend can be specified in the `migration-hints` property as shown:

```
migration_hints="backend=tasfs2";
```

If the backend is not specified, then `default` is used if it has been set to something else.

Managers

At least one manager must be defined in the `managers:` section.

Frontend

Each configuration file must define `frontends:`, which currently can contain one or more frontends. The URI inside defines which file system to monitor and its mount point. Currently only the Lustre® file system is fully supported. The PosixFS file system can only copy files, but no metadata (ownership, right, extended attributes,

…) are conserved. The `archive_id` property is optional and is either a number or an array of numbers ([1,2,3,4]).

Backends

There must be at least one backend defined. Like with a frontend, it is also defined by a URI, and includes the file system type and a path. Only the QFS file system is fully supported. The path must be absolute (starting with a `/`). There can be a default backend named `default`. In Robinhood, the destination backend can be specified in the `migration-hints` property:

```
migration_hints="backend=tasfs2";
```

If the backend does not exist, an error displays. If the backend is not specified, then `default` is used. In that case, if `default` has not been set, and error displays. If the backend is defined but not available, the copy will fail. Files on a `qfs` will be saved in `tar` format, will all their metadata. Files on a PosixFS file system will only have their content saved.

Migration Agents

At least one migration agent must be defined with a host and port number. The agents may or may not be available during the start of the migration manager. It will be probed continuously until it appears.

File Format

The CMM and CMA can share the same configuration file or a copy of the same configuration file. Unknown parameters are ignored. All comments start with the `#` sign and are ignored. A comment can be inserted after a setting and empty lines are ignored.

The options are a property name followed by a string value (surrounded by double quotes), a number, a list (surrounded by braces) or an array of identical values (surrounded by brackets). Lists and arrays can include any type of properties. The first level of configuration apply to all processes. Some options can be overridden in a specific sub-section (such as cryptography or logging).

`logging:`

Set the logging destination. This could be the standard output (`stdout`), the standard error (`stderr`), the system log (`syslog`) which will usually go in `/var/log/messages`, or a file name.

`loglevel:`

Set the maximum logging level to display. In order of importance, the options are `alert`, `crit`, `error`, `warning`, `notice`, `info`, and `debug`. The default is `error`.

`progress_time, progress_size:`

CMM only. Progress_time is the minimum interval, in seconds, at which the CMA will send a progress report of a copy.`progress_size` is the minimum interval, in bytes, at which the CMA will send a progress report of a copy. Both options are inactive when set to 0 (default).

When used with Lustre®, this results can be obtained with:

`lfs hsm_action` *`filename`*

`crypto:`

Indicates which cryptography is desired. This is a list, and it can contain the following properties:

type:

Type of cryptography. It can be `none` or `c25519`. The default is none. none indicates no cryptography at all, and the other properties are meaningless. Curve25519 is a strong cryptography algorithm. It must be a public key and a private key, which can be generated by the command line tool `curve_keygen`. Since no authentication is currently implemented, both keys should be kept private (owned by `root` with no read access to anyone else).

secret_key:

(c25519) a string containing the secret key generated by `curve_keygen`.

public_key:

(c25519) a string containing the public key generated by `curve_keygen`.

managers:

This section describes the migration managers. Each property name sets the name of the manager. Each manager is defined by the following 2 properties:
host:

The hostname where the manager is running.

port:

Which port number to bind to.

The CMM will parse the whole list and must find exactly only one that matches the host. The `host` can be `localhost`, which will match the current system. This can be used to share the same configuration among different and independent systems.

frontends

This declares the source the file system and its HSM components. The variables to set depend on the file system itself. The following properties are supported:
uri

The name of the mount point. This is mandatory. This is the name of the file system (Lustre® or posixfs), followed by two forward slashes, followed by the mountpoint or main directory, and including the leading forward slash. Currently only the Lustre® file system is fully supported.

The posixfs file system can only copy files, but no metadata (ownership, right, extended attributes, ...) are conserved and should be used for debugging only.

Example:

```
uri="lustre:///mnt/tas01";
```

archive_id

Lustre®. Indicates which archive id to register with Lustre®. The valid numbers are 1 to 32. It can be a single number (e.g. 4), or an array of numbers separated by commas (e.g. [4,5,6,7]). The CMM will only receive HSM requests belonging to these archives. If not set, then the CMM will receive the requests for all archives. This is the default.

Example:

```
archive_id=1;
```

lustre_fifo_event

Lustre®. Full path of a fifo that can be read by an outside program to monitor the Connector operations. The fifo will be created if it does not exist. The output format is one JSON string per line. See the *Lustre Operations Manual* for more information.

Example:

```
lustre_fifo_event="/var/run/lustre_hsm_event";
```

cmd_fifo

PosixFS. Full pathname of the pipe (fifo) to which HSM commands are written for the PosixFS file system.

Example:

```
cmd_fifo="/var/run/tas_connector/cmd_fifo";
```

backends:

A section describing the backends. Each property name is the name of the backend. The name `default` indicates the catch-all backend; if the application archiving doesn't specify a backend name, then this backend will be used. The valid backend properties are:

uri

URI of the backend storage. It is required, and is the type of the file system, followed by a double forward slash, followed by the absolute path of the mountpoint. The possible values for a file system are QFS and PosixFS. Only the QFS file system is fully supported.

Files on a QFS will be saved in TAR format, will all their metadata. Files on a PosixFS file system will only have their content saved.

During operations of the TAS Connector, if a backend is defined but not available (for instance if the file system is not mounted), the copies will fail.

Example:

```
uri="qfs:///vsm/tasfs1";
```

format

Describes how the files are contained on the backend. It is either tar to write files in a tar file or file for a normal copy. When using QFS as a backend it is recommended to use tar.

Example:

```
format="tar";
```

When sending the request to the TAS Connector it is possible to select which backend to use. If Robinhood is used as a policy engine, the destination backend can be specified in the `migration-hints` property:

```
migration_hints="backend=tasfs2";
```

If `lfs` is used, its `--data` option will do the same thing:

```
lfs hsm_archive --data "backend=tasfs2" ...
```

agents:

A section describing the CMA nodes. Each property name is the name of the migration agent, and their values are a list of more properties containing:

host:

Which hostname the agent is running on.

port:

Which port number to bind to.

`workers:`

Number of worker threads to do the copies. Defaults to 1. For some relative safety, the accepted range is 1 to 1024.

`crypto:`

Same as the global crypto section, used to override it for this agent only.

TAS Connector Configuration

```
# TAS Connector configuration
# Global variables
logging="stderr";
logging_level="debug";

progress_size=10485760;
stripe_size=1048576;            # 1MiB stripping
min_filesize_striping=10485760;

# Optional crypto keys, shared by all
crypto: {
  type="c25519";
  public_key="Yne@$w-vo<fVvi]a<NY6T1ed:M$fCG*[IaLV{hID";
  secret_key="D:)Q[IlAW!ahhC2ac:9*A}h:p?([4%wOTJ%JR%cs";
};

# Managers
managers: {
  cmm1: {
      host="localhost";
      port=5957;
  };
};

# Frontends for the manager
frontends: {
    fe1: {
      uri="lustre:///mnt/tas01";
      archive_id=[1];
      lustre_fifo_event="/tmp/fooevent";
      max_hsm_req = 650;
   };

   fe2: {
       uri="posixfs:///mnt/tas01/fs_frontend";
       cmd_fifo="/tmp/cmd_fifo";
       default_backend="bk2";
   };
};

# Backends
backends: {
    bk1: { uri="qfs:///vsm/tasfs1"; format="tar"; };
    bk2: { uri="posixfs:///mnt/tas01/posixbackend"; format="file"; };
    default: { uri="qfs:///vsm/tasfs1"; format="tar"; }
};

# Agents
agents: {
    cma1: { host="localhost"; port=5657; workers=16; };
    cma2: { host="localhost"; port=5757; workers=16; };
    cma3: { host="localhost"; port=5557; workers=16; };
    cma4: { host="localhost"; port=5857; workers=16; };
};
```

See Also

`tas_cmm(1)`,`tas_cma(1)`

TAS Connector Archive Format

To keep the inode usage low, the files are archived in `tar` format to save the data and the extended attributes, including the stripes information, in a single file. The `tar` format is standard PaX, and can be extracted with the regular GNU Tar, BSD tar and star.

The `tar` file is stored in the backend archive, in a subdirectory, which path is made from a hash of its Lustre® FID.

For instance the file whose FID is `0x2000013b3:0x44:0x0` is stored on the backend in `/vsm/tasfs1/3/5/9/0/d/0x2000013b3:0x44:0x0`.

A CMA generated `tar` file will have the following characteristics:

- Only one data file per tar file. One file on Lustre® is stored in its own `tar` file.
- The `tar` file contains a single file, where the file name is the path name inside the Lustre® file system (the Lustre® mount point is not part of the file name).

Archive File

```
$ tar tvf /vsm/tasfs1/3/5/9/0/d/0x2000032e1:0x69:0x0

-rwxr-xr-x munge/apache 117024 2014-04-09 18:00 fztest/ls
```

Directory Name

To find the directory name, the FID is hashed with MD5, and five 4-bit words are extracted from the digest to make each components.

The total number of directory inodes is then:

$16+16^2+16^3+16^4+16^5 = 1{,}118{,}480$

The TAS Connector can reasonably store about 10000 files per directory. If the files are reasonably spread, with $16^5=1048576$ directories, the TAS Connector could store about 10 billion files per QFS. The current reasonable limit for QFS is about 200 million files per directory.

TAS Connector Data Communication

The CMM and the CMA nodes communicate through the network, using 0MQ sockets. (*http://zeromq.org/*). 0MQ provides different types of sockets, of which only PUSH/PULL (*n*-to-1) and REP/REQ (1-to-1) are used.

CMM

The CMM has:

- One control REP socket to receive queries from a utility.

- One PULL socket to receive data from CMAs. This is used for copy updates.
- One REQ socket per CMA to send its requests.

The port number indicated in the configuration file is the control socket listening port. The PULL socket is listening on that number plus one.

CMA

The CMA has:

- One control REP socket to receive queries from a remote CMM or utility.
- One PUSH socket per CMM to send feedback.

The port number indicated in the configuration file is the control socket listening port.

Request Types

A request from a CMM to a CMA goes through a REQ socket. The CMA will, almost immediately, acknowledge the request, and send a result. Depending on the nature of the request, it can be executed immediately (such as a CMM registration) and return a REP_COMPLETE code with a success or error code or, queued for processing (a data copy) and return a REP_QUEUED. When the execution is completed, a message will be sent from the CMA PUSH socket to the CMM PULL socket.

After the request is completed (successfully or not), it is disposed by the CMA, then by the CMM.

Authentication and Encryption

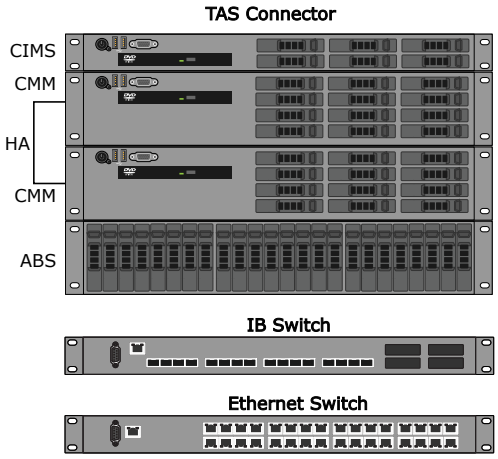
There are 2 levels of authentication and encryption that are supported on the network:

- No authentication and no encryption. For use on totally trusted networks.
- Strong encryption with elliptic curve algorithm, and authentication to join a trusted security domain. For use anywhere.

TAS Connector Hardware Components

A Cray TAS Connector system includes a Cray Integrated Management Services (CIMS) node, redundant migration manager nodes (CMM), migration agent nodes (CMA), and Ethernet, InfiniBand, or Fibre Channel (FC) switch infrastructure.

Figure 13. TAS Connector Gateway Hardware Components



- The Cray Integrated Management Services (CIMS) node is a 1U rack-mounted server that manages all the hardware and software components in the system and runs Bright Cluster Manager (Bright) software (CMDaemon or `cmd`). The CIMS node also stores, manages, and configures the migration manager node (CMM) and migration agent node (CMA) software images, and the software images for other service nodes in the system.
- The CMM and CMA nodes PXE boot from the CIMS node, therefore, software modifications for the CMM and CMA nodes must be made using the `chroot` shell in the software image on the CIMS node, then be pushed out to the service node(s) using Bright.
- The CMA nodes run a migration agent daemon that performs the file copies between Lustre® and the TAS HSM. The CMA node has no specific knowledge of the Lustre® file system or the TAS HSM. CMA node simply executes requests sent by the active CMM node. The CMA is an independent server providing a service to requesters, such as a CMM; it exists even if no CMMs are connected. A CMA can accept connections from many CMMs, which must register with it before submitting any work. A CMA is multi-threaded; it can perform multiple copies in parallel. As a consequence, there should only be a single CMA per host, although this is not a requirement.
- One or more Ethernet switches to support the maintenance network (`esmaint-net`, and `ipmi-net`), metadata network (`metadata-net`), customer administration (`site-admin-net`) and user networks.
- One or more InfiniBand (IB) switches to support high-speed data transfers from the CMM and CMA nodes to the NetApp storage devices or other high-performance storage systems.

- One or more Fibre Channel (FC) switches to support Fibre Channel connections from the CIMS node to CMM and CMA nodes.
- An external high-performance storage system supports the CMM nodes as administrative block storage (ABS).

CIMS Node Hardware Overview

The CIMS node is a 1U or 2U rack mounted server that runs the Bright software and provides a centralized platform to manage the system hardware and software.

The CIMS node is currently a Dell R630 1U server configured with the following features:

- Two 2.5GHz IVB 4-core Intel® Processors
- 64GB of memory
- 4 1TB disks

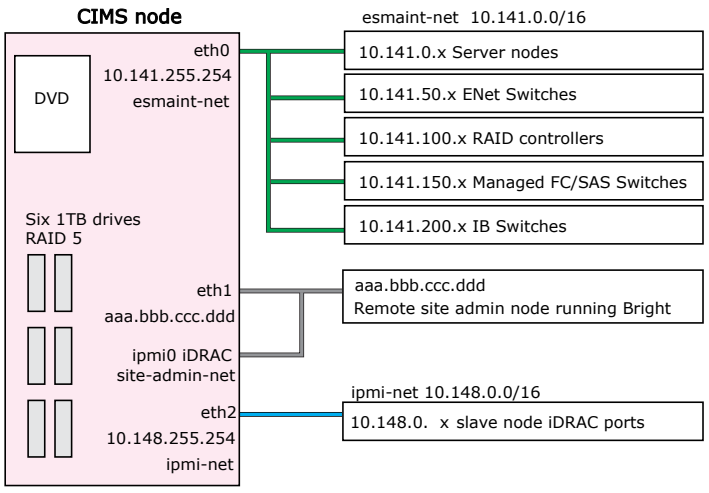
Other nodes managed by the CIMS node are called *service nodes*. A DVD drive provides a method for installing software from DVD release media. All system management software, service node software, and software updates and upgrades are installed from the CIMS node DVD drive.

A CIMS node includes six 1-TB disks, configured as two RAID-5 virtual disks (`/dev/sda` and `/dev/sdb`).

PCIe slots can be used to add IB, FC, SAS, or GigE connectivity to the CIMS. All service nodes in a DMP system are managed, monitored, and provisioned by the CIMS over the `esmaint-net` network. An Intelligent Platform Management Interface (IPMI) network (`ipmi-net`) is used to control power and monitor hardware using simple network monitoring protocol (SNMP).

The figure shows how the CIMS is connected to other nodes in the system either through a GigE switch divided into VLANs, or via separate Ethernet switches.

Figure 14. TAS CIMS Node Networks



The CIMS disk partitioning is configured differently for a stand-alone CIMS or a high-availability (HA) CIMS during the initial installation procedure of the Bright and Cray ESM software. The HA CIMS configurations make use of Distributed Replicated Block Device (DRBD) shared storage.

The CIMS node is supported in a high-availability (HA) configuration in some instances. TAS systems are typically not configured with two CIMS nodes.

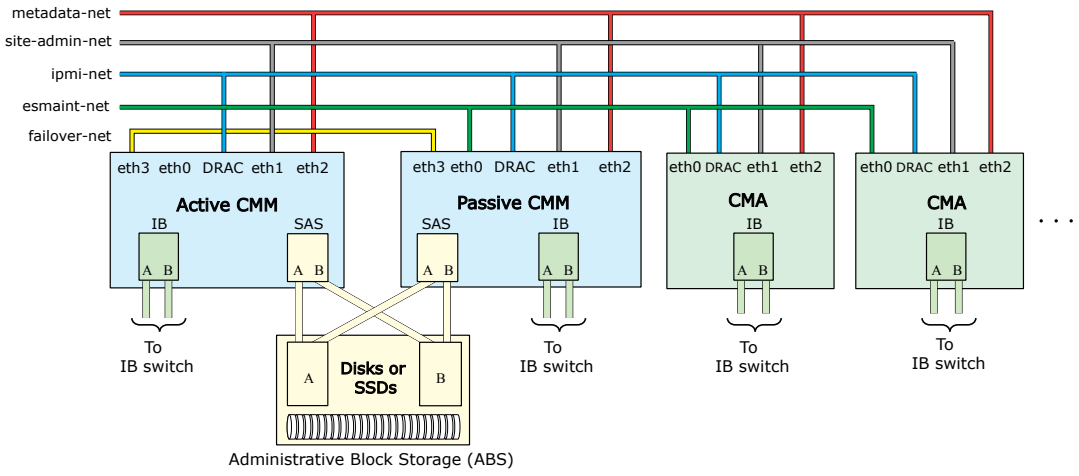
CMM and CMA Node Hardware

CMM and CMA Nodes

The TAS Connector CMM and CMA nodes are currently a Dell R730 2U servers configured with the following features:

- Two 2.6GHz IVB 8-core Intel® Processors
- 128GB of memory
- 2 600GB disks

The network connections for the CMM and CMA nodes are configured as shown.



TAS Connector Gateway Networks

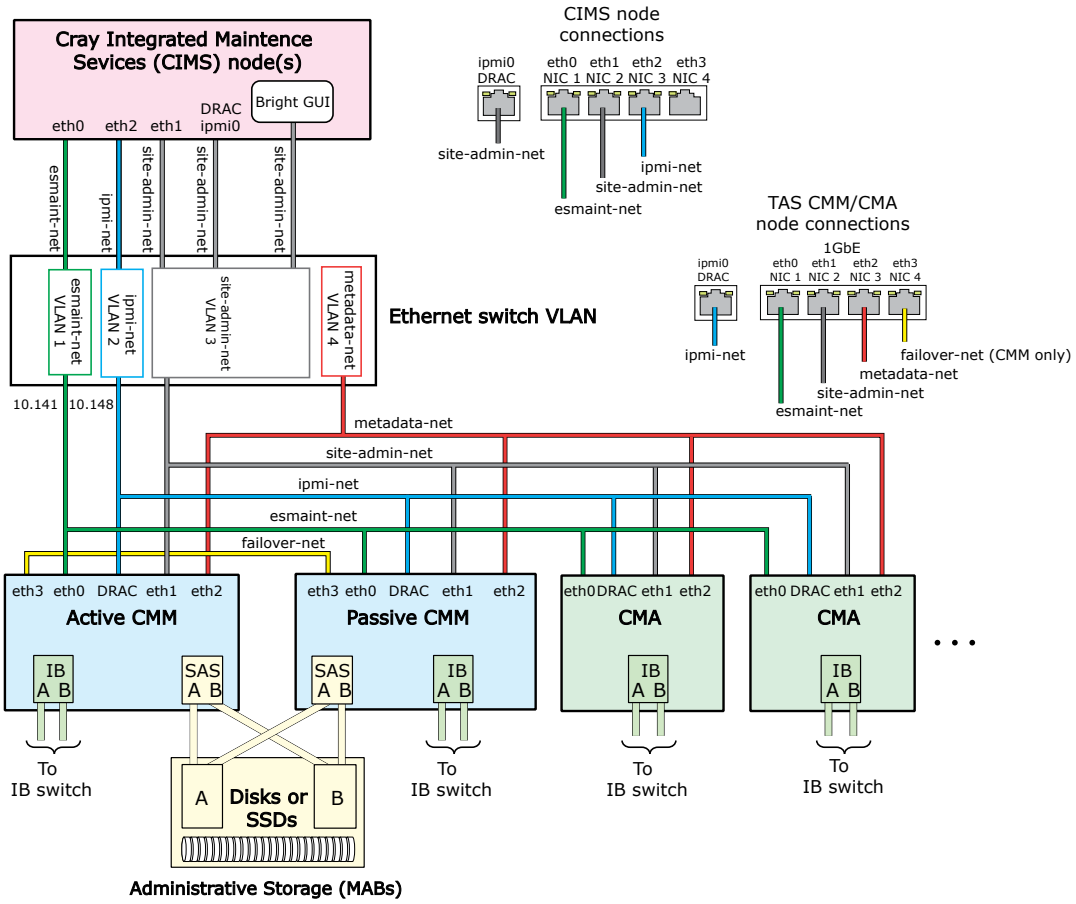
There are several required networks for a Cray TAS Connector system. TAS Connector networks are configured on the CIMS node during the software installation process.

Bright software uses *internal* and *external* designations to classify networks. The `esmain-net`, `ipmi-net`, for example are internal networks accessible only to the CIMS node. External networks in a TAS Connector system are the *`site-admin-net`*, which enable administrators to gain access. There are other network classifications within Bright, such as cloud and global, but these are not used. There may be additional networks defined, depending on the requirements of the system.

The primary networks used in a TAS Connector system are:

`esmaint-net`	An internal management network that connects the CIMS node(s) with the service nodes, switches, and RAID controllers. This network enables Bright to manage and provision the service nodes and other devices in the system. When using the Bright GUI (`cmgui`) or Cluster Management Shell (`cmsh`) this network is classified as the internal management network.
`ipmi-net`	Internal Intelligent Platform Management Interface (IPMI) connected to Dell™ Remote Access Controller (DRAC). Provides remote console and power management of the nodes.
`site-admin-net`	External administration network used by site administrators to log into the CIMS server. The name and IP address of this network are customized during installation. The CIMS IPMI interface (iDRAC) may also be on this network to provide remote console and power management of the system.
`metadata-net`	Metadata network for TAS Connector system.
`failover-net`	Internal failover network used between two servers in an HA configuration for heartbeats between the active/passive CMM nodes. This network does not connect to a managed switch.

Figure 15. TAS Connector Network Architecture



TAS Connector Robinhood Policy Engine

Introduction

The Robinhood Policy Engine is an Open-Source software developed at CEA/DAM for monitoring and purging large temporary file systems. It performs all tasks in parallel, so it is particularly adapted for managing and monitoring large Lustre® file system usage per OST. Robinhood can also purge files per OST and/or OST pools with millions of entries and petabytes of data.

Robinhood can synchronize a cluster file system with a Cray TAS HSM by applying administrator-defined migration and purge policies. In the daemon mode, Robinhood continuously scans a Lustre file system refreshes the list of file system entries and populating a MariaDB database with this information. Robinhood continuously reads MDT change logs to update its database in soft real-time. It regularly monitors file system and OST usage, and purge entries whenever needed.

Robinhood Service

This service starts one `robinhood` instance for each configuration file it finds in `/etc/robinhood.d/`*`database`* directory.

Internal Architecture

Robinhood uses a MariaDB relational structured query language (SQL) database engine for managing Lustre file system entries. MariaDB can manage large Lustre file systems because the size of the Lustre file system entries is not limited by the memory of the migration manager (CMM) node hardware. Scan and purge actions can be run on different nodes: no direct communication is needed between them; they only need to be database clients. Administrators can collect custom and complex statistics about file system from MariaDB content using SQL.

File system entry processing is highly parallelized: a pool of threads is dedicated to scanning the namespace (`readdir` operations). Those threads then push listed entries to a pipeline. Then, other operations are performed asynchronously by another pool of threads such as

- Get attributes
- Check if entry is up-to-date in database
- Get stripe info if it is not already known
- Check alert rules and send alerts
- Insert/update the entry in the database

Robinhood policies and alert rules are flexible and easy to configure using Boolean expressions. Robinhood can test a large range of attributes such as name, path, type, owner, group, size, access or modification time, extended attributes, depth in namespace tree, number of entries in directory, etc.

Robinhood supports various triggers for purge such as thresholds on OST usage or file system usage. Quota-like triggers can be configured based on volume of data or if the inode count of a file system has exceeded a threshold.

Interaction with the Policy Engine

Robinhood and TAS Connector do not communicate directly, however the first can send some parameters to the second through Lustre. Each TAS migration manager instance may register one or more `archive_id`'s. There are 32 `archive_id`'s supported, numbered from 1 to 32. `archive_id` 0 is a catch-all, and IDs 1 to 32 are unique.

When Robinhood issues an HSM command, it will be accompanied with the `archive_id` number, and Lustre distributes that command to all the migration manager (CMM) nodes that have registered with it, plus the manager that registered the default archive 0. Additionally, Robinhood supports the `migration-hints` property. It is a string containing a set of property/option couple that will be passed as-is to the registered manager. Currently only the `backend` property is supported.

Robinhood Configuration File

The Robinhood configuration file, `/etc/robinhood.d/`*`database`*`/`*`database`*`.conf` for example, consists of several blocks. Each block can contain key/value pairs (separated by semi-colons), sub-blocks, Boolean expressions or set definitions.

General Parameters

General parameters are set in a `General` configuration block. The `fs_path` definition is the path of the file system to be managed. This must be an absolute path. The `stay_in_fs` parameter, if TRUE, ensures that Robinhood entries are in the same device as `fs_path`, which prevents Robinhood from traversing mount points. The `lock_file` setting enables Robinhood to suspend its activity when this file exists. Refer to the Robinhood documentation for additional settings.

```
General
{
    # file system to be monitored
    fs_path = "/lustre" ;

    # file for suspending all actions
    lock_file = "/var/locks/robinhood.lock" ;

    # check that objects are in the same device as 'fs_path',
    # so it will not traverse mount points
    stay_in_fs = TRUE ;

    # check that the file system is mounted
    check_mounted = TRUE ;
}
```

Log Configuration

The parameters described in this section can be specified in the 'Log' block. They control logging, reporting and alerts.

```
# Log configuration
Log
{
    # Log verbosity level
    # Possible values are: CRIT, MAJOR, EVENT, VERB, DEBUG, FULL
    debug_level = EVENT ;

    # Log file
    log_file = "/var/log/robinhood.log" ;

    # File for reporting purge events
    report_file = "/var/log/robinhood_reports.log" ;
. . .
```

Purge Policies

In general, files are purged in the order of their last access time (LRU list). It is also possible to specify conditions to allow/avoid entries to be purged, depending on their file class, and file properties. Refer to *TAS Connector Purge Policies* on page 62 for more information.

File Classes

Define *file classes* in the Robinhood configuration file to apply different purge policies depending on file properties. A file class is defined by a `FileClass` block in `/etc/robinhood.d/`*`database`*`/robinhood.conf`. All file class definitions must be grouped in the `Filesets` block of the configuration file. Each file class has an identifier (that you can use for addressing it in policies) and a definition (a condition for entries to be in this file class). File classes can be defined as the union or the intersection of other file classes, using `inter` and `union` keywords in the file class definition.

A file class definition is structured as follows:

```
Filesets {
FileClass my_class_1 {
Definition {
tree == "/fs/dir_A"
and
owner == root
}
}
FileClass my_class_2 {
…
}
FileClass my_inter_class {
Definition { my_class_1 inter my_class_3 }
}
…
}
```

After modifying file class definitions or target file classes of policies, the file class information must be reset in Robinhood database. To do so, run the `rbh-config reset_classes` command.

Backend Property

This name will be matched against the backend names found in the configuration file. If no `backend` property is found, then the `default` backend is used. If the requested backend does not exists, the HSM command is deemed invalid and dropped.

Below is a partial Robinhood sample file followed with a matching partial migration manager configuration file:

```
# Robinhood
Filesets
{
        FileClass xattr_foo
        {
                Definition
                {
                        xattr.user.foo != ""
                }
                migration_hints="backend=\"mybackend1\"";
                archive_id = 7;
         }
}
```

```
# TAS CMM
archive_id = 7;
backends: (
    mybackend1: { path = "/vsm/tasfs1"; fstype="qfs"; }
    default: { path = "/vsm/tasfs1"; fstype="qfs"; }
);
```

Refer to the *Robinhood PolicyEngine Admin Guide*, available from *http://robinhood.sf.net/* for more information and support managing Robinhood.

TAS Connector MariaDB Database

MariaDB provides a simple `mysql` SQL shell with GNU readline capabilities and supports interactive and noninteractive use. When used interactively, query results are presented in an ASCII-table format. When used noninteractively (for example, as a filter), the result is presented in tab-separated format. The output format can be changed using command options.

TAS Connector CMM nodes are configured with a database (named `lhsm` for example). Use the Robinhood `rbh-config` command options to perform database maintenance and configuration.

Enter the following command to connect to the `lhsm` database from a CMM node.

```
[root@tas-cmm1 ~]# mysql --database lhsm
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8112
Server version: 10.0.16-MariaDB MariaDB Server

Copyright (c) 2000, 2014, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [lhsm]>
```

TAS Robinhood Database Configuration

The `rbh-config` script (installed in `/usr/sbin` on the CMM nodes) can configure and maintain the Robinhood database (MariaDB). It can be used as an interactive command, or in batch mode.

The database configuration is described in the `ListManager` section of the Robinhood configuration file `/etc/robinhood.d/lhsm/lhsm.conf` on the CMM node specifies global settings, log levels, list manager, and MySQL password, and various policy configuration settings.

Refer to the *Robinhood PolicyEngine Admin Guide*, available from *http://robinhood.sf.net/* for more information and support managing Robinhood.

In these examples the database is named `lhsm`.

Robinhood Configuration File Settings

/etc/robinhood.d/lhsm.conf General Section

The `General` section includes the `fs_path` setting which specifies which Lustre® file system to monitor.

```
cmm_node# cd /etc/robinhood.d/lhsm
cmm_node# vi lhsm.conf
General
{
    # filesystem to be monitored
```

```
    fs_path = "/lustre" ;

    # file for suspending all actions
    lock_file = "/var/locks/robinhood.lock" ;

    # check that objects are in the same device as 'fs_path',
    # so it will not traverse mount points
    stay_in_fs = TRUE ;

    # check that the file system is mounted
    check_mounted = TRUE ;
}
```

/etc/robinhood.d/lhsm.conf ListManager Section

The ListManager section controls how information is committed to the database. The commit_behavior is set to transaction to periodically commit Lustre file system changes. This section also includes the MariaDB password setting.

```
ListManager
{
    # Method for committing information to database.
    # Possible values are:
    # - "autocommit": weak transactions (more efficient, but database inconsistencies
may occur)
    # - "transaction": manage operations in transactions (best consistency, lower
performance)
    # - "periodic(<nb_transaction>)": periodically commit (every <n> transactions).
    commit_behavior = transaction ;

    # Minimum time (in seconds) to wait before trying to reestablish a lost connection.
    # Then this time is multiplied by 2 until reaching connect_retry_interval_max
    connect_retry_interval_min = 1 ;
    connect_retry_interval_max = 30 ;
    # disable the following options if you are not interested in
    # user or group stats (to speed up scan)
    user_acct  = enabled ;
    group_acct = enabled ;

    MySQL
    {
        server = "localhost" ;
        db     = "lhsm" ;
        user   = "robinhood" ;
        password_file = "/etc/robinhood.d/.dbpassword" ;
        # port   = 3306 ;
        # socket = "/tmp/mysql.sock" ;
        engine = InnoDB ;
    }
}
```

/etc/robinhood.d/lhsm.conf Policies FileClass Section

Under the Policies section Lustre file system and archive ID.

```
Filesets
{
    FileClass   all
    {
        definition
        {
            tree == "/lustre"
        }
        # target archive
        archive_id = 1 ;
    }
}
```

/etc/robinhood.d/lhsm.conf Migration Policies Section (Archiving)

Under the `migration_policies` the migration policy is configured using the following lines. A policy called `standard_copy` is configured to copy files from file system to the HSM if a newly created file that has not be archive is older than 1 minute, or has not been archived for longer than an hour, or if the file was modified longer than 30 seconds ago.

```
ignore
    {
        tree == /lustre/.lustre
    }
    policy      standard_copy
    {
        target_fileclass = all;

        condition
        {
            ((last_archive == 0 and creation > 1m) or last_archive > 1h) and last_mod >
30s
        }
        # target archive (/!\ policy archive_id overrides fileset archive_id)
        archive_id = 1;
    }
```

/etc/robinhood.d/lhsm.conf Remove Policies Section

The `hsm_remove_policy` section to delay object removal.

```
hsm_remove_policy
{
    # set this parameter to 'off' for disabling HSM object removal
    hsm_remove = enabled;
    # delay before impacting object removal in HSM
    deferred_remove_delay = 2h;
}
```

/etc/robinhood.d/lhsm.conf Purge Policies Section

The `purge_policies` section defines when to purge (release) files.

```
purge_policies
{
    # do not purge files owned by "foo" or "charlie"
    ignore
    {
        owner == "root"
    }
    # Default purge policy.
    # This applies to files that don't match previous fileclasses, i.e:
    #   - don't match the 'ignore' block
    #   - don't match a fileclass of 'ignore_fileclass' directives
    #   - don't match any 'target_fileclass' of purge policies above
    policy      default
    {
        condition
        {
            last_access > 1h
        }
    }
}
```

TAS Connector Robinhood `rbh-config` Script

The `rbh-config` script is installed in `/usr/sbin` and maintains the Robinhood database. It can be used as an interactive command, or in batch mode. To run `rbh-config` in interactive mode: just specify an action, it prompts for additional parameters.

Synopsys

`rbh-config` *`action`*

rbh-config

```
tas-cmm# rbh-config database
```

To run `rbh-config` in batch mode, specify all required parameters on command line.

```
tas-cmm# rbh-config database "robinhood_scratch" "passw0rd"
```

Available actions are:

Table 5. Robinhood rbh-config Command Actions

Action	Description
`precheck_db`	Check database packages and service.
`create_db`	Create Robinhood database.
`empty_db`	Clear Robinhood database content.
`test_db`	Checks if the database exists, and does a test login.
`repair_db`	Check and fix tables after a `mysql` server crash.
`reset_acct`	To rebuild accounting information if corrupted (stop/start robinhood daemon: see `rbh-config` inline help).
`reset_classes`	Reset file classes after a change in config file.
`enable_chglogs`	Enable ChangeLogs (must be executed on MDT).
`backup_db`	Backup robinhood database.
`optimize_db`	Defragments the database to improve performance and reduce disk usage.

TAS Connector Purge Policies

Files are purged (released) in the order of their last access time (LRU list) depending on their file class and file properties. To define purge policies, specify:

- Sets of entries that must never be purged (ignored)
- Purge policies to be applied to file classes
- A default purge policy for entries that don't match any file class

In the `/etc/robinhood.d/database/database.conf` configuration file.

```
purge_policies
{
    # do not purge files owned by "foo" or "charlie"
    ignore
    {
        owner == "root"
    }

    # Default purge policy.
    # This applies to files that don't match previous fileclasses, i.e:
    #   - don't match the 'ignore' block
    #   - don't match a fileclass of 'ignore_fileclass' directives
    #   - don't match any 'target_fileclass' of purge policies above
    policy      default
    {
        condition
        {
            last_access > 1h
        }
    }
}
```

A `purge_policies` block contains:

`ignore` sub-blocks

Boolean expressions to white-list file system entries depending on their properties. For example:

```
Ignore { size == 0 or type == "symlink" }
```

`policy` sub-blocks

Specify conditions for purging entries of file classes. A policy has a custom name, one or several target file classes, and a condition for purging files. For example:

```
Policy purge_classes_2and3
{
   target_fileclass = class_2;
   target_fileclass = class_3;
   condition
   {
      Last_access > 1h
   }
}
```

A default policy that applies to files that do not match any previous file class or `ignore` directive. It is a special `policy` block whose name is `default` and with no `target_fileclass`.

```
Policy default
{
   condition
   {
      last_access > 30min
   }
}
```

Target file classes are matched in the order they appear in the `purge_policies` block. Be sure to specify the more restrictive classes first. Thus, in the following example, the second policy can not be matched, because A already matches all entries in A_inter_B:

```
Filesets {
Fileclass A { … }
Fileclass B { … }
Fileclass A_inter_B { definition { A inter B } }
}
purge_policies
{
   policy purge_1
   {
```

```
      target_fileclass = A; # all entries of fileclass A
      …
   }
   policy purge_2
   {
      target_fileclass = A_inter_B; # never matched!!!
      …
   }
}
```

Purge Triggers

Triggers describe conditions for starting/stopping purges. They are defined by `purge_trigger` blocks in the `/etc/robinhood.d/`*`database`*`/`*`database`*`.conf` file. Each trigger consists of:

- The type of condition (on global filesystem usage, on OST usage, on volume used by a user or a group…)
- A purge start condition
- A purge target condition
- An interval for checking start condition.
- Notification options

Type of Condition

global_usage	Purge start/stop condition is based on the spaces used in the whole filesystem (based on `df` return). All entries in the file system are considered for such a purge.
OST_usage	Purge start/stop condition is based on the space used on each OST (based on `lfs df`). Only files stored in an OST are considered for such a purge.
user_usage(user1, user2…)	Purge start/stop condition is based on the space used by a user (type of quota). Only files that are owned by a user over the limit are considered for such a purge. If it is used with no arguments, all users will be affected by this policy. A list of users can also be specified for restricting the policy to a given set of users (comma-separated list of users between parenthesis).
group_usage(group1, group2…)	Purge start/stop condition is based on the space used by a group (type of quota). Only files that are owned by a group over the limit are considered for purge. If it is used with no arguments, all groups will be affected by this policy. A list of groups can also be specified for restricting the policy to a given set of groups (comma-separated list of groups between parenthesis).

Start Condition

This is mandatory for all types of conditions. A purge start condition can be specified by two ways: percentage or volume.

high_threshold_pct	Specifies a percentage of space used over which a purge is launched.
high_threshold_vol	Specifies a volume of space used over which a purge is launched. The value for this parameter can be suffixed by KB, MB, GB, TB…
high_threshold_cnt	Condition based on the number of inodes in the filesystem. It can be used for global_usage, user_usage and group_usage triggers. The value can be suffixed by K, M, …

Stop condition

This is mandatory for all types of conditions. A purge stop condition can also be specified by two ways: percentage or volume.

low_threshold_pct Specifies a percentage of space used under which a purge stops.

low_threshold_vol Specifies a volume of space used under which a purge stops. The value for this parameter can be suffixed by KB, MB, TB… (the value is interpreted as bytes if no suffix is specified).

low_threshold_cnt Condition based on the number (count) of inodes in the filesystem. It can be used for `global_usage`, `user_usage`, and `group_usage` triggers. The value can be suffixed by K, M, …

Runtime interval

The time interval for checking a condition is set by parameter `check_interval`. The value for this parameter can be suffixed by `sec`, `min`, `hour`, `day`, `week`, `year`… (the value is interpreted as seconds if no suffix is specified).

Check `df` every 5 minutes, purge if space used is > 85% of file system, and stop when space used is 84.5%

```
Purge_Trigger
{
   trigger_on = global_usage ;
   high_threshold_pct = 85% ;
   low_threshold_pct = 84.5% ;
   check_interval = 5min ;
}
```

Check OST usage every 5 minutes, start a purge of files on an OST if it space used is over 90% and stop purging when space used on the OST falls to 85%

```
Purge_Trigger
{
   trigger_on = OST_usage ;
   high_threshold_pct = 90% ;
   low_threshold_pct = 85% ;
   check_interval = 5min ;
}
```

Daily check the space used by each user of a given list. If one of them uses more than 1TB, its files are purged until it uses less than 800GB. Also send an alert in this case.

```
Purge_Trigger
{
   trigger_on = user_usage(foo, charlie, roger, project*) ;
   high_threshold_vol = 1TB ;
   low_threshold_vol = 800GB ;
   check_interval = 1day ;
   alert_high = TRUE ;
}
```

Check that user inode usage is less than 100k entries (and send a notification in this case):

```
Purge_Trigger
{
   trigger_on = user_usage ;
   high_threshold_cnt = 100k ;
   low_threshold_cnt = 100k ;
```

```
    check_interval = 1day ;
    alert_high = TRUE ;
}
```

Purge Parameters

Purge parameters are specified in a `purge_parameters` block.

nb_threads_purge	Integer for the number of purge operations that can be performed in parallel. `nb_threads_purge = 8 ;`
post_purge_df_latency	The duration, immediately after purging data, `df` and `ost df` may return a wrong value, especially if freeing disk space is asynchronous. So, it is necessary to wait for a while before issuing a new `df` or `ost df` command after a purge. This duration is set by this parameter. `post_purge_df_latency = 1min ;`
purge_queue_size	Advanced parameter (integer) for leveraging purge thread load.
db_result_size_max	An integer that impacts memory usage of MySQL server and Robinhood daemon. The higher it is, the more memory they will use, but less DB requests will be needed.
recheck_ignored_classes	(Boolean), this option results in checking previously ignored entries when applying purge policies (default behavior). Disable it to speed-up policy application.

Refer to the *Robinhood PolicyEngine Admin Guide*, available from *http://robinhood.sf.net/* for more information and support for managing Robinhood.