

---

# **xCAT3 Documentation**

***Release 2.17.0***

**IBM Corporation**

**Apr 23, 2024**



# CONTENTS

<b>1</b>	<b>Table of Contents</b>	<b>3</b>
1.1	Overview . . . . .	3
1.2	Install Guides . . . . .	16
1.3	Get Started . . . . .	34
1.4	Admin Guide . . . . .	39
1.5	Advanced Topics . . . . .	832
1.6	Questions & Answers . . . . .	1063
1.7	Reference Implementation . . . . .	1066
1.8	Troubleshooting . . . . .	1080
1.9	Developers . . . . .	1084
1.10	Need Help . . . . .	1103
1.11	Security Notices . . . . .	1103



xCAT stands for Extreme Cloud Administration Toolkit.

xCAT offers complete management of clouds, clusters, HPC, grids, datacenters, renderfarms, online gaming infrastructure, and whatever tomorrows next buzzword may be.

**xCAT enables the administrator to:**

1. Discover the hardware servers
2. Execute remote system management
3. Provision operating systems on physical or virtual machines
4. Provision machines in Diskful (stateful) and Diskless (stateless)
5. Install and configure user applications
6. Parallel system management
7. Integrate xCAT in Cloud

You've reached xCAT documentation site, The main page product page is <http://xcat.org>

**xCAT** is an open source project hosted on [GitHub](#). Go to GitHub to view the source, open issues, ask questions, and participate in the project.

Enjoy!



## TABLE OF CONTENTS

### 1.1 Overview

xCAT enables you to easily manage large number of servers for any type of technical computing workload. xCAT is known for exceptional scaling, wide variety of supported hardware and operating systems, virtualization platforms, and complete “day0” setup capabilities.

#### 1.1.1 Differentiators

- xCAT Scales  
Beyond all IT budgets, up to 100,000s of nodes with distributed architecture.
- Open Source  
Eclipse Public License. Support contracts are also available, contact IBM.
- Supports Multiple Operating Systems  
RHEL, SLES, Ubuntu, Debian, CentOS, Fedora, Scientific Linux, Oracle Linux, Windows, Esxi, RHEV, and more!
- Support Multiple Hardware  
IBM Power, IBM Power LE, x86\_64
- Support Multiple Virtualization Infrastructures  
IBM PowerKVM, KVM, IBM zVM, ESXI, XEN
- Support Multiple Installation Options  
Diskful (Install to Hard Disk), Diskless (Runs in memory), Cloning
- Built in Automatic discovery  
No need to power on one machine at a time for discovery. Nodes that fail can be replaced and back in action simply by powering the new one on.
- RestFUL API  
Provides a Rest API interface for the third-party software to integrate with

## **1.1.2 Features**

1. Discover the hardware servers
  - Manually define
  - MTMS-based discovery
  - Switch-based discovery
  - Sequential-based discovery
2. Execute remote system management against the discovered server
  - Remote power control
  - Remote console support
  - Remote inventory/vitals information query
  - Remote event log query
3. Provision Operating Systems on physical (Bare-metal) or virtual machines
  - RHEL
  - SLES
  - Ubuntu
  - Debian
  - Fedora
  - CentOS
  - Scientific Linux
  - Oracle Linux
  - PowerKVM
  - Esxi
  - RHEV
  - Windows
4. Provision machines in
  - Diskful (Scripted install, Clone)
  - Stateless
5. Install and configure user applications
  - During OS install
  - After the OS install
  - HPC products - GPFS, Parallel Environment, LSF, compilers ...
  - Big Data - Hadoop, Symphony
  - Cloud - Openstack, Chef
6. Parallel system management
  - Parallel shell (Run shell command against nodes in parallel)
  - Parallel copy



- Parallel ping

## 7. Integrate xCAT in Cloud

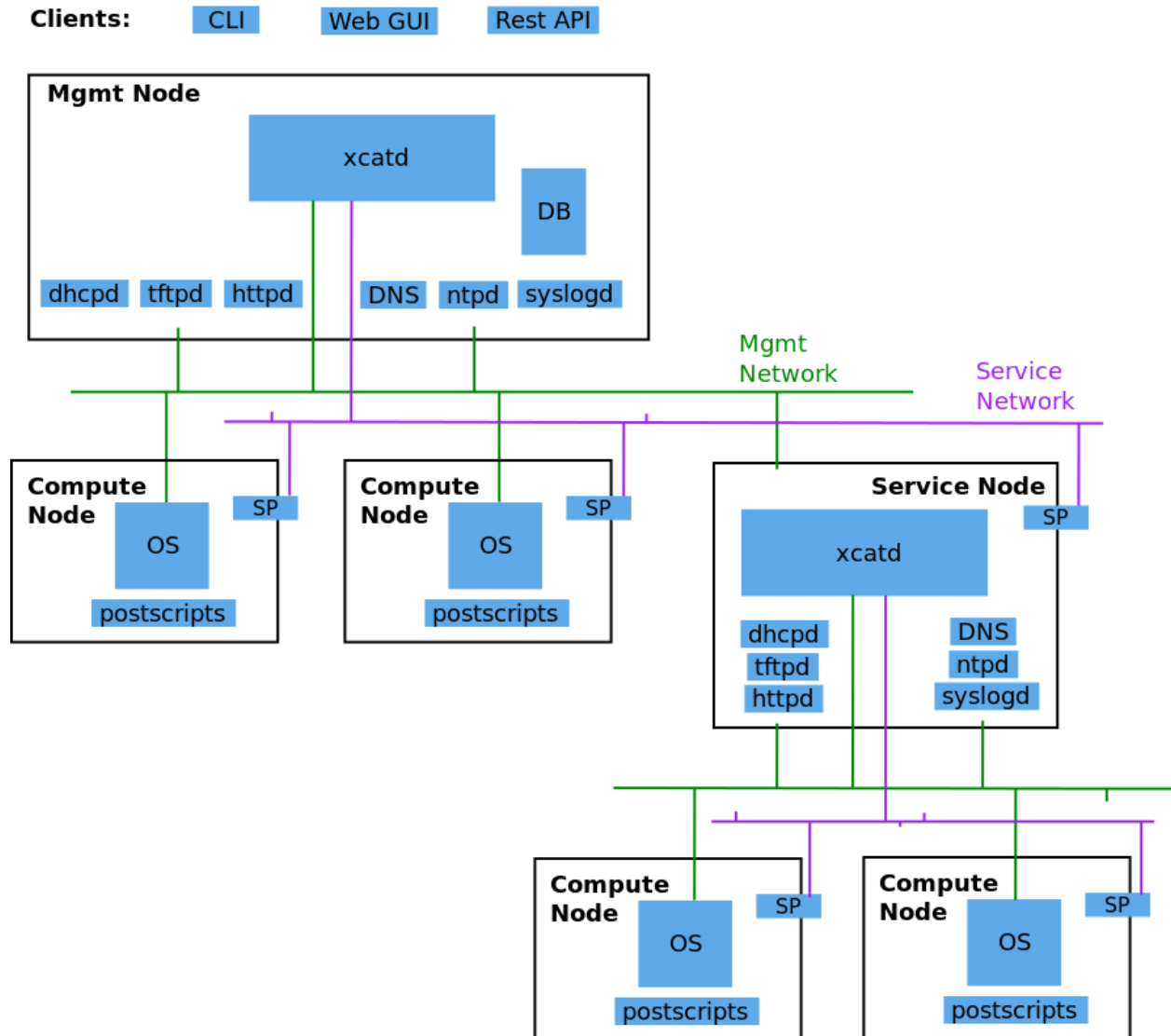
- Openstack
- SoftLayer

### 1.1.3 Operating System & Hardware Support Matrix

	Power	Power LE	zVM	Power KVM	x86_64	x86_64 KVM	x86_64 Esxi
RHEL	yes	yes	yes	yes	yes	yes	yes
SLES	yes	yes	yes	yes	yes	yes	yes
Ubuntu	no	yes	no	yes	yes	yes	yes
CentOS	no	no	no	no	yes	yes	yes
Windows	no	no	no	no	yes	yes	yes

### 1.1.4 Architecture

The following diagram shows the basic structure of xCAT:



#### xCAT Management Node (xCAT Mgmt Node):

The server where xCAT software is installed and used as the single point to perform system management over the entire cluster. On this node, a database is configured to store the xCAT node definitions. Network services (dhcp, tftp, http, etc) are enabled to respond in Operating system deployment.

#### Service Node:

One or more defined “slave” servers operating under the Management Node to assist in system management to reduce the load (cpu, network bandwidth) when using a single Management Node. This concept is necessary when managing very large clusters.

#### Compute Node:

The compute nodes are the target servers which xCAT is managing.

#### Network Services (dhcp, tftp, http,etc):

The various network services necessary to perform Operating System deployment over the network. xCAT will bring up and configure the network services automatically without any intervention from the System Administrator.

#### Service Processor (SP):

A module embedded in the hardware server used to perform the out-of-band hardware control. (e.g. Integrated

Management Module (IMM), Flexible Service Processor (FSP), Baseboard Management Controller (BMC), etc)

#### Management network:

The network used by the Management Node (or Service Node) to install operating systems and manage the nodes. The Management Node and in-band Network Interface Card (NIC) of the nodes are connected to this network. If you have a large cluster utilizing Service Nodes, sometimes this network is segregated into separate VLANs for each Service Node.

#### Service network:

The network used by the Management Node (or Service Node) to control the nodes using out-of-band management using the Service Processor. If the Service Processor is configured in shared mode (meaning the NIC of the Service process is used for the SP and the host), then this network can be combined with the management network.

#### Application network:

The network used by the applications on the Compute nodes to communicate among each other.

#### Site (Public) network:

The network used by users to access the Management Nodes or access the Compute Nodes directly.

#### RestAPIs:

The RestAPI interface can be used by the third-party application to integrate with xCAT.

## 1.1.5 xCAT2 Release Information

The following tables documents the xCAT release versions and release dates. For more detailed information regarding new functions, supported OSs, bug fixes, and download links, refer to the specific release notes.

### xCAT 2.16.x

Table 1: 2.16.x Release Information

Ver- sion	Release Date	New OS Supported	Release Notes
2.16.5	2023-03-08	RHEL 8.6	<a href="#">2.16.5 Release Notes</a>
2.16.4	2022-06-20	RHEL 8.5,SLES15.3	<a href="#">2.16.4 Release Notes</a>
2.16.3	2021-11-17	RHEL 8.4	<a href="#">2.16.3 Release Notes</a>
2.16.2	2021-05-25	RHEL 8.3	<a href="#">2.16.2 Release Notes</a>
2.16.1	2020-11-06	RHEL 8.2	<a href="#">2.16.1 Release Notes</a>
2.16.0	2020-06-17	RHEL 8.1,SLES 15	<a href="#">2.16.0 Release Notes</a>

### xCAT 2.15.x

Table 2: 2.15.x Release Information

Version	Release Date	New OS Supported	Release Notes
2.15.1	2020-03-06	RHEL 7.7	<a href="#">2.15.1 Release Notes</a>
2.15.0	2019-11-11	RHEL 8.0	<a href="#">2.15.0 Release Notes</a>

**xCAT 2.14.x**

Table 3: 2.14.x Release Information

Version	Release Date	New OS Supported	Release Notes
2.14.6	2019-03-29		<a href="#">2.14.6 Release Notes</a>
2.14.5	2018-12-07	RHEL 7.6	<a href="#">2.14.5 Release Notes</a>
2.14.4	2018-10-19	Ubuntu 18.04.1	<a href="#">2.14.4 Release Notes</a>
2.14.3	2018-08-24	SLES 12.3	<a href="#">2.14.3 Release Notes</a>
2.14.2	2018-07-13	RHEL 6.10, Ubuntu 18.04	<a href="#">2.14.2 Release Notes</a>
2.14.1	2018-06-01	RHV 4.2, RHEL 7.5 (Power8)	<a href="#">2.14.1 Release Notes</a>
2.14.0	2018-04-20	RHEL 7.5	<a href="#">2.14.0 Release Notes</a>

**xCAT 2.13.x**

Table 4: 2.13.x Release Information

Version	Release Date	New OS Supported	Release Notes
2.13.11	2018-03-09		<a href="#">2.13.11 Release Notes</a>
2.13.10	2018-01-26		<a href="#">2.13.10 Release Notes</a>
2.13.9	2017-12-18		<a href="#">2.13.9 Release Notes</a>
2.13.8	2017-11-03		<a href="#">2.13.8 Release Notes</a>
2.13.7	2017-09-22		<a href="#">2.13.7 Release Notes</a>
2.13.6	2017-08-10	RHEL 7.4	<a href="#">2.13.6 Release Notes</a>
2.13.5	2017-06-30		<a href="#">2.13.5 Release Notes</a>
2.13.4	2017-05-09	RHV 4.1	<a href="#">2.13.4 Release Notes</a>
2.13.3	2017-04-14	RHEL 6.9	<a href="#">2.13.3 Release Notes</a>
2.13.2	2017-02-24		<a href="#">2.13.2 Release Notes</a>
2.13.1	2017-01-13		<a href="#">2.13.1 Release Notes</a>
2.13.0	2016-12-09	SLES 12.2	<a href="#">2.13.0 Release Notes</a>

**xCAT 2.12.x**

Table 5: 2.12.x Release Information

Version	Release Date	New OS Supported	Release Notes
2.12.4	2016-11-11	RHEL 7.3 LE, RHEV 4.0	<a href="#">2.12.4 Release Notes</a>
2.12.3	2016-09-30		<a href="#">2.12.3 Release Notes</a>
2.12.2	2016-08-19	Ubuntu 16.04.1	<a href="#">2.12.2 Release Notes</a>
2.12.1	2016-07-08		<a href="#">2.12.1 Release Notes</a>
2.12.0	2016-05-20	RHEL 6.8, Ubuntu 14.4.4 LE, Ubuntu 16.04	<a href="#">2.12.0 Release Notes</a>

## xCAT 2.11.x

xCAT Version	New OS	New Hardware	New Feature
xCAT 2.11.1 2016/04/22 <a href="#">2.11.1 Release Notes</a>			<ul style="list-style-type: none"> <li>• Bug fix</li> </ul>
xCAT 2.11 2015/12/11 <a href="#">2.11 Release Notes</a>	<ul style="list-style-type: none"> <li>• RHEL 7.2 LE</li> <li>• UBT 14.4.3 LE</li> <li>• UBT 15.10 LE</li> <li>• PowerKVM 3.1</li> </ul>	<ul style="list-style-type: none"> <li>• S822LC(GCA)</li> <li>• S822LC(GTA)</li> <li>• S812LC</li> <li>• NeuCloud OP</li> <li>• ZoomNet RP</li> </ul>	<ul style="list-style-type: none"> <li>• NVIDIA GPU for OpenPOWER</li> <li>• Infiniband for OpenPOWER</li> <li>• SW KIT support for OpenPOWER</li> <li>• renergy command for OpenPOWER</li> <li>• rflash command for OpenPOWER</li> <li>• Add xCAT Troubleshooting Log</li> <li>• xCAT Log Classification</li> <li>• RAID Configuration</li> <li>• Accelerate genimage process</li> <li>• Add bmcdiscover Command</li> <li>• Enhance xcatdebug-mode</li> <li>• new xCAT doc in ReadTheDocs</li> </ul>

## xCAT 2.10.x

xCAT Version	New OS	New Hardware	New Feature
xCAT 2.10 2015/07/31 <a href="#">2.10 Release Notes</a>	<ul style="list-style-type: none"> <li>• RHEL 7.1 LE</li> <li>• UBT 15.4 LE</li> <li>• SLES 12 LE</li> <li>• RHEL 6.7</li> <li>• CentOS 7.1</li> <li>• SLES 11 SP4</li> </ul>	<ul style="list-style-type: none"> <li>• Power 8 LE</li> </ul>	<ul style="list-style-type: none"> <li>• Ubuntu LE -&gt; RH 7.1 Mix</li> <li>• Cuda install for Ubuntu 14.4.2</li> <li>• additional kernel parameters</li> <li>• customized disk part (Ubuntu)</li> <li>• RAID configure base iprconfig</li> <li>• New command: switchdiscover</li> <li>• New command: makentp</li> <li>• New command: bmcdiscovery</li> <li>• Support getmacs -noping</li> <li>• site.xcatdebugmode</li> <li>• validate netboot attribute</li> <li>• buildcore on local server</li> <li>• copycds generates fewer osimage</li> <li>• nodeset only accepts osimage=</li> </ul>

## xCAT 2.9.x

xCAT Version	New OS	New Hardware	New Feature
xCAT 2.9.3 for AIX 2016/03/11 <a href="#">2.9.3 Release Notes</a>	<ul style="list-style-type: none"> <li>• AIX 7.2.0</li> <li>• AIX 7.1.4.1</li> </ul>		<ul style="list-style-type: none"> <li>• new format in synclist (node)</li> </ul>
xCAT 2.9.2 for AIX 2015/11/11 <a href="#">2.9.2 Release Notes</a>	<ul style="list-style-type: none"> <li>• AIX 6.1.8.6</li> <li>• AIX 6.1.9.5</li> <li>• AIX 7.1.3.5</li> </ul>	<ul style="list-style-type: none"> <li>• Power 8 for AIX</li> </ul>	<ul style="list-style-type: none"> <li>• ssl version control in xcatd</li> </ul>
xCAT 2.9.1 <sup>1</sup> 2015/03/20 <a href="#">2.9.1 Release Notes</a>	<ul style="list-style-type: none"> <li>• RHEL 7.1</li> <li>• UBT 14.04.2</li> <li>• SLES 11 SP3 and later ONLY</li> </ul>		<ul style="list-style-type: none"> <li>• Nvidia GPU</li> <li>• Ubuntu Local Mirror</li> <li>• SLES12 diskless</li> <li>• Energy management for Power 8</li> <li>• RHEL 7.1 LE -&gt; BE mix cluster</li> <li>• nics.nicextraparams</li> <li>• xCAT in Docker Image</li> <li>• confluent replaces conserver</li> <li>• TLSv1 in xcatd</li> <li>• New GPG key for xCAT packages</li> <li>• fast restart xcatd (systemd)</li> <li>• netboot method: grub2-tftp</li> <li>• netboot method: grub2-http</li> </ul>
xCAT 2.9 2014/12/12 <a href="#">2.9 Release Notes</a>	<ul style="list-style-type: none"> <li>• UBT 14.4 LE</li> <li>• UBT 14.4.1 LE</li> <li>• UBT 14.10</li> <li>• SLES 12</li> <li>• RHEL 6.6</li> <li>• AIX 7.1.3.15</li> <li>• PowerKVM</li> </ul>	<ul style="list-style-type: none"> <li>• Power 8 LE</li> </ul>	<ul style="list-style-type: none"> <li>• sysclone enhancements</li> <li>• site.auditnosyslog</li> <li>• site.nmapoptions</li> <li>• customize postscripts</li> <li>• Power 8 LE hw discover</li> <li>• IB support for P8 LE</li> </ul>

<sup>1</sup> xCAT 2.9.1 onwards provides support for Kernel-based Virtual Machines (KVM) and requires an operating system that ships the perl-Sys-Virt package.





## xCAT 2.8.x

xCAT Version	New OS	New Hardware	New Feature
xCAT 2.8.4 2014/03/23 <a href="#">2.8.4 Release Notes</a>	<ul style="list-style-type: none"> <li>• RHEL 6.5</li> <li>• RHEL 5.10</li> </ul>		<ul style="list-style-type: none"> <li>• RHEL 7 experimental,</li> <li>• support xCAT clusterzones</li> <li>• commands enhancements</li> </ul>
xCAT 2.8.3 2013/11/15 <a href="#">2.8.3 Release Notes</a>	<ul style="list-style-type: none"> <li>• AIX 7.3.1.1</li> <li>• AIX 7.3.1.0</li> <li>• AIX 7.1.2</li> </ul>	<ul style="list-style-type: none"> <li>• Xeon Phi (P2)</li> <li>• NS nx360M4</li> </ul>	<ul style="list-style-type: none"> <li>• xcatd flow control</li> <li>• sysclone x86_64 image</li> <li>• enhance genitird and nodeset</li> <li>• enhance confignics, KIT</li> <li>• enhance sequential discovery</li> <li>• deploy OpenStack on Ubuntu</li> </ul>
xCAT 2.8.2 2013/06/26 <a href="#">2.8.2 Release Notes</a>	<ul style="list-style-type: none"> <li>• SLES 11 SP3</li> </ul>	<ul style="list-style-type: none"> <li>• Xeon Phi (P1)</li> </ul>	<ul style="list-style-type: none"> <li>• HPC KIT for ppc64</li> <li>• sysclone x86_64 image (P1)</li> <li>• enhance xdsh, updatenode</li> <li>• localdisk for diskless</li> <li>• enhance sequential discovery</li> <li>• deploy OpenStack on Ubuntu</li> </ul>
xCAT 2.8.1 2013/06/26 <a href="#">2.8.1 Release Notes</a>	<ul style="list-style-type: none"> <li>• RHEL 6.4</li> <li>• RHEL 5.9</li> </ul>		<ul style="list-style-type: none"> <li>• energy management for flex</li> <li>• sequential discovery</li> <li>• KIT enhancements</li> <li>• osimage enhancements</li> <li>• IPv6 enhancements</li> <li>• def/xdsh/xdcp enhancements</li> <li>• updatenode enhancements</li> </ul>
xCAT 2.8 2013/02/28 <a href="#">2.8 Release Notes</a>	<ul style="list-style-type: none"> <li>• UBT 12.04</li> <li>• WIN S 2012</li> <li>• WIN 8 Hv</li> </ul>		<ul style="list-style-type: none"> <li>• Flex IMM setup</li> <li>• Multiple Hostname</li> <li>• KIT support</li> <li>• KVM/zVM enhancements</li> </ul>
<b>1.1. Overview</b>			<ul style="list-style-type: none"> <li>• RHEV Support</li> <li>• Localdisk for statelite</li> <li>• Manage MN itslef</li> </ul>



## xCAT 2.7.x

xCAT Version	New OS	New Hardware	New Feature
xCAT 2.7.8 2014/01/24 <a href="#">2.7.8 Release Notes</a>	<ul style="list-style-type: none"> <li>AIX 7.1.3.1</li> <li>AIX 7.1.3.0</li> <li>AIX 6.1.9.1</li> </ul>		
xCAT 2.7.7 2013/03/17 <a href="#">2.7.7 Release Notes</a>	<ul style="list-style-type: none"> <li>RHEL 6.4</li> </ul>		<ul style="list-style-type: none"> <li>sinv for devices</li> <li>Flex energy mgt and rbeacon</li> </ul>
xCAT 2.7.6 2012/11/30 <a href="#">2.7.6 Release Notes</a>	<ul style="list-style-type: none"> <li>SLES 10 SP4</li> <li>AIX 6.1.8</li> <li>AIX 7.1.2</li> </ul>		<ul style="list-style-type: none"> <li>HPC Integration updates</li> </ul>
xCAT 2.7.5 2012/10/29 <a href="#">2.7.5 Release Notes</a>	<ul style="list-style-type: none"> <li>RHEL 6.3</li> </ul>		<ul style="list-style-type: none"> <li>virtualization with RHEV</li> <li>hardware discovery for x Flex</li> <li>enhanced AIX HASN</li> </ul>
xCAT 2.7.4 2012/08/27 <a href="#">2.7.4 Release Notes</a>	<ul style="list-style-type: none"> <li>SLES11 SP2</li> </ul>	<ul style="list-style-type: none"> <li>Flex</li> </ul>	<ul style="list-style-type: none"> <li>improved IPMI for large systems</li> </ul>
xCAT 2.7.3 2012/06/22 <a href="#">2.7.3 Release Notes</a>	<ul style="list-style-type: none"> <li>SLES11 SP2</li> <li>RHEL 6.2</li> </ul>	<ul style="list-style-type: none"> <li>Flex</li> </ul>	<ul style="list-style-type: none"> <li>HPC Integration updates</li> </ul>
xCAT 2.7.2 2012/05/25 <a href="#">2.7.2 Release Notes</a>	<ul style="list-style-type: none"> <li>AIX 7.1.1.3</li> </ul>	<ul style="list-style-type: none"> <li>Power 775</li> <li>Flex for P</li> </ul>	<ul style="list-style-type: none"> <li>SLES 11 kdump</li> <li>HPC Integration updates</li> </ul>
xCAT 2.7.1 2012/04/20 <a href="#">2.7.1 Release Notes</a>	<ul style="list-style-type: none"> <li>RHEL 6.3</li> </ul>		<ul style="list-style-type: none"> <li>minor enhancements</li> <li>bug fixes</li> </ul>
xCAT 2.7 2012/03/19 <a href="#">2.7 Release Notes</a>	<ul style="list-style-type: none"> <li>RHEL 6.2</li> </ul>		<ul style="list-style-type: none"> <li>xcata memory usage reduced</li> <li>xcatdebug for xcata and plugins</li> <li>lstree command</li> <li>x86_64 genesis boot image</li> </ul>

## 1.2 Install Guides

### 1.2.1 Installation Guide for Red Hat Enterprise Linux

For the current list of operating systems supported and verified by the development team for the different releases of xCAT, see the *xCAT2 Release Notes*.

**Disclaimer** These instructions are intended to only be guidelines and specific details may differ slightly based on the operating system version. Always refer to the operating system documentation for the latest recommended procedures.

#### Prepare the Management Node

These steps prepare the Management Node for xCAT Installation

#### Install an OS on the Management Node

Install one of the supported operating systems on your target management node.

The system requirements for your xCAT management node largely depend on the size of the cluster you plan to manage and the type of provisioning used (diskful, diskless, system clones, etc). The majority of system load comes during cluster provisioning time.

#### Memory Requirements:

Cluster Size	Memory (GB)
Small (< 16)	4-6
Medium	6-8
Large	> 16

#### Configure the Base OS Repository

xCAT uses the yum package manager on RHEL Linux distributions to install and resolve dependency packages provided by the base operating system. Follow this section to create the repository for the base operating system on the Management Node

1. Copy the DVD iso file to /tmp on the Management Node. This example will use file RHEL-LE-7.1-20150219.1-Server-ppc64le-dvd1.iso
2. Mount the iso to /mnt/iso/rhels7.1 on the Management Node.

```
mkdir -p /mnt/iso/rhels7.1
mount -o loop /tmp/RHEL-LE-7.1-20150219.1-Server-ppc64le-dvd1.iso /mnt/iso/rhels7.1
```

3. Create a yum repository file /etc/yum.repos.d/rhels71-dvd.repo that points to the locally mounted iso image from the above step. The file contents should appear as the following:

```
[rhel-7.1-dvd-server]
name=RHEL 7 SERVER packages
baseurl=file:///mnt/iso/rhels7.1/Server
enabled=1
gpgcheck=1
```

## Configure the Management Node

By setting properties on the Management Node before installing the xCAT software will allow xCAT to automatically configure key attributes in the xCAT site table during the install.

1. Ensure a hostname is configured on the management node by issuing the `hostname` command. *[It's recommended to use a fully qualified domain name (FQDN) when setting the hostname]*

1. To set the hostname of `xcatmn.cluster.com`:

```
hostname xcatmn.cluster.com
```

2. Add the hostname to the `/etc/sysconfig/network` in order to persist the hostname on reboot.
3. Reboot the server and verify the hostname by running the following commands:
  - `hostname`
  - `hostname -d` - should display the domain
2. Reduce the risk of the Management Node IP address being lost by setting the IP to **STATIC** in the `/etc/sysconfig/network-scripts/ifcfg-<dev>` configuration files.
3. Configure any domain search strings and nameservers to the `/etc/resolv.conf` file.

## Installing xCAT

The following sections describe the different methods for installing xCAT.

### Automatic Install Using go-xcat

`go-xcat` is a tool that can be used to fully install or update xCAT. `go-xcat` will automatically download the correct package manager repository file from `xcat.org` and use the public repository to install xCAT. If the xCAT management node does not have internet connectivity, use process described in the Manual Installation section of the guide.

1. Download the `go-xcat` tool using `wget`:

```
wget https://raw.githubusercontent.com/xcat2/xcat-core/master/xCAT-server/share/
↪xcat/tools/go-xcat -O - >/tmp/go-xcat
chmod +x /tmp/go-xcat
```

2. Run the `go-xcat` tool:

```
/tmp/go-xcat install          # installs the latest stable version of xCAT
/tmp/go-xcat -x devel install # installs the latest development version of xCAT
```

## Manual Install Using Software Repositories

xCAT consists of two software packages: `xcat-core` and `xcat-dep`

1. **xcat-core** xCAT's main software package and is provided in one of the following options:

- **Latest Release (Stable) Builds**

*This is the latest GA (Generally Availability) build that has been tested thoroughly*

- **Development Builds**

*This is the snapshot builds of the new version of xCAT in development. This version has not been released yet, use as your own risk*

2. **xcat-dep** xCAT's dependency package. This package is provided as a convenience for the user and contains dependency packages required by xCAT that are not provided by the operating system.

xCAT is installed by configuring software repositories for `xcat-core` and `xcat-dep` and using yum package manager. The repositories can be publicly hosted or locally hosted.

## Configure xCAT Software Repository

xCAT software and repo files can be obtained from: <http://xcat.org/download.html>

### Internet Repository

[**xcat-core**]

For the xCAT version you want to install, download the `xcat-core.repo` file and copy it to `/etc/yum.repos.d`

[**xcat-dep**]

From the [xCAT-dep Online Repository](#), navigate to the correct subdirectory for the target machine and download the `xcat-dep.repo` file and copy it to `/etc/yum.repos.d`.

Continue to the next section to install xCAT.

### Local Repository

[**xcat-core**]

1. Download `xcat-core`:

```
# downloading the latest stable build, xcat-core-<version>-linux.tar.bz2
mkdir -p ~/xcat
cd ~/xcat/
wget http://xcat.org/files/xcat/xcat-core/<version>.x_Linux/xcat-core/xcat-core-
-><version>-linux.tar.bz2
```

2. Extract `xcat-core`:

```
tar xcat-core-<version>-linux.tar.bz2
```

3. Configure the local repository for `xcat-core` by running `mklocalrepo.sh` script in the `xcat-core` directory:

```
cd ~/xcat/xcat-core
./mklocalrepo.sh
```

### [xcat-dep]

Unless you are downloading xcat-dep to match a specific version of xCAT, it's recommended to download the latest version of xcat-dep.

1. Download xcat-dep:

```
# downloading the latest stable version, xcat-dep-<version>-linux.tar.bz2
mkdir -p ~/xcat/
cd ~/xcat
wget http://xcat.org/files/xcat/xcat-dep/2.x_Linux/xcat-dep-<version>-linux.tar.bz2
```

2. Extract xcat-dep:

```
tar jxvf xcat-dep-<version>-linux.tar.bz2
```

3. Configure the local repository for xcat-dep by switching to the architecture and os subdirectory of the node you are installing on, then run the mklocalrepo.sh script:

```
cd ~/xcat/xcat-dep/
# On redhat 7.1 ppc64le: cd rh7/ppc64le
cd <os>/<arch>
./mklocalrepo.sh
```

## Install xCAT

Install xCAT with the following command:

```
yum clean all (optional)
yum install xCAT
```

**Note:** During the install, you must accept the *xCAT Security Key* to continue:

```
Retrieving key from file:///root/xcat/xcat-dep/rh6/ppc64/repodata/repomd.xml.key
Importing GPG key 0xC6565BC9:
  Userid: "xCAT Security Key <xcat@cn.ibm.com>"
  From : /root/xcat/xcat-dep/rh6/ppc64/repodata/repomd.xml.key
Is this ok [y/N]:
```

## Verify xCAT Installation

Quick verification of the xCAT Install can be done running the following steps:

1. Source the profile to add xCAT Commands to your path:

```
source /etc/profile.d/xcat.sh
```

2. Check the xCAT version:

```
lsxcatd -a
```

3. Check to verify that the xCAT database is initialized by dumping out the site table:

```
tabdump site
```

The output should be similar to the following:

```
#key,value,comments,disable
"blademaxp","64",,
"domain","pok.stglabs.ibm.com",,
"fsptimeout","0",,
"installdir","/install",,
"ipmimaxp","64",,
"ipmiretries","3",,
...
```

## Starting and Stopping

xCAT is started automatically after the installation, but the following commands can be used to start, stop, restart, and check xCAT status.

- start xCAT:

```
service xcatd start
[systemd] systemctl start xcatd.service
```

- stop xCAT:

```
service xcatd stop
[systemd] systemctl stop xcatd.service
```

- restart xCAT:

```
service xcatd restart
[systemd] systemctl restart xcatd.service
```

- check xCAT status:

```
service xcatd status
[systemd] systemctl status xcatd.service
```

## Updating xCAT

If at a later date you want to update xCAT on the Management Node, first, update the software repositories and then run:

```
yum clean metadata # or, yum clean all
yum update '*xCAT*'

# To check and update the packages provided by xcat-dep:
yum update '*xcat*'
```



If running in a hierarchical environment, Service Nodes must be the same xCAT version as the Management Node. To update Service Nodes: [Diskless](#) or [Diskful](#)

## 1.2.2 Installation Guide for SUSE Linux Enterprise Server

For the current list of operating systems supported and verified by the development team for the different releases of xCAT, see the [xCAT2 Release Notes](#).

**Disclaimer** These instructions are intended to only be guidelines and specific details may differ slightly based on the operating system version. Always refer to the operating system documentation for the latest recommended procedures.

### Prepare the Management Node

These steps prepare the Management Node for xCAT Installation

### Install an OS on the Management Node

Install one of the supported operating systems on your target management node.

The system requirements for your xCAT management node largely depend on the size of the cluster you plan to manage and the type of provisioning used (diskful, diskless, system clones, etc). The majority of system load comes during cluster provisioning time.

#### Memory Requirements:

Cluster Size	Memory (GB)
Small (< 16)	4-6
Medium	6-8
Large	> 16

### Configure the Base OS Repository

xCAT uses the zypper package manager on SLES Linux distributions to install and resolve dependency packages provided by the base operating system. Follow this section to create the repository for the base operating system on the Management Node

1. Copy the DVD iso file to /tmp on the Management Node:

```
# This example will use SLE-12-Server-DVD-ppc64le-GM-DVD1.iso
```

2. Mount the iso to /mnt/iso/sles12 on the Management Node.

```
mkdir -p /mnt/iso/sles12
mount -o loop /tmp/SLE-12-Server-DVD-ppc64le-GM-DVD1.iso /mnt/iso/sles12
```

3. Create a zypper repository file /etc/zypp/repos.d/sles12le-base.repo that points to the locally mounted iso image from the above step. The file contents should appear as the following:

```
[sles-12-le-server]
name=SLES 12 ppc64le Server Packages
baseurl=file:///mnt/iso/sles12/suse
enabled=1
pgpcheck=1
```

## Configure the Management Node

By setting properties on the Management Node before installing the xCAT software will allow xCAT to automatically configure key attributes in the xCAT site table during the install.

1. Ensure a hostname is configured on the management node by issuing the `hostname` command. *[It's recommended to use a fully qualified domain name (FQDN) when setting the hostname]*

1. To set the hostname of `xcatmn.cluster.com`:

```
hostname xcatmn.cluster.com
```

2. Add the hostname to the `/etc/hostname` in order to persist the hostname on reboot.
  3. Reboot the server and verify the hostname by running the following commands:
    - `hostname`
    - `hostname -d` - should display the domain
2. Reduce the risk of the Management Node IP address being lost by setting the IP to **STATIC** in the `/etc/sysconfig/network/ifcfg-<dev>` configuration files.
3. Configure any domain search strings and nameservers to the `/etc/resolv.conf` file.

## Installing xCAT

The following sections describe the different methods for installing xCAT.

### Automatic Install Using go-xcat

`go-xcat` is a tool that can be used to fully install or update xCAT. `go-xcat` will automatically download the correct package manager repository file from `xcat.org` and use the public repository to install xCAT. If the xCAT management node does not have internet connectivity, use process described in the Manual Installation section of the guide.

1. Download the `go-xcat` tool using `wget`:

```
wget https://raw.githubusercontent.com/xcat2/xcat-core/master/xCAT-server/share/
↪xcat/tools/go-xcat -O - >/tmp/go-xcat
chmod +x /tmp/go-xcat
```

2. Run the `go-xcat` tool:

```
/tmp/go-xcat install          # installs the latest stable version of xCAT
/tmp/go-xcat -x devel install  # installs the latest development version of xCAT
```

## Manual Install Using Software Repositories

xCAT consists of two software packages: `xcat-core` and `xcat-dep`

1. **xcat-core** xCAT's main software package and is provided in one of the following options:

- **Latest Release (Stable) Builds**

*This is the latest GA (Generally Availability) build that has been tested thoroughly*

- **Development Builds**

*This is the snapshot builds of the new version of xCAT in development. This version has not been released yet, use as your own risk*

2. **xcat-dep** xCAT's dependency package. This package is provided as a convenience for the user and contains dependency packages required by xCAT that are not provided by the operating system.

xCAT is installed by configuring software repositories for `xcat-core` and `xcat-dep` and using yum package manager. The repositories can be publicly hosted or locally hosted.

## Configure xCAT Software Repository

xCAT software and repo files can be obtained from: <http://xcat.org/download.html>

### Internet Repository

#### [xcat-core]

For the xCAT version you want to install, download the `xcat-core.repo` file and copy it to `/etc/zypp/repos.d`

#### [xcat-dep]

From the [xCAT-dep Online Repository](#), navigate to the correct subdirectory for the target machine and download the `xcat-dep.repo` file and copy it to `/etc/zypp/repos.d`.

Continue to the next section to install xCAT.

### Local Repository

#### [xcat-core]

1. Download `xcat-core`:

```
# downloading the latest stable build, xcat-core-<version>-linux.tar.bz2
mkdir -p ~/xcat
cd ~/xcat/
wget http://xcat.org/files/xcat/xcat-core/<version>.x_Linux/xcat-core/xcat-core-
-><version>-linux.tar.bz2
```

2. Extract `xcat-core`:

```
tar xcat-core-<version>-linux.tar.bz2
```

3. Configure the local repository for `xcat-core` by running `mklocalrepo.sh` script in the `xcat-core` directory:

```
cd ~/xcat/xcat-core
./mklocalrepo.sh
```

### [xcat-dep]

Unless you are downloading xcat-dep to match a specific version of xCAT, it's recommended to download the latest version of xcat-dep.

1. Download xcat-dep:

```
# downloading the latest stable version, xcat-dep-<version>-linux.tar.bz2
mkdir -p ~/xcat/
cd ~/xcat
wget http://xcat.org/files/xcat/xcat-dep/2.x_Linux/xcat-dep-<version>-linux.tar.bz2
```

2. Extract xcat-dep:

```
tar jxvf xcat-dep-<version>-linux.tar.bz2
```

3. Configure the local repository for xcat-dep by switching to the architecture and os subdirectory of the node you are installing on, then run the mklocalrepo.sh script:

```
cd ~/xcat/xcat-dep/
# On redhat 7.1 ppc64le: cd rh7/ppc64le
cd <os>/<arch>
./mklocalrepo.sh
```

## Install xCAT

Install xCAT with the following command:

```
zypper clean all (optional)
zypper install xCAT
```

**Note:** During the install, you must accept the *xCAT Security Key* to continue

## Verify xCAT Installation

Quick verification of the xCAT Install can be done running the following steps:

1. Source the profile to add xCAT Commands to your path:

```
source /etc/profile.d/xcat.sh
```

2. Check the xCAT version:

```
lsxcatd -a
```

3. Check to verify that the xCAT database is initialized by dumping out the site table:

```
tabdump site
```

The output should be similar to the following:

```
#key,value,comments,disable
"blademaxp","64",,
"domain","pok.stglabs.ibm.com",,
"fsptimeout","0",,
"installdir","/install",,
"ipmimaxp","64",,
"ipmiretries","3",,
...
```

## Starting and Stopping

xCAT is started automatically after the installation, but the following commands can be used to start, stop, restart, and check xCAT status.

- start xCAT:

```
service xcatd start
[systemd] systemctl start xcatd.service
```

- stop xCAT:

```
service xcatd stop
[systemd] systemctl stop xcatd.service
```

- restart xCAT:

```
service xcatd restart
[systemd] systemctl restart xcatd.service
```

- check xCAT status:

```
service xcatd status
[systemd] systemctl status xcatd.service
```

## Updating xCAT

If at a later date you want to update xCAT, first, update the software repositories and then run:

```
zypper refresh
zypper update "*xCAT*"

# To check and update the packages provided by xcat-dep:
zypper update "*xcat*"
```

### 1.2.3 Installation Guide for Ubuntu Server LTS

For the current list of operating systems supported and verified by the development team for the different releases of xCAT, see the *xCAT2 Release Notes*.

**Disclaimer** These instructions are intended to only be guidelines and specific details may differ slightly based on the operating system version. Always refer to the operating system documentation for the latest recommended procedures.

#### Prepare the Management Node

These steps prepare the Management Node or xCAT Installation

#### Install an OS on the Management Node

Install one of the supported operating systems on your target management node.

The system requirements for your xCAT management node largely depend on the size of the cluster you plan to manage and the type of provisioning used (diskful, diskless, system clones, etc). The majority of system load comes during cluster provisioning time.

#### Memory Requirements:

Cluster Size	Memory (GB)
Small (< 16)	4-6
Medium	6-8
Large	> 16

#### Configure the Base OS Repository

xCAT uses the apt package manager on Ubuntu Linux distributions to install and resolve dependency packages provided by the base operating system. Follow this section to create the repository for the base operating system on the Management Node

1. Copy the DVD iso file to /tmp on the Management Node:

```
# This example will use ubuntu-18.04-server-ppc64el.iso
cp /path/to/ubuntu-18.04-server-ppc64el.iso /tmp
```

2. Mount the iso to /mnt/iso/ubuntu on the Management Node.

```
mkdir -p /mnt/iso/ubuntu
mount -o loop /tmp/ubuntu-18.04-server-ppc64el.iso /mnt/iso/ubuntu
```

3. Create an apt repository file /etc/apt/repos.d/ubuntu18-dvd.repo that points to the locally mounted iso image from the above step. The file contents should appear as the following:

```
[ubuntu-dvd-server]
name=Ubuntu 18.04 Server packages
baseurl=file:///mnt/iso/ubuntu/Server
enabled=1
gpgcheck=1
```

## Configure the Management Node

By setting properties on the Management Node before installing the xCAT software will allow xCAT to automatically configure key attributes in the xCAT site table during the install.

1. Ensure a hostname is configured on the management node by issuing the `hostname` command. *[It's recommended to use a fully qualified domain name (FQDN) when setting the hostname]*
  1. To set the hostname of `xcatmn.cluster.com`:
 

```
hostname xcatmn.cluster.com
```
  2. Add the hostname to the `/etc/hostname` and `/etc/hosts` to persist the hostname on reboot.
  3. Reboot or run `service hostname restart` to allow the hostname to take effect and verify the hostname command returns correctly:
    - `hostname`
    - `hostname -d` - should display the domain
2. Reduce the risk of the Management Node IP address being lost by setting the interface IP to **STATIC** in the `/etc/network/interfaces` configuration file.
3. Configure any domain search strings and nameservers using the `resolvconf` command.

## Installing xCAT

The following sections describe the different methods for installing xCAT.

### Automatic Install Using go-xcat

`go-xcat` is a tool that can be used to fully install or update xCAT. `go-xcat` will automatically download the correct package manager repository file from `xcat.org` and use the public repository to install xCAT. If the xCAT management node does not have internet connectivity, use process described in the Manual Installation section of the guide.

1. Download the `go-xcat` tool using `wget`:

```
wget https://raw.githubusercontent.com/xcat2/xcat-core/master/xCAT-server/share/
↪xcat/tools/go-xcat -O - >/tmp/go-xcat
chmod +x /tmp/go-xcat
```

2. Run the `go-xcat` tool:

```
/tmp/go-xcat install          # installs the latest stable version of xCAT
/tmp/go-xcat -x devel install # installs the latest development version of xCAT
```

## Manual Install Using Software Repositories

xCAT consists of two software packages: `xcat-core` and `xcat-dep`

1. **xcat-core** xCAT's main software package and is provided in one of the following options:

- **Latest Release (Stable) Builds**

*This is the latest GA (Generally Availability) build that has been tested thoroughly*

- **Development Builds**

*This is the snapshot builds of the new version of xCAT in development. This version has not been released yet, use as your own risk*

2. **xcat-dep** xCAT's dependency package. This package is provided as a convenience for the user and contains dependency packages required by xCAT that are not provided by the operating system.

xCAT is installed by configuring software repositories for `xcat-core` and `xcat-dep` and using yum package manager. The repositories can be publicly hosted or locally hosted.

## Configure xCAT Software Repository

xCAT software and repo files can be obtained from: <http://xcat.org/download.html>

### Internet Repository

#### [xcat-core]

From the xCAT download page, find the build you want to install and add to `/etc/apt/sources.list`.

To configure the xCAT stable build, add the following line to `/etc/apt/sources.list`:

```
[For x86_64 servers]
deb [arch=amd64] http://xcat.org/files/xcat/repos/apt/latest/xcat-core bionic main
[For ppc64el servers]
deb [arch=ppc64el] http://xcat.org/files/xcat/repos/apt/latest/xcat-core bionic main
```

#### [xcat-dep]

To configure the xCAT deps online repository, add the following line to `/etc/apt/sources.list`:

```
[For x86_64 servers]
deb [arch=amd64] http://xcat.org/files/xcat/repos/apt/latest/xcat-dep bionic main
[For ppc64el servers]
deb [arch=ppc64el] http://xcat.org/files/xcat/repos/apt/latest/xcat-dep bionic main
```

Continue to the next section to install xCAT.



## Local Repository

### [xcat-core]

1. Download xcat-core:

```
# downloading the latest stable build, xcat-core-<version>-ubuntu.tar.bz2
mkdir -p ~/xcat
cd ~/xcat/
wget http://xcat.org/files/xcat/xcat-core/<version>.x_Ubuntu/xcat-core/xcat-core-
-><version>-ubuntu.tar.bz2
```

2. Extract xcat-core:

```
tar jxvf xcat-core-<version>-ubuntu.tar.bz2
```

3. Configure the local repository for xcat-core by running mklocalrepo.sh script in the xcat-core directory:

```
cd ~/xcat/xcat-core
./mklocalrepo.sh
```

### [xcat-dep]

Unless you are downloading xcat-dep to match a specific version of xCAT, it's recommended to download the latest version of xcat-dep.

1. Download xcat-dep:

```
# downloading the latest stable version, xcat-dep-<version>-ubuntu.tar.bz2
mkdir -p ~/xcat/
cd ~/xcat
wget http://xcat.org/files/xcat/xcat-dep/2.x_Ubuntu/xcat-dep-<version>-ubuntu.tar.
->bz2
```

2. Extract xcat-dep:

```
tar jxvf xcat-dep-<version>-ubuntu.tar.bz2
```

3. Configure the local repository for xcat-dep by running the mklocalrepo.sh script:

```
cd ~/xcat/xcat-dep/
./mklocalrepo.sh
```

## Install xCAT

The xCAT GPG Public Key must be added for apt to verify the xCAT packages

```
wget -O - - "http://xcat.org/files/xcat/repos/apt/apt.key" | apt-key add -
```

Add the necessary apt-repositories to the management node

```
# Install the add-apt-repository command
apt-get install software-properties-common
```

(continues on next page)

(continued from previous page)

```
# For x86_64:
add-apt-repository "deb http://archive.ubuntu.com/ubuntu $(lsb_release -sc) main"
add-apt-repository "deb http://archive.ubuntu.com/ubuntu $(lsb_release -sc)-updates main"
add-apt-repository "deb http://archive.ubuntu.com/ubuntu $(lsb_release -sc) universe"
add-apt-repository "deb http://archive.ubuntu.com/ubuntu $(lsb_release -sc)-updates_
↪universe"

# For ppc64el:
add-apt-repository "deb http://ports.ubuntu.com/ubuntu-ports $(lsb_release -sc) main"
add-apt-repository "deb http://ports.ubuntu.com/ubuntu-ports $(lsb_release -sc)-updates_
↪main"
add-apt-repository "deb http://ports.ubuntu.com/ubuntu-ports $(lsb_release -sc) universe"
add-apt-repository "deb http://ports.ubuntu.com/ubuntu-ports $(lsb_release -sc)-updates_
↪universe"
```

Install xCAT<sup>1</sup> with the following command:

```
apt-get clean all
apt-get update
apt-get install xcat
```

## Verify xCAT Installation

Quick verification of the xCAT Install can be done running the following steps:

1. Source the profile to add xCAT Commands to your path:

```
source /etc/profile.d/xcat.sh
```

2. Check the xCAT version:

```
lsxcatd -a
```

3. Check to verify that the xCAT database is initialized by dumping out the site table:

```
tabdump site
```

The output should be similar to the following:

```
#key,value,comments,disable
"blademaxp","64",,
"domain","pok.stglabs.ibm.com",,
"fsptimeout","0",,
"installdir","/install",,
"ipmimaxp","64",,
"ipmiretries","3",,
...
```

<sup>1</sup> Starting with Ubuntu 16.04, the package name 'xCAT' is required to be all lowercase

## Starting and Stopping

xCAT is started automatically after the installation, but the following commands can be used to start, stop, restart, and check xCAT status.

- start xCAT:

```
service xcatd start
[systemd] systemctl start xcatd.service
```

- stop xCAT:

```
service xcatd stop
[systemd] systemctl stop xcatd.service
```

- restart xCAT:

```
service xcatd restart
[systemd] systemctl restart xcatd.service
```

- check xCAT status:

```
service xcatd status
[systemd] systemctl status xcatd.service
```

## Updating xCAT

If at a later date you want to update xCAT, first, update the software repositories and then run:

```
apt-get update
apt-get -y --only-upgrade install . *xcat.*
```

## 1.2.4 Maintenance

### Backup and Restore xCAT

It's useful to backup xcat data sometime. For example, you need to upgrading to another version of xCAT, or you need to change management server and move xcat form one to another, or you need to make backups regularly and restore production environment for any accident. Below section will help you backup and restore xcat data.

#### Backup User Data

If need to backup xcat database, you can use *dumpxCATdb* command like below.

```
dumpxCATdb -p <path_to_save_the_database>
```

**[Note]** Maybe you need to dump some environment data for problem report when you hit defect, you can use *xcatsnap* command like below.

```
xcatsnap -B -d <path_to_save_the_data>
```

## Restore User Data

If need to restore xCAT environment, after *xCAT software installation*, you can restore xCAT DB using the *restorex-CATdb* command pointing to the data files dumped in the past.

```
restorexCATdb -p <path_to_save_the_database>
```

## Remove xCAT

**We're sorry to see you go!** Here are some steps for removing the xCAT product.

### Clean Up xCAT Related Configuration

1. To clean up the node information from dhcp

```
makedhcp -d -a
```

2. To clean up the node information in tftpboot

```
nodeset all offline
```

3. To clean up the node information from /etc/hosts (optional)

Keeping xCAT nodes information in /etc/hosts is harmless, but if you really want to remove them from /etc/hosts run:

```
makehosts -d all
```

4. To clean up the node information from DNS (optional)

After removing all the nodes from /etc/hosts, run below command to clean up the node information from DNS.

```
makedns -n
```

## Stop xCAT Service

1. Stop xCAT service

```
service xcatd stop
```

2. Stop xCAT related services (optional)

XCAT uses various network services on the management node and service nodes, the network services setup by xCAT may need to be cleaned up on the management node and service nodes before uninstalling xCAT.

- **NFS** : Stop nfs service, unexport all the file systems exported by xCAT, and remove the xCAT file systems from /etc/exports.
- **HTTP**: Stop http service, remove the xcat.conf in the http configuration directory.
- **TFTP**: Stop tftp service, remove the tftp files created by xCAT in tftp directory.
- **DHCP**: Stop dhcp service, remove the configuration made by xCAT in dhcp configuration files.

- **DNS** : Stop the named service, remove the named entries created by xCAT from the named database.

## Remove xCAT Files

### 1. Remove xCAT Packages

To automatically remove all xCAT packages, run the following command

```
/opt/xcat/share/xcat/tools/go-xcat uninstall
```

There is no easy way to identify all xCAT packages. For packages shipped by xCAT, you can manually remove them by using one of the commands below.

[RHEL]

```
yum remove conserver-xcat elilo-xcat goconserver grub2-xcat ipmitool-xcat perl-
↳xCAT syslinux-xcat xCAT xCAT-SoftLayer xCAT-buildkit xCAT-client xCAT-
↳confluent xCAT-csm xCAT-genesis-base-ppc64 xCAT-genesis-base-x86_64 xCAT-
↳genesis-scripts-ppc64 xCAT-genesis-scripts-x86_64 xCAT-openbmc-py xCAT-probe
↳xCAT-server xnba-undi yaboot-xcat
```

[SLES]

```
zypper remove conserver-xcat elilo-xcat goconserver grub2-xcat ipmitool-xcat
↳perl-xCAT syslinux-xcat xCAT xCAT-SoftLayer xCAT-buildkit xCAT-client xCAT-
↳confluent xCAT-csm xCAT-genesis-base-ppc64 xCAT-genesis-base-x86_64 xCAT-
↳genesis-scripts-ppc64 xCAT-genesis-scripts-x86_64 xCAT-openbmc-py xCAT-probe
↳xCAT-server xnba-undi yaboot-xcat
```

[Ubuntu]

```
apt-get remove conserver-xcat elilo-xcat goconserver grub2-xcat ipmitool-xcat
↳perl-xcat syslinux-xcat xcat xcat-buildkit xcat-client xcat-confluent xcat-
↳genesis-base-amd64 xcat-genesis-base-ppc64 xcat-genesis-scripts-amd64 xcat-
↳genesis-scripts-ppc64 xcat-probe xcat-server xcat-test xcat-vlan xcatsn xnba-
↳undi
```

To do an even more thorough cleanup, use links below to get a list of RPMs installed by xCAT. Some RPMs may not to be installed in a specific environment.

- XCAT Core Packages List (xcat-core)

[RHEL and SLES]

```
http://xcat.org/files/xcat/repos/yum/<version>/xcat-core/
```

[Ubuntu]

```
http://xcat.org/files/xcat/repos/apt/<version>/xcat-core/pool/main
```

- XCAT Dependency Packages (xcat-dep)

[RHEL and SLES]

```
http://xcat.org/files/xcat/repos/yum/xcat-dep/<os>/<arch>
```

[Ubuntu]

```
http://xcat.org/files/xcat/repos/apt/xcat-dep/pool/main
```

When `yum install xCAT` is used to install xCAT, dependency RPMs provided by the Operating System will be installed. Keeping those rpms installed on the system is harmless.

2. Remove xCAT certificate file

```
rm -rf /root/.xcat
```

3. Remove xCAT data files

By default, xCAT uses SQLite, remove SQLite data files under `/etc/xcat/`.

```
rm -rf /etc/xcat
```

4. Remove xCAT related files (optional)

XCAT might have also created additional files and directories below. Take caution when removing these files as they may be used for other purposes in your environment.

```
/install
/tftpboot
/etc/yum.repos.d/xCAT-*
/etc/sysconfig/xcat
/etc/apache2/conf.d/xCAT-*
/etc/logrotate.d/xCAT-*
/etc/rsyslogd.d/xCAT-*
/var/log/xcat
/opt/xcat/
/mnt/xcat
```

## Remove Databases

- *Removing xCAT DB from PostgreSQL.*
- *Removing xCAT DB from MySQL/MariaDB.*

## 1.3 Get Started

### 1.3.1 Quick Start Guide

xCAT can be a comprehensive system to manage infrastructure elements in Data Center, bare-metal servers, switches, PDUs, and Operation System distributions. This quick start guide will instruct you to set up a xCAT system and manage an IPMI managed bare metal server with Red Hat-based distribution in 15 minutes.

The steps below will be focused on RHEL7, however they should work for other distribution, such as CentOS, SLES, etc, details *Operating System & Hardware Support Matrix*

## Prerequisites

Assume there are two servers named `xcatmn.mydomain.com` and `cn1.mydomain.com`.

1. They are in the same subnet `192.168.0.0`.
2. `cn1.mydomain.com` has BMC which `xcatmn.mydomain.com` can access it.
3. `xcatmn.mydomain.com` has Red Hat OS installed, and uses IP `192.168.0.2`.
4. `xcatmn.mydomain.com` has access to internet.
5. `cn1.mydomain.com` BMC IP address is `10.4.40.254`.
6. Prepare a full DVD for OS provision, and not a Live CD ISO, for this example, will use `RHEL-7.6-20181010.0-Server-x86_64-dvd1.iso` ISO, you can download it from Red Hat website.

All the following steps should be executed in `xcatmn.mydomain.com`.

## Prepare the Management Node `xcatmn.mydomain.com`

1. Disable SELinux:

```
echo 0 > /selinux/enforce
sed -i 's/^SELINUX=.*$/SELINUX=disabled/' /etc/selinux/config
```

2. Set the hostname of `xcatmn.mydomain.com`:

```
hostname xcatmn.mydomain.com
```

3. Set the IP to STATIC in the `/etc/sysconfig/network-scripts/ifcfg-<proc_nic>` file
4. Update your `/etc/resolv.conf` with DNS settings and make sure that the node could visit github and xcat official website.
5. Configure any domain search strings and nameservers to the `/etc/resolv.conf` file
6. Add `xcatmn` into `/etc/hosts`:

```
192.168.0.2 xcatmn xcatmn.mydomain.com
```

7. Install xCAT:

```
wget https://raw.githubusercontent.com/xcat2/xcat-core/master/xCAT-server/share/
↪xcat/tools/go-xcat -O - >/tmp/go-xcat
chmod +x /tmp/go-xcat
/tmp/go-xcat --yes install
source /etc/profile.d/xcat.sh
```

8. Configure the system password for the ``root` user on the compute nodes:

```
chtab key=system passwd.username=root passwd.password=abc123
```

## Stage 1 Add your first node and control it with out-of-band BMC interface

1. Define compute node cn1:

```
mkdef -t node cn1 --template x86_64-template ip=192.168.0.3 mac=42:3d:0a:05:27:0c_
↪ bmc=10.4.40.254 bmcusername=USERID bmcpasswd=PASSWORD
```

2. Configure DNS:

```
makehosts cn1
makedns -n
```

3. Check cn1 Hardware Control:

cn1 power management:

```
rpwr cn1 on
rpwr cn1 state
cn1: on
```

cn1 firmware information:

```
rinv cn1 firm
cn1: UEFI Version: 1.31 (TDE134EUS 2013/08/27)
cn1: Backup UEFI Version: 1.00 (TDE112DUS )
cn1: Backup IMM Version: 1.25 (1A0026K 2012/02/23)
cn1: BMC Firmware: 3.10 (1A0048H 2013/08/22 18:49:44)
```

## Stage 2 Provision a node and manage it with parallel shell

1. In order to PXE boot, you need a DHCP server to hand out addresses and direct the booting system to the TFTP server where it can download the network boot files. Configure DHCP:

```
makedhcp -n
```

2. Copy all contents of Distribution ISO into /install directory, create OS repository and osimage for OS provision:

```
copycds RHEL-7.6-20181010.0-Server-x86_64-dvd1.iso
```

After copycds, the corresponding basic osimage will be generated automatically. Later, package list or postscripts for target compute nodes can be customised. List generated osimages:

```
lsdef -t osimage
```

3. Use xcatprobe to precheck xCAT management node ready for OS provision:

```
xcatprobe xcatmn
[mn]: Checking all xCAT daemons are running...
↪ [ OK ]
[mn]: Checking xcatd can receive command request...
↪ [ OK ]
[mn]: Checking 'site' table is configured...
↪ [ OK ]
```

(continues on next page)



(continued from previous page)

```
[mn]: Checking provision network is configured...
→ [ OK ]
[mn]: Checking 'passwd' table is configured...
→ [ OK ]
[mn]: Checking important directories(installdir,tftpd-dir) are configured...
→ [ OK ]
[mn]: Checking SELinux is disabled...
→ [ OK ]
[mn]: Checking HTTP service is configured...
→ [ OK ]
[mn]: Checking TFTP service is configured...
→ [ OK ]
[mn]: Checking DNS service is configured...
→ [ OK ]
[mn]: Checking DHCP service is configured...
→ [ OK ]
...
[mn]: Checking dhcpd.leases file is less than 100M...
→ [ OK ]
===== SUMMARY =====
[MN]: Checking on MN...
→ [ OK ]
```

#### 4. Start the Diskful OS Deployment:

```
rinstall cn1 osimage=rhels7.6-x86_64-install-compute
```

#### 5. Monitor Installation Process:

```
makegocons cn1
rcons cn1
```

**Note:** The keystroke `ctrl+e c .` will disconnect you from the console.

After 5-10 min verify provision status is booted:

```
lsdef cn1 -i status
Object name: cn1
status=booted
```

Use `xdsh` to check `cn1` OS version, OS provision is successful:

```
xdsh cn1 more /etc/*release
cn1: ::::::::::::::
cn1: /etc/os-release
cn1: ::::::::::::::
cn1: NAME="Red Hat Enterprise Linux Server"
cn1: VERSION="7.6 (Maipo)"
... ..
```

### 1.3.2 Workflow Guide

If xCAT looks suitable for your requirement, following steps are recommended to set up an xCAT cluster.

1. Find a server for xCAT management node

The server can be a bare-metal server or a virtual machine. The major factor for selecting a server is the number of machines in your cluster. The bigger the cluster is, the performance of server need to be better.

The architecture of xCAT management node is recommended to be same as the target compute node in the cluster.

2. Install xCAT on your selected server

The server where xCAT is installed will be the **xCAT Management Node**.

Refer to the doc: [xCAT Install Guide](#) to learn how to install xCAT on a server.

Refer to the doc: [xCAT Admin Guide](#) to learn how to manage xCAT Management server.

3. Discover target compute nodes in the cluster

Define the target nodes in the xCAT database before managing them.

For a small cluster (less than 5), you can collect the information of target nodes one by one and then define them manually through `mkdef` command.

For a bigger cluster, you can use the automatic method to discover the target nodes. The discovered nodes will be defined to xCAT database. You can use `lsdef` to display them.

Refer to the doc: [xCAT discovery Guide](#) to learn how to discover and define compute nodes.

4. Perform hardware control operations against the target compute nodes

Verify the hardware control for defined nodes. e.g. `rpower <node> stat`.

Refer to the doc: [Hardware Management](#) to learn how to perform the remote hardware control.

5. Deploy OS on the target nodes

- Prepare the OS images
- Customize the OS images (Optional)
- Perform the OS deployment

Refer to the doc: [Diskful Install](#), [Diskless Install](#) to learn how to deploy OS for a target node.

6. Update the OS after the deployment

You may require to update the OS of certain target nodes after the OS deployment, try the `updatenode` command. `updatenode` command can execute the following tasks for target nodes:

- Install additional software/application for the target nodes
- Sync some files to the target nodes
- Run some postscript for the target nodes

Refer to the doc: [Updatenode](#) to learn how to use `updatenode` command.

7. Run parallel commands

When managing a cluster with hundreds or thousands of nodes, operating on many nodes in parallel might be necessary. xCAT has some parallel commands for that.

- Parallel shell

- Parallel copy
- Parallel ping

Refer to the *Parallel Commands* to learn how to use parallel commands.

#### 8. Contribute to xCAT (Optional)

While using xCAT, if you find something (code, documentation, ...) that can be improved and you want to contribute that to xCAT, do that for your and other xCAT users benefit. And welcome to xCAT community!

Refer to the *Developers* to learn how to contribute to xCAT community.

## 1.4 Admin Guide

The admin guide is intended to help with learning how to manage a cluster using xCAT with the following major sections:

- **Basic Concepts** Introduces some of the basic concepts in xCAT.
- **Manage Cluster** Describes managing clusters under xCAT. The management procedures are organized based on the hardware type since management may vary depending on the hardware architecture.
- **Reference** xCAT reference sections.

### 1.4.1 Basic Concepts

xCAT is not hard to use but you still need to learn some basic concepts of xCAT before starting to manage a real cluster.

- **xCAT Objects**

The unit which can be managed in the xCAT is defined as an object. xCAT abstracts several types of objects from the cluster information to represent the physical or logical entities in the cluster. Each xCAT object has a set of attributes, each attribute is mapped from a specified field of a xCAT database table. The xCAT users can get cluster information and perform cluster management work through operations against the objects.

- **xCAT Database**

All the data for the xCAT Objects (node, group, network, osimage, policy ... and global configuration) are stored in xCAT Database. Tens of tables are created as the back-end of xCAT Objects. Generally the data in the database is used by user through **xCAT Objects**. But xCAT also offers a bunch of commands to handle the database directly.

- **Global Configuration**

xCAT has a bunch of **Global Configuration** for xCAT user to control the behaviors of xCAT. Some of the configuration items are mandatory for an xCAT cluster that you must set them correctly before starting to use xCAT.

- **xCAT Network**

xCAT's goal is to manage and configure a significant number of servers remotely and automatically through a central management server. All the hardware discovery/management, OS deployment/configuration and application install/configuration are performed through network. You need to have a deep understand of how xCAT will use network before setting up a cluster.

**Get Into the Detail of the Concepts:**

## xCAT Objects

Basically, xCAT has 20 types of objects. They are:

auditlog	boottarget	eventlog	firmware	group
kit	kitcomponent	kitrepo	monitoring	network
node	notification	osdistro	osdistroudate	osimage
policy	rack	route	site	zone

This section will introduce you to several important types of objects and give you an overview of how to view and manipulate them.

You can get the detail description of each object by `man <object type>` e.g. `man node`.

- **node Object**

The **node** is the most important object in xCAT. Any physical server, virtual machine or SP (Service Processor for Hardware Control) can be defined as a node object.

For example, I have a physical server which has the following attributes:

```
groups: all,x86_64
    The groups that this node belongs to.
arch: x86_64
    The architecture of the server is x86_64.
bmc: 10.4.14.254
    The IP of BMC which will be used for hardware control.
bmcusername: ADMIN
    The username of bmc.
bmcpassword: admin
    The password of bmc.
mac: 6C:AE:8B:1B:E8:52
    The mac address of the ethernet adapter that will be used to
    deploy OS for the node.
mgt: ipmi
    The management method which will be used to manage the node.
    This node will use ipmi protocol.
netboot: xnba
    The network bootloader that will be used to deploy OS for the node.
provmeth: rhels7.1-x86_64-install-compute
    The osimage that will be deployed to the node.
```

I want to name the node to be **cn1** (Compute Node #1) in xCAT. Then I define this node in xCAT with following command:

```
$mkdef -t node cn1 groups=all,x86_64 arch=x86_64 bmc=10.4.14.254
                    bmcusername=ADMIN bmcpassword=admin mac=6C:AE:8B:1B:E8:52
                    mgt=ipmi netboot=xnba provmeth=rhels7.1-x86_64-install-compute
```

After the define, I can use `lsdef` command to display the defined node:

```
$lsdef cn1
Object name: cn1
    arch=x86_64
    bmc=10.4.14.254
    bmcpassword=admin
```

(continues on next page)

(continued from previous page)

```
bmcusername=ADMIN
groups=all,x86_64
mac=6C:AE:8B:1B:E8:52
mgt=ipmi
netboot=xnba
postbootscripts=otherpkgs
postscripts=syslog,remoteshell,syncfiles
provmethod=rhels7.1-x86_64-install-compute
```

Then I can try to remotely **power on** the node **cn1**:

```
$rpower cn1 on
```

### • group Object

**group** is an object which includes multiple **node object**. When you set **group** attribute for a **node object** to a group name like **x86\_64**, the group **x86\_64** is automatically generated and the node is assigned to the group.

The benefits of using **group object**:

#### – Handle multiple nodes through group

I defined another server **cn2** which is similar with **cn1**, then my group **x86\_64** has two nodes: **cn1** and **cn2**.

```
$ lsdef -t group x86_64
Object name: x86_64
  cons=ipmi
  members=cn1,cn2
```

Then I can power on all the nodes in the group **x86\_64**.

```
$ rpower x86_64 on
```

#### – Inherit attributes from group

If the **group object** of **node object** has certain attribute that **node object** does not have, the node will inherit this attribute from its **group**.

I set the **cons** attribute for the **group object x86\_64**.

```
$ chdef -t group x86_64 cons=ipmi
1 object definitions have been created or modified.

$ lsdef -t group x86_64
Object name: x86_64
  cons=ipmi
  members=cn1,cn2
```

The I can see the **cn1** inherits the attribute **cons** from the group **x86\_64**:

```
$ lsdef cn1
Object name: cn1
  arch=x86_64
  bmc=10.4.14.254
  bmcpassword=admin
  bmcusername=ADMIN
```

(continues on next page)

(continued from previous page)

```
cons=ipmi
groups=all,x86_64
mac=6C:AE:8B:1B:E8:52
mgt=ipmi
netboot=xnba
postbootscripts=otherpkgs
postscripts=syslog,remoteshell,syncfiles
provmethod=rhels7.1-x86_64-install-compute
```

It is useful to define common attributes in **group object** so that newly added node will inherit them automatically. Since the attributes are defined in the **group object**, you don't need to touch the individual nodes attributes.

- Use Regular Expression to generate value for node attributes

This is powerful feature in xCAT that you can generate individual attribute value from node name instead of assigning them one by one. Refer to [Use Regular Expression in xCAT Database Table](#).

- **osimage Object**

An **osimage** object represents an Operating System which can be deployed in xCAT. xCAT always generates several default **osimage** objects for certain Operating System when executing `copycds` command to generate the package repository for the OS.

You can display all the defined **osimage** object:

```
$ lsdef -t osimage
```

Display the detail attributes of one **osimage** named **rhels7.1-x86\_64-install-compute**:

```
$ lsdef -t osimage rhels7.1-x86_64-install-compute
Object name: rhels7.1-x86_64-install-compute
  imagetype=linux
  osarch=x86_64
  osdistrname=rhels7.1-x86_64
  osname=Linux
  osvers=rhels7.1
  otherpkgdir=/install/post/otherpkgs/rhels7.1/x86_64
  pkgdir=/install/rhels7.1/x86_64
  pkglist=/opt/xcat/share/xcat/install/rh/compute.rhels7.pkglist
  profile=compute
  provmethod=install
  synclists=/root/syncfiles.list
  template=/opt/xcat/share/xcat/install/rh/compute.rhels7.tmpl
```

This **osimage** represents a **Linux rhels7.1** Operating System. The package repository is in `/install/rhels7.1/x86_64` and the packages which will be installed is listed in the file `/opt/xcat/share/xcat/install/rh/compute.rhels7.pkglist` ...

I can bind the **osimage** to **node** when I want to deploy **osimage rhels7.1-x86\_64-install-compute** on my **node cn1**:

```
$ nodeset cn1 osimage=rhels7.1-x86_64-install-compute
```

Then in the next network boot, the node **cn1** will start to deploy **rhels7.1**.

- **Manipulating Objects**

You already saw that I used the commands `mkdef`, `lsdef`, `chdef` to manipulate the objects. xCAT has 4 objects management commands to manage all the xCAT objects.

- `mkdef` : create object definitions
- `chdef` : modify object definitions
- `lsdef` : list object definitions
- `rmdef` : remove object definitions

To get the detail usage of the commands, refer to the man page. e.g. `man mkdef`

### Get Into the Detail of the xCAT Objects:

#### node

#### Description

The definition of physical units in the cluster, such as `lpar`, virtual machine, `frame`, `cec`, `hmc`, `switch`.

#### Key Attributes

- **os:**  
The operating system deployed on this node. Valid values: `AIX`, `rhels*`, `rhelc*`, `rhas*`, `centos*`, `SL*`, `fedora*`, `sles*` (where `*` is the version #)
- **arch:**  
The hardware architecture of this node. Valid values: `x86_64`, `ppc64`, `x86`, `ia64`.
- **groups:**  
Usually, there are a set of nodes with some attributes in common, xCAT admin can define a node group containing these nodes, so that the management task can be issued against the group instead of individual nodes. A node can be a member of different groups, so the value of this attributes is a comma-delimited list of groups. At least one group is required to create a node. The new created group names should not be prefixed with “\_\_” as this token has been preserved as the internal group name.
- **mgt:**  
The method to do general hardware management of the node. This attribute can be determined by the machine type of the node. Valid values: `ipmi`, `blade`, `hmc`, `ivm`, `fsp`, `bpa`, `kvm`, `esx`, `rhevm`.
- **mac:**  
The mac address of the network card on the node, which is connected with the installation server and can be used as the network installation device.
- **ip:**  
The IP address of the node.
- **netboot:**  
The type of network boot method for this node, determined by the OS to provision, the architecture and machine type of the node. Valid values:

Arch and Machine Type	OS	valid netboot options
x86, x86_64	ALL	pxe, xnba
ppc64	<=rhel6, <=sles11.3	yaboot
ppc64	>=rhels7, >=sles11.4	grub2,grub2-http,grub2-tftp
ppc64le NonVirtualize	ALL	petitboot
ppc64le PowerKVM Guest	ALL	grub2,grub2-http,grub2-tftp

- **postscripts:**

Comma separated list of scripts, that should be run on this node after diskful installation or diskless boot, finish some system configuration and maintenance work. For installation of RedHat, CentOS, Fedora, the scripts will be run before the reboot. For installation of SLES, the scripts will be run after the reboot but before the init.d process.

- **postbootscripts:**

Comma separated list of scripts, that should be run on this node as a SysV init job on the 1st reboot after installation or diskless boot, finish some system configuration and maintenance work.

- **provmethod:**

The provisioning method for node deployment. Usually, this attribute is an osimage object name.

- **status:**

The current status of the node, which is updated by xCAT. This value can be used to monitor the provision process. Valid values: powering-off, installing, booting/netbooting, booted.

## Use Cases

- Case 1: There is a ppc64le node named “cn1”, the mac of installation NIC is “ca:68:d3:ae:db:03”, the ip assigned is “10.0.0.100”, the network boot method is “grub2”, place it into the group “all”. Use the following command

```
mkdef -t node -o cn1 arch=ppc64 mac="ca:68:d3:ae:db:03" ip="10.0.0.100" netboot=
↪ "grub2" groups="all"
```

- Case 2:

List all the node objects

```
node ls
```

This can also be done with

```
lsdef -t node
```

- Case 3: List the mac of object “cn1”

```
lsdef -t node -o cn1 -i mac
```

- Case 4: There is a node definition “cn1”, modify its network boot method to “yaboot”

```
chdef -t node -o cn1 netboot=yaboot
```

- Case 5: There is a node definition “cn1”, create a node definition “cn2” with the same attributes with “cn1”, except the mac addr(ca:68:d3:ae:db:04) and ip address(10.0.0.101)

*step 1:* write the definition of “cn1” to a stanza file named “cn.stanza”



```
lsdef -z cn1 > /tmp/cn.stanza
```

The content of “/tmp/cn.stanza” will look like

```
# <xCAT data object stanza file>
cn1:
  objtype=node
  groups=all
  ip=10.0.0.100
  mac=ca:68:d3:ae:db:03
  netboot=grub2
```

step 2: modify the “/tmp/cn.stanza” according to the “cn2” attributes

```
# <xCAT data object stanza file>
cn2:
  objtype=node
  groups=all
  ip=10.0.0.101
  mac=ca:68:d3:ae:db:04
  netboot=grub2
```

step 3: create “cn2” definition with “cn.stanza”

```
cat /tmp/cn.stanza |mkdef -z
```

## group

XCAT supports both static and dynamic groups. A static group is defined to contain a specific set of cluster nodes. A dynamic node group is one that has its members determined by specifying a selection criteria for node attributes. If a nodes attribute values match the selection criteria then it is dynamically included as a member of the group. The actual group membership will change over time as nodes have attributes set or unset. This provides flexible control over group membership by defining the attributes that define the group, rather than the specific node names that belong to the group. The selection criteria is a list of `attr<operator>val` pairs that can be used to determine the members of a group, (see below).

**Note :** Dynamic node group support is available in xCAT version 2.3 and later.

In xCAT, the definition of a static group has been extended to include additional attributes that would normally be assigned to individual nodes. When a node is part of a static group definition, it can inherit the attributes assigned to the group. This feature can make it easier to define and manage cluster nodes in that you can generally assign nodes to the appropriate group and then just manage the group definition instead of multiple node definitions. This feature is not supported for dynamic groups.

To list all the attributes that may be set for a group definition you can run

```
lsdef -t group -h
```

When a node is included in one or more static groups, a particular node attribute could actually be stored in several different object definitions. It could be in the node definition itself or it could be in one or more static group definitions. The precedence for determining which value to use is to choose the attribute value specified in the node definition if it is provided. If not, then each static group that the node belongs to will be checked to see if the attribute is set. The first value that is found is the value that is used. The static groups are checked in the order that they are specified in the groups attribute of the node definition.

NOTE : In a large cluster environment it is recommended to focus on group definitions as much as possible and avoid setting the attribute values in the individual node definition. (Of course some attribute values, such as a MAC addresses etc., are only appropriate for individual nodes.) Care must be taken to avoid confusion over which values will be inherited by the nodes.

Group definitions can be created using the `mkdef` command, changed using the `chdef` command, listed using the `lsdef` command and removed using the `rmdef` command.

### Creating a static node group

There are two basic ways to create xCAT static node groups. You can either set the `groups` attribute of the node definition or you can create a group definition directly.

You can set the `groups` attribute of the node definition when you are defining the node with the `mkdef` or `nodeadd` command or you can modify the attribute later using the `chdef` or `nodech` command. For example, if you want a set of nodes to be added to the group “aixnodes”, you could run `chdef` or `nodech` as follows

```
chdef -t node -p -o node01,node02,node03 groups=aixnodes
```

or

```
nodech node01,node02,node03 groups=aixnodes
```

The `-p` (plus) option specifies that “aixnodes” be added to any existing value for the `groups` attribute. The `-p` (plus) option is not supported by `nodech` command.

The second option would be to create a new group definition directly using the `mkdef` command as follows

```
mkdef -t group -o aixnodes members="node01,node02,node03"
```

These two options will result in exactly the same definitions and attribute values being created in the xCAT database.

### Creating a dynamic node group

The selection criteria for a dynamic node group is specified by providing a list of `attr<operator>val` pairs that can be used to determine the members of a group. The valid operators include: `==`, `!=`, `~=` and `!~`. The `attr` field can be any node definition attribute returned by the `lsdef` command. The `val` field in selection criteria can be a simple sting or a regular expression. A regular expression can only be specified when using the `~=` or `!~` operators. See <http://www.perl.com/doc/manual/html/pod/perlre.html> for information on the format and syntax of regular expressions.

Operator descriptions

```
== Select nodes where the attribute value is exactly this value.
!= Select nodes where the attribute value is not this specific value.
~= Select nodes where the attribute value matches this regular expression.
!~ Select nodes where the attribute value does not match this regular expression.
```

The selection criteria can be specified using one or more `-w attr<operator>val` options on the command line.

If the `val` field includes spaces or any other characters that will be parsed by shell then the `attr<operator>val` needs to be quoted.

For example, to create a dynamic node group called “mygroup”, where the hardware control point is “hmc01” and the partition profile is not set to service

```
mkdef -t group -o mygroup -d -w hcp==hmc01 -w pprofile!=service
```

To create a dynamic node group called “pslesnodes”, where the operating system name includes “sles” and the architecture includes “ppc”

```
mkdef -t group -o pslesnodes -d -w os=~sles[0-9]+ -w arch=~ppc
```

To create a dynamic node group called nonpbladenodes where the node hardware management method is not set to blade and the architecture does not include ppc

```
mkdef -t group -o nonpbladenodes -d -w mgt!=blade -w 'arch!~ppc'
```

## osimage

### Description

A logical definition of image which can be used to provision the node.

### Key Attributes

- **imagetype:**  
The type of operating system this definition represents (linux, AIX).
- **osarch:**  
The hardware architecture of the nodes this image supports. Valid values: x86\_64, ppc64, ppc64le.
- **osvers:**  
The Linux distribution name and release number of the image. Valid values: rhels\*, rhelc\*, rhas\*, centos\*, SL\*, fedora\*, sles\* (where \* is the version #).
- **pkgdir:**  
The name of the directory where the copied OS distro content are stored.
- **pkglist:**  
The fully qualified name of a file, which contains the list of packages shipped in Linux distribution ISO which will be installed on the node.
- **otherpkgdir**  
When xCAT user needs to install some additional packages not shipped in Linux distribution ISO, those packages can be placed in the directory specified in this attribute. xCAT user should take care of dependency problems themselves, by putting all the dependency packages not shipped in Linux distribution ISO in this directory and creating repository in this directory.
- **otherpkglist:**  
The fully qualified name of a file, which contains the list of user specified additional packages not shipped in Linux distribution ISO which will be installed on the node.
- **template:**  
The fully qualified name of the template file that will be used to create the OS installer configuration file for stateful installation (e.g. kickstart for RedHat, autoyast for SLES and preseeds for Ubuntu).

## Use Cases

- Case 1:

List all the osimage objects

```
lsdef -t osimage
```

- Case 2:

Create a osimage definition “customized-rhels7-ppc64-install-compute” based on an existing osimage “rhels7-ppc64-install-compute”, the osimage “customized-rhels7-ppc64-install-compute” will inherit all the attributes of “rhels7-ppc64-install-compute” except installing the additional packages specified in the file “/tmp/otherpkg.list”:

*step 1* : write the osimage definition “rhels7-ppc64-install-compute” to a stanza file “osimage.stanza”

```
lsdef -z -t osimage -o rhels7-ppc64-install-compute > /tmp/osimage.stanza
```

The content will look like

```
# <xCAT data object stanza file>

rhels7-ppc64-install-compute:
  objtype=osimage
  imagetype=linux
  osarch=ppc64
  osdistrname=rhels7-ppc64
  osname=Linux
  osvers=rhels7
  otherpkgdir=/install/post/otherpkgs/rhels7/ppc64
  pkgdir=/install/rhels7/ppc64
  pkglist=/opt/xcats/share/xcats/install/rh/compute.rhels7.pkglist
  profile=compute
  provmethod=install
  template=/opt/xcats/share/xcats/install/rh/compute.rhels7.tpl
```

*step 2* : modify the stanza file according to the attributes of “customized-rhels7-ppc64-install-compute”

```
# <xCAT data object stanza file>

customized-rhels7-ppc64-install-compute:
  objtype=osimage
  imagetype=linux
  osarch=ppc64
  osdistrname=rhels7-ppc64
  osname=Linux
  osvers=rhels7
  otherpkglist=/tmp/otherpkg.list
  otherpkgdir=/install/post/otherpkgs/rhels7/ppc64
  pkgdir=/install/rhels7/ppc64
  pkglist=/opt/xcats/share/xcats/install/rh/compute.rhels7.pkglist
  profile=compute
  provmethod=install
  template=/opt/xcats/share/xcats/install/rh/compute.rhels7.tpl
```

*step 3* : create the osimage “customized-rhels7-ppc64-install-compute” from the stanza file

```
cat /tmp/osimage.stanza |mkdef -z
```

## xCAT Database

All of the xCAT Objects and Configuration data are stored in xCAT database. By default, xCAT uses **SQLite** - an OS contained simple database engine. The powerful open source database engines like MySQL, MariaDB, PostgreSQL are also supported for a large cluster.

xCAT defines about 70 tables to store different data. You can get the xCAT database definition from file `/opt/xcat/lib/perl/xcAT/Schema.pm`.

You can run `tabdump` command to get all the xCAT database tables. Or run `tabdump -d <tablename>` or `man <tablename>` to get the detail information on columns and table definitions.

```
$ tabdump
$ tabdump site
$ tabdump -d site
$ man site
```

For a complete reference, see the man page for `xcatdb`: `man xcatdb`.

### The tables in xCAT:

- **site table**

Global settings for the whole cluster. This table is different from the other tables. Each entry in **site table** is a `key=>value` pair. Refer to the [Global Configuration](#) page for the major global attributes or run `man site` to get all global attributes.

- **policy table**

Controls who has authority to run specific xCAT operations. It is the Access Control List (ACL) in xCAT.

- **passwd table**

Contains default userids and passwords for xCAT to access cluster components. In most cases, xCAT will also set the userid/password in the relevant component (Generally for SP like bmc, fsp.) when it is being configured or installed. The default userids/passwords in passwd table for specific cluster components can be overridden by the columns in other tables, e.g. `mpa`, `ipmi`, `ppchcp`, etc.

- **networks table**

Contains the network definitions in the cluster.

You can manipulate the networks through `*def` command against the **network object**.

```
$ lsdef -t network
```

- ...

### Manipulate xCAT Database Tables

xCAT offers 5 commands to manipulate the database tables:

- **tabdump**

Displays the header and all the rows of the specified table in CSV (comma separated values) format.

- **tabedit**

Opens the specified table in the user’s editor, allows them to edit any text, and then writes changes back to the database table. The table is flattened into a CSV (comma separated values) format file before giving it to the editor. After the editor is exited, the CSV file will be translated back into the database format.

- **tabgrep**

List table names in which an entry for the given node appears.

- **dumpxCATdb**

Dumps all the xCAT db tables to CSV files under the specified directory, often used to backup the xCAT database for xCAT reinstallation or management node migration.

- **restorexCATdb**

Restore the xCAT db tables from the CSV files under the specified directory.

### Advanced Topic: How to use Regular Expression in xCAT tables:

## Groups and Regular Expressions in Tables

### Using Regular Expressions in the xCAT Tables

The xCAT database has a number of tables, some with rows that are keyed by node name (such as `noderes` and `nodehm`) and others that are not keyed by node name (for example, the `policy` table). The tables that are keyed by node name have some extra features that enable a more template-based style to be used:

Any group name can be used in lieu of a node name in the node field, and that row will then provide “default” attribute values for any node in that group. A row with a specific node name can then override one or more attribute values for that specific node. For example, if the `nodehm` table contains

```
#node,power,mgt,cons,termserver,termport,conserver,serialport,serialspeed,serialflow,
↪getmac,cmdmapping,comments,disable
"mygroup",,"ipmi",,,,,,"19200",,,,,
"node1",,,,,,,,"115200",,,,,
```

In the above example, the node group called “mygroup” sets `mgt=ipmi` and `serialspeed=19200`. Any nodes that are in this group will have those attribute values, unless overridden. For example, if “node2” is a member of “mygroup”, it will automatically inherit these attribute values (even though it is not explicitly listed in this table). In the case of “node1” above, it inherits `mgt=ipmi`, but overrides the `serialspeed` to be 115200, instead of 19200. A useful, typical way to use this capability is to create a node group for your nodes and for all the attribute values that are the same for every node, set them at the group level. Then you only have to set attributes for each node that vary from node to node.

xCAT extends the group capability so that it can also be used for attribute values that vary from node to node in a very regular pattern. For example, if in the `ipmi` table you want the `bmc` attribute to be set to whatever the nodename is with “-bmc” appended to the end of it, then use this in the `ipmi` table

```
#node,bmc,bmcport,taggedvlan,bmcid,username,password,comments,disable
"compute",,"/z/-bmc/",,,,,,
```

In this example, “compute” is a node group that contains all of the compute nodes. The 2nd attribute (`bmc`) is a regular expression that is similar to a substitution pattern. The 1st part `\z` matches the end of the node name and substitutes `-bmc`, effectively appending it to the node name.

Another example is if “node1” is assigned the IP address “10.0.0.1”, node2 is assigned the IP address “10.0.0.2”, etc., then this could be represented in the `hosts` table with the single row

A more involved example is with the `vm` table. If your `kvm` nodes have node names `c01f01x01v01`, `c01f02x03v04`, etc., and the `kvm` host names are `c01f01x01`, `c01f02x03`, etc., then you might have an `vm` table like

[illegible]

kvms

$$|\backslash D+(\backslash d+)\backslash D+(\backslash d+)\backslash D+(\backslash d+)\backslash D+(\backslash d+)|c(\$1)f(\$2)x(\$3)|$$

```
| \D+(\d+)\D+(\d+)\D+(\d+)\D+(\d+) | dir:///install/vms/vm($4+0) |
```

Just as the explained above, when the node definition “c01f02x03v04” is created with

```
# mkdef -t node -o c01f02x03v04 groups=kvms
1 object definitions have been created or modified.
```

```
# lsdef c01f02x03v04
Object name: c01f02x03v04
groups=kvms
```

51

(continued from previous page)

```
postbootscripts=otherpkgs
postscripts=syslog,remoteshell,syncfiles
vmcpus=2
vmhost=c01f02x03
vmmemory=3072
vmnicnicmodel=virtio
vmnics=virbr2
vmstorage=dir:///install/vms/vm4
```

See [perlre](#) for more information on perl regular expressions.

## Easy Regular expressions

As of xCAT 2.8.1, you can use a modified version of the regular expression support described in the previous section. You do not need to enter the node information (1st part of the expression), it will be derived from the input nodename. You only need to supply the 2nd part of the expression to determine the value to give the attribute.

For example:

If node1 is assigned the IP address 10.0.0.1, node2 is assigned the IP address 10.0.0.2, etc., then this could be represented in the hosts table with the single row:

Using full regular expression support you would put this in the hosts table.

```
chdef -t group compute ip="|node(\d+)|10.0.0.($1+0)|"
tabdump hosts
#node,ip,hostnames,otherinterfaces,comments,disable
"compute","|node(\d+)|10.0.0.($1+0)|",,,,
```

Using easy regular expression support you would put this in the hosts table.

```
chdef -t group compute ip="|10.0.0.($1+0)|"
tabdump hosts
#node,ip,hostnames,otherinterfaces,comments,disable
"compute","|10.0.0.($1+0)|",,,,
```

In the easy regx example, the expression only has the 2nd part of the expression from the previous example. xCAT will evaluate the node name, matching the number part of the node name, and create the 1st part of the expression. The 2nd part supplied is what value to give the ip attribute. The resulting output is the same.

## Regular Expression Helper Functions

xCAT provides several functions that can simplify regular expressions.

### a2idx ASCII Character to Index

Usage: a2idx(character)

Turns a single character into a 1-indexed index. 'a' maps to 1 and 'z' maps to 26.

### a2zidx ASCII Character to 0-Index

Usage: a2zidx(character)

Turns a single character into a 0-indexed index. 'a' maps to 0 and 'z' maps to 25.



### dim2idx Dimensions to Index

Usage: `dim2idx(value, [count, value...])`

Converts dimensions (such as row, column, chassis, etc) into an index. An example system consists of 8 racks, two rows with four columns each.

row1-col1	row1-col2	row1-col3	row1-col4
row2-col1	row2-col2	row2-col3	row2-col4

To obtain the rack index, use `|row(\d+)-col(\d+)|(dim2idx($1, 4, $2))|`. This maps the racks to:

1	2	3	4
5	6	7	8

Note that the size of the highest dimension (2 rows) is not needed, and all values are one-indexed.

If each rack contains 20 nodes, use `|row(\d+)-col(\d+)-node(\d+)|(dim2idx($1, 4, $2, 20, $3))` to determine a node index (useful for determining IP addresses).

### skip Skip indices

Usage: `skip(index, skiplist)`

Return an index with certain values skipped. The skip list uses the format `start[:count][,start[:count]. .]`. Using the example above, to skip racks 3 and 4, use:

`|row(\d+)-col(\d+)|(skip(dim2idx($1, 4, $2), '3:2'))|`

The result would be:

1	2		
3	4	5	6

### ipadd Add to an IP address

Usage: `ipadd(octet1, octet2, octet3, octet4, toadd, skipstart, skipend)`

This function is useful when you need to cross octets. Optionally skip addresses at the start and end of octets (like .0 or .255 - technically those are valid IP addresses, but sometimes software makes poor assumptions about which broadcast and gateway addresses).

### Verify your regular expression

After you create your table with regular expression, make sure they are evaluating as you expect.

```
lsdef node1 | grep ip
ip=10.0.0.1
```

## Global Configuration

All the xCAT global configurations are stored in site table, xCAT Admin can adjust the configuration by modifying the site attribute with `tabedit`.

This section only presents some key global configurations, for the complete reference on the xCAT global configurations, refer to the `tabdump -d site`.

## Database Attributes

- `excludenodes`: A set of comma separated nodes and/or groups that would automatically be subtracted from any `noderange`, it can be used for excluding some failed nodes from any xCAT command. See [noderange](#) for details on supported formats.
- `nodestatus`: If set to `n`, the `nodelist.status` column will not be updated during the node deployment, node discovery and power operations. The default is to update.

## DHCP Attributes

- `dhcpinterfaces`: The network interfaces DHCP should listen on. If it is the same for all nodes, use a simple comma-separated list of NICs. To specify different NICs for different nodes

```
xcatmn|eth1,eth2;service|bond0.
```

In this example `xcatmn` is the name of the xCAT MN, and DHCP there should listen on `eth1` and `eth2`. On all of the nodes in group `service` DHCP should listen on the `bond0` nic.

- `dhcplease`: The lease time for the dhcp client. The default value is 43200.
- `managedaddressmode`: The mode of networking configuration during node provision. If set to `static`, the network configuration will be configured in static mode based on the node and network definition on MN. If set to `dhcp`, the network will be configured with dhcp protocol. The default is `dhcp`.

## DNS Attributes

- `domain`: The DNS domain name used for the cluster.
- `forwarders`: The DNS servers at your site that can provide names outside of the cluster. The `makedns` command will configure the DNS on the management node to forward requests it does not know to these servers. **Note** that the DNS servers on the service nodes will ignore this value and always be configured to forward requests to the management node.
- `master`: The hostname of the xCAT management node, as known by the nodes.
- `nameservers`: A comma delimited list of DNS servers that each node in the cluster should use. This value will end up in the nameserver settings of the `/etc/resolv.conf` on each node. It is common (but not required) to set this attribute value to the IP addr of the xCAT management node, if you have set up the DNS on the management node by running `makedns`. In a hierarchical cluster, you can also set this attribute to `<xcatmaster>` to mean the DNS server for each node should be the node that is managing it (either its service node or the management node).
- `dnsinterfaces`: The network interfaces DNS server should listen on. If it is the same for all nodes, use a simple comma-separated list of NICs. To specify different NICs for different nodes

```
xcatmn | eth1,eth2;service | bond0.
```

In this example xcatmn is the name of the xCAT MN, and DNS there should listen on eth1 and eth2. On all of the nodes in group service DNS should listen on the bond0 nic.

**NOTE:** if using this attribute to block certain interfaces, make sure the ip that maps to your hostname of xCAT MN is not blocked since xCAT needs to use this ip to communicate with the local DNS server on MN.

## Install/Deployment Attributes

- **installdir:** The local directory name used to hold the node deployment packages.
- **runbootscripts:** If set to **yes** the scripts listed in the **postbootscripts** attribute in the **osimage** and **postscripts** tables will be run during each reboot of stateful (diskful) nodes. This attribute has no effect on stateless nodes. Run the following command after you change the value of this attribute

```
updatenode <nodes> -P setuppostbootscripts
```

- **precreatemybootscripts:** (**yes/1** or **no/0**). Default is **no**. If **yes**, it will instruct xCAT at **nodeset** and **updatenode** time to query the db once for all of the nodes passed into the cmd and create the mybootscript file for each node, and put them in a directory of **tftpdir**(such as: **/tftpboot**). If **no**, it will not generate the mybootscript file in the **tftpdir**.
- **xcatdebugmode:** the xCAT debug level. xCAT provides a batch of techniques to help user debug problems while using xCAT, especially on OS provision, such as collecting logs of the whole installation process and accessing the installing system via ssh, etc. These techniques will be enabled according to different xCAT debug levels specified by 'xcatdebugmode', currently supported values:

```
'0':  disable debug mode
'1':  enable basic debug mode
'2':  enable expert debug mode
```

For the details on 'basic debug mode' and 'expert debug mode', refer to xCAT documentation.

## Remoteshell Attributes

- **sshbetweennodes:** Comma separated list of groups of compute nodes to enable passwordless root ssh during install, or **xdsh -K**. Default is **ALLGROUPS**. Set to **NOGROUPS** if you do not wish to enable it for any group of compute nodes. If using the zone table, this attribute is not used.

## Services Attributes

- **consoleondemand:** When set to **yes**, conserver connects and creates the console output only when the user opens the console. Default is **no** on Linux, **yes** on AIX.
- **timezone:** The timezone for all the nodes in the cluster(e.g. **America/New\_York**).
- **tftpdir:** tftp directory path. Default is **/tftpboot**.
- **tftpflags:** The flags used to start tftpd. Default is **-v -l -s /tftpboot -m /etc/tftpmmapfile4xcat.conf** if **tftplflags** is not set.

## Virtualization Attributes

- **persistkvmguests:** Keep the kvm definition on the kvm hypervisor when you power off the kvm guest node. This is useful for you to manually change the kvm xml definition file in `virsh` for debugging. Set anything means enable.

## xCAT Daemon attributes

- **xcatdport:** The port used by xcatd daemon for client/server communication.
- **xcatiport:** The port used by xcatd to receive installation status updates from nodes.
- **xcatlport:** The port used by xcatd command log writer process to collect command output.
- **xcatsslversion:** The ssl version by xcatd. Default is SSLv3.
- **xcatsslciphers:** The ssl cipher by xcatd. Default is 3DES.

## Network Planning

For a cluster, several networks are necessary to enable the cluster management and production.

- **Management network**

This network is used by the management node to install and manage the OS of the nodes. The MN and in-band NIC of the nodes are connected to this network. If you have a large cluster with service nodes, sometimes this network is segregated into separate VLANs for each service node.

Following network services need be set up in this network to supply the OS deployment, application install/configuration service.

- **DNS(Domain Name Service)**

The dns server, usually the management node or service node, provides the domain name service for the entire cluster.

- **HTTP(HyperText Transfer Protocol)**

The http server, usually the management node or service node, acts as the download server for the `initrd` and kernel, the configuration file for the installer and repository for the online installation.

- **DHCP(Dynamic Host Configuration Protocol)**

The dhcp server, usually the management node or service node, provides the dhcp service for the entire cluster.

- **TFTP(Trivial File Transfer Protocol)**

The tftp server, usually the management node or service node, acts as the download server for bootloader binaries, bootloader configuration file, `initrd` and kernel.

- **NFS(Network File System)**

The NFS server, usually the management node or service node, provides the file system sharing between the management node and service node, or persistent file system support for the stateless node.

- **NTP(Network Time Protocol)**

The NTP server, usually the management node or service node, provide the network time service for the entire cluster.

- **Service network**

This network is used by the management node to control the nodes out of band via the SP like BMC, FSP. If the BMCs are configured in shared mode<sup>1</sup>, then this network can be combined with the management network.

- **Application network**

This network is used by the applications on the compute nodes. Usually an IB network for HPC cluster.

- **Site (Public) network** This network is used to access the management node and sometimes for the compute nodes to provide services to the site.

From the system management perspective, the **Management network** and **Service network** are necessary to perform the hardware control and OS deployment.

#### xCAT Network Planning for a New Cluster:

### xCAT Network Planning

Before setting up your cluster, there are a few things that are important to think through first, because it is much easier to go in the direction you want right from the beginning, instead of changing course midway through.

### Do You Need Hierarchy in Your Cluster?

#### Service Nodes

For very large clusters, xCAT has the ability to distribute the management operations to service nodes. This allows the management node to delegate all management responsibilities for a set of compute or storage nodes to a service node so that the management node doesn't get overloaded. Although xCAT automates a lot of the aspects of deploying and configuring the services, it still adds complexity to your cluster. So the question is: at what size cluster do you need to start using service nodes? The exact answer depends on a lot of factors (mgmt node size, network speed, node type, OS, frequency of node deployment, etc.), but here are some general guidelines for how many nodes a single management node (or single service node) can handle:

- **[Linux]:**
  - Stateful or Stateless: 500 nodes
  - Statelite: 250 nodes
- **[AIX]:**
  - 150 nodes

These numbers can be higher (approximately double) if you are willing to “stage” the more intensive operations, like node deployment.

Of course, there are some reasons to use service nodes that are not related to scale, for example, if some of your nodes are far away (network-wise) from the mgmt node.

<sup>1</sup> shared mode: In “Shared” mode, the BMC network interface and the in-band network interface will share the same network port.

## Network Hierarchy

For large clusters, you may want to divide the management network into separate subnets to limit the broadcast domains. (Service nodes and subnets don't have to coincide, although they often do.) xCAT clusters as large as 3500 nodes have used a single broadcast domain.

Some cluster administrators also choose to sub-divide the application interconnect to limit the network contention between separate parallel jobs.

## Design an xCAT Cluster for High Availability

Everyone wants their cluster to be as reliable and available as possible, but there are multiple ways to achieve that end goal. Availability and complexity are inversely proportional. You should choose an approach that balances these 2 in a way that fits your environment the best. Here's a few choices in order of least complex to more complex.

### Service Node Pools With No HA Software

**Service node pools** is an xCAT approach in which more than one service node (SN) is in the broadcast domain for a set of nodes. When each node netboots, it chooses an available SN by which one responds to its DHCP request 1st. When services are set up on the node (e.g. DNS), xCAT configures the services to use at that SN and one other SN in the pool. That way, if one SN goes down, the node can keep running, and the next time it netboots it will automatically choose another SN.

This approach is most often used with stateless nodes because that environment is more dynamic. It can possibly be used with stateful nodes (with a little more effort), but that type of node doesn't netboot nearly as often so a more manual operation (snmove) is needed in that case move a node to different SNs.

It is best to have the SNs be as robust as possible, for example, if they are diskful, configure them with at least 2 disks that are RAID'ed together.

In smaller clusters, the management node (MN) can be part of the SN pool with one other SN.

In larger clusters, if the network topology dictates that the MN is only for managing the SNs (not the compute nodes), then you need a plan for what to do if the MN fails. Since the cluster can continue to run if the MN is down temporarily, the plan could be as simple as have a backup MN w/o any disks. If the primary MN fails, move its RAID'ed disks to the backup MN and power it on.

### HA Management Node

If you want to use HA software on your management node to synchronize data and fail over services to a backup MN, see [TODO Highly\_Available\_Management\_Node], which discusses the different options and the pros and cons.

It is important to note that some HA-related software like DRDB, Pacemaker, and Corosync is not officially supported by IBM, meaning that if you have a problem specifically with that software, you will have to go to the open source community or another vendor to get a fix.

## HA Service Nodes

When you have NFS-based diskless (statelite) nodes, there is sometimes the motivation make the NFS serving highly available among all of the service nodes. This is not recommended because it is a very complex configuration. In our opinion, the complexity of this setup can nullify much of the availability you hope to gain. If you need your compute nodes to be highly available, you should strongly consider stateful or stateless nodes.

If you still have reasons to pursue HA service nodes:

- For **[AIX]**, see [TODO XCAT\_HASN\_with\_GPFS]
- For **[Linux]**, a couple prototype clusters have been set up in which the NFS service on the SNs is provided by GPFS CNFS (Clustered NFS). A howto is being written to describe the setup as an example. Stay tuned.

## xCAT Cluster OS Running Type

Whether a node is a physical server or a virtual machine, it needs to run an Operating System to support user applications. Generally, the OS is installed in the hard disk of the compute node. But xCAT also support the type that running OS in the RAM.

This section gives the pros and cons of each OS running type, and describes the cluster characteristics that will impact from each.

### Stateful (diskful)

Traditional cluster with OS on each node's local disk.

- Main advantage

This approach is familiar to most admins, and they typically have many years of experience with it.

- Main disadvantage

Admin has to manage all of the individual OS copies, has to face the failure of hard disk. For certain application which requires all the compute nodes have exactly same state, this is also changeable for admin.

### Stateless (diskless)

Nodes boot from a RAMdisk OS image downloaded from the xCAT mgmt node or service node at boot time.

- Main advantage

Central management of OS image, but nodes are not tethered to the mgmt node or service node it booted from. Whenever you need a new OS for the node, just reboot the node.

- Main disadvantage

You can't use a large image with many different applications in the image for varied users, because it uses too much of the node's memory to store the ramdisk. (To mitigate this disadvantage, you can put your large application binaries and libraries in shared storage to reduce the ramdisk size. This requires some manual configuration of the image).

Each node can also have a local "scratch" disk for swap, /tmp, /var, log files, dumps, etc. The purpose of the scratch disk is to provide a location for files that are written to by the node that can become quite large or for files that you don't want to disappear when the node reboots. There should be nothing put on the scratch disk that represents the node's "state", so that if the disk fails you can simply replace it and reboot the node. A scratch disk would typically be used for situations like: job

scheduling preemption is required (which needs a lot of swap space), the applications write large temp files, or you want to keep gpfs log or trace files persistently. (As a partial alternative to using the scratch disk, customers can choose to put `/tmp/var/tmp`, and log files (except GPFS logs files) in GPFS, but must be willing to accept the dependency on GPFS). This can be done by enabling the 'localdisk' support. For the details, refer to the section [TODO Enabling the localdisk Option].

## OSimage Definition

The attribute **provmethod** is used to identify that the osimage is diskful or diskless:

```
$ lsdef -t osimage rhels7.1-x86_64-install-compute -i provmethod
Object name: rhels7.1-x86_64-install-compute
  provmethod=install
```

### install:

Diskful

### netboot:

Diskless

## 1.4.2 Manage Clusters

The following provides detailed information to help start managing your cluster using xCAT.

The sections are organized based on hardware architecture.

### IBM POWER LE / OpenPOWER

Most of the content is general information for xCAT, the focus and examples are for management of IBM OpenPOWER servers.

#### IBM OpenPOWER Servers

- based on POWER8 Processor Technology is IPMI managed
- based on POWER9 Processor Technology is OpenBMC managed

## Configure xCAT

After installing xCAT onto the management node, configure some basic attributes for your cluster into xCAT.

### Set attributes in the site table

1. Verify the following attributes have been correctly set in the xCAT `site` table.
  - domain
  - forwarders
  - master<sup>1</sup>
  - nameservers

---

<sup>1</sup> The value of the `master` attribute in the site table should be set as the IP address of the management node responsible for the compute node.



For more information on the keywords, see the DHCP ATTRIBUTES in the [site](#) table.

If the fields are not set or need to be changed, use the xCAT chdef command:

```
chdef -t site domain="domain_string"
chdef -t site fowardsers="forwarders"
chdef -t site master="xcat_master_ip"
chdef -t site nameservers="nameserver1,nameserver2,etc"
```

## Initialize DNS services

1. Initialize the DNS<sup>2</sup> services on the xCAT Management Node:

```
makedns -n
```

Verify DNS is working by running nslookup against your Management Node:

```
nslookup <management_node_hostname>
```

For more information on DNS, refer to [Cluster Name Resolution](#)

## Set attributes in the networks table

1. Display the network settings defined in the xCAT networks table using: `tabdump networks`

```
#netname,net,mask,mgtifname,gateway,dhcpserver,tftpserver,nameservers,ntpserver,
↪logserver,
dynamicrange,staticrange,staticrangeincrement,nodehostname,ddnsdomain,vlanid,domain,
↪mtu,
comments,disable
"10_0_0_0-255_0_0_0","10.0.0.0","255.0.0.0","eth0","10.0.0.101","10.4.27.5",,,,,,
↪,,,,,
```

A default network is created for the detected primary network using the same netmask and gateway. There may be additional network entries in the table for each network present on the management node where xCAT is installed.

2. To define additional networks, use one of the following options:

- **[Recommended]** Use `mkdef` to create/update an entry into networks table.

To create a network entry for 192.168.X.X/16 with a gateway of 192.168.1.254:

```
mkdef -t network -o net1 net=192.168.0.0 mask=255.255.0.0 gateway=192.168.1.254
```

- Use the `tabedit` command to modify the networks table directly in an editor:

```
tabedit networks
```

- Use the `makenetworks` command to automatically generate a entry in the networks table:

```
makenetworks
```

<sup>2</sup> Setting up name resolution and the ability to have hostname resolved to IP addresses is **required** for xCAT.

### 3. Verify the network statements

**Domain** and **nameserver** attributes must be configured in the **networks** table or in the **site** table for xCAT to function properly.

## Initialize DHCP services

Configure DHCP to listen on different network interfaces [**Optional**]

The default behavior of xCAT is to configure DHCP to listen on all interfaces defined in the **networks** table.

The **dhcpinterfaces** keyword in the **site** table allows administrators to limit the interfaces that DHCP will listen over. If the management node has 4 interfaces, (eth0, eth1, eth2, and eth3), and you want DHCP to listen only on “eth1” and “eth3”, set **dhcpinterfaces** using:

```
chdef -t site dhcpinterfaces="eth1,eth3"
```

To set “eth1” and “eth3” on the management node and “bond0” on all nodes in the **nodegroup=“service”**, set **dhcpinterfaces** using:

```
chdef -t site dhcpinterfaces="eth1,eth3;service|bond0"
```

or, to explicitly identify the management node with hostname **xcatmn**:

```
chdef -t site dhcpinterfaces="xcatmn|eth1,eth3;service|bond0"
```

## noboot

For the *IBM OpenPOWER S822LC for HPC (“Minsky”)* nodes, the BMC and compute “eth0” share the left-side integrated ethernet port and compute “eth1” is the right-side integrated ethernet port. For these servers, it is recommended to use two physical cables allowing the BMC port to be dedicated and “eth1” used by the OS. When an open range is configured on the two networks, the xCAT Genesis kernel will be sent to the BMC interface and causes problems during hardware discovery. To support this scenario, on the xCAT management node, if “eth1” is connected to the BMC network and “eth3” is connected to the compute network, disable genesis boot for the BMC network by setting **:noboot** in **dhcpinterfaces** using:

```
chdef -t site dhcpinterfaces="eth1:noboot,eth3"

# run the mknb command to remove the genesis
# configuration file for the specified network
mknb ppc64
```

For more information, see **dhcpinterfaces** keyword in the *site* table.

After making any DHCP changes, create a new DHCP configuration file with the networks defined using the **makedhcp** command.

```
makedhcp -n
```

## Configure passwords

1. Configure the system password for the root user on the compute nodes. This password can be provided in encrypted or clear text form using the `chtab` command.

- Clear text:

```
chtab key=system passwd.username=root passwd.password=abc123
```

- Encrypted using openssl:

```
chtab key=system passwd.username=root passwd.password=`openssl passwd -6 abc123`
```

2. Configure the passwords for Management modules of the compute nodes.

- For OpenBMC managed systems:

```
chtab key=openbmc passwd.username=root passwd.password=openBmc
```

- For IPMI/BMC managed systems:

```
chtab key=ipmi passwd.username=ADMIN passwd.password=admin
```

- For HMC managed systems:

```
chtab key=hmc passwd.username=hscroot passwd.password=abc123
```

If the username/password is different for multiple HMCs, set the username and password attribute for each HMC node object in xCAT

- For Blade managed systems:

```
chtab key=blade passwd.username=USERID passwd.password=PASSWORD
```

- For FSP/BPA (Flexible Service Processor/Bulk Power Assembly) the factory default passwords must be changed before running commands against them.

```
rspconfig frame general_passwd=general,<newpassword>
rspconfig frame admin_passwd=admin,<newpassword>
rspconfig frame HMC_passwd=,<newpassword>
```

3. If using the xCAT REST API

1. Create a non-root user that will be used to make the REST API calls.

```
useradd xcatws
passwd xcatws # set the password
```

2. Create an entry for the user into the xCAT passwd table.

```
chtab key=xcat passwd.username=xcatws passwd.password=<xcatws_password>
```

3. Set a policy in the xCAT policy table to allow the user to make calls against xCAT.

```
mkdef -t policy 6 name=xcatws rule=allow
```

When making calls to the xCAT REST API, pass in the credentials using the following attributes: `userName` and `userPW`

## Hardware Discovery & Define Node

In order to manage machines using xCAT, the machines need to be defined as xCAT `node` objects in the database. The *xCAT Objects* documentation describes the process for manually creating `node` objects one by one using the xCAT `mkdef` command. This is valid when managing a small sized cluster but can be error prone and cumbersome when managing large sized clusters.

xCAT provides several *automatic hardware discovery* methods to assist with hardware discovery by helping to simplify the process of detecting service processors (SP) and collecting various server information. The following are methods that xCAT supports:

### MTMS-based Discovery

MTMS stands for **M**achine **T**ype/**M**odel and **S**erial. This is one way to uniquely identify each physical server.

MTMS-based hardware discovery assumes the administrator has the model type and serial number information for the physical servers and a plan for mapping the servers to intended hostname/IP addresses.

#### Overview

1. Automatically search and collect MTMS information from the servers
2. Write **discovered-bmc-nodes** to xCAT (recommended to set different BMC IP address)
3. Create **predefined-compute-nodes** to xCAT providing additional properties
4. Power on the nodes which triggers xCAT hardware discovery engine

#### Pros

- Limited effort to get servers defined using xCAT hardware discovery engine

#### Cons

- When compared to switch-based discovery, the administrator needs to create the **predefined-compute-nodes** for each of the **discovered-bmc-nodes**. This could become difficult for a large number of servers.

### Verification

Before starting hardware discovery, ensure the following is configured to make the discovery process as smooth as possible.

#### Password Table

In order to communicate with IPMI-based hardware (with BMCs), verify that the xCAT `passwd` table contains an entry for `ipmi` which defines the default username and password to communicate with the IPMI-based servers.

```
tabdump passwd | grep ipmi
```

If not configured, use the following command to set `username=ADMIN` and `password=admin`.

```
chtab key=ipmi passwd.username=ADMIN passwd.password=admin
```

## Genesis Package

The **xCAT-genesis** packages provides the utility to create the genesis network boot rootimage used by xCAT when doing hardware discovery. It should be installed during the xCAT install and would cause problems if missing.

Verify that the `genesis-scripts` and `genesis-base` packages are installed:

- [RHEL/SLES]:

```
rpm -qa | grep -i genesis
```

- [Ubuntu]:

```
dpkg -l | grep -i genesis
```

If missing:

1. Install them from the `xcat-dep` repository using the Operating Specific package manager (`yum`, `zypper`, `apt-get`, etc)

- [RHEL]:

```
yum install xCAT-genesis
```

- [SLES]:

```
zypper install xCAT-genesis
```

- [Ubuntu]:

```
apt-get install xCAT-genesis
```

2. Create the network boot rootimage with the following command: `mknb ppc64`.

The genesis kernel should be copied to `/tftpboot/xcat`.

## Discovery

When the IPMI-based servers are connected to power, the BMCs will boot up and attempt to obtain an IP address from an open range dhcp server on your network. In the case for xCAT managed networks, xCAT should be configured serve an open range dhcp IP addresses with the `dynamicrange` attribute in the `networks` table.

When the BMCs have an IP address and is pingable from the xCAT management node, administrators can discover the BMCs using the xCAT's `bmcdiscover` command and obtain basic information to start the hardware discovery process.

xCAT Hardware discover uses the xCAT genesis kernel (diskless) to discover additional attributes of the compute node and automatically populate the node definitions in xCAT.

## Set static BMC IP using different IP address (recommended)

The following example outlines the MTMS based hardware discovery for a single IPMI-based compute node.

Compute Node Information	Value
Model Type	8247-221
Serial Number	10112CA
Hostname	cn01
IP address	10.0.101.1

The BMC IP address is obtained by the open range dhcp server and the plan in this scenario is to change the IP address for the BMC to a static IP address in a different subnet than the open range addresses. The static IP address in this example is in the same subnet as the open range to simplify the networking configuration on the xCAT management node.

BMC Information	Value
IP address - dhcp	50.0.100.1
IP address - static	50.0.101.1

1. Detect the BMCs and add the node definitions into xCAT.

Use the `bmcdiscover` command to discover the BMCs responding over an IP range and write the output into the xCAT database. This discovered BMC node is used to control the physical server during hardware discovery and will be deleted after the correct server node object is matched to a pre-defined node. You **must** use the `-w` option to write the output into the xCAT database.

To discover the BMC with an IP address range of 50.0.100.1-100:

```
bmcdiscover --range 50.0.100.1-100 -z -w
```

The discovered nodes will be written to xCAT database. The discovered BMC nodes are in the form **node-model\_type-serial**. To view the discovered nodes:

```
lsdef /node-.*
```

**Note:** The `bmcdiscover` command will use the username/password from the `passwd` table corresponding to `key=ipmi`. To overwrite with a different username/password use the `-u` and `-p` option to `bmcdiscover`.

2. **Pre-define** the compute nodes:

Use the `bmcdiscover` command to help discover the nodes over an IP range and easily create a starting file to define the compute nodes into xCAT.

To discover the compute nodes for the BMCs with an IP address of 50.0.100.1, use the command:

```
bmcdiscover --range 50.0.100.1 -z > predefined.stanzas
```

The discovered nodes have the naming convention: `node-<model-type>-<serial-number>`

```
# cat predefined.stanzas
node-8247-221-10112ca:
  objtype=node
  groups=all
  bmc=50.0.100.1
```

(continues on next page)

(continued from previous page)

```
cons=ipmi
mgt=ipmi
mtm=8247-22L
serial=10112CA
```

3. Edit the `predefined.stanzas` file and change the discovered nodes to the intended hostname and IP address.

1. Edit the `predefined.stanzas` file:

```
vi predefined.stanzas
```

2. Rename the discovered object names to their intended compute node hostnames based on the MTMS mapping:

```
node-8247-221-10112ca ==> cn01
```

3. Add a `ip` attribute and give it the compute node IP address:

```
ip=10.0.101.1
```

4. Remove `nodetype` and `hwtype` if defined in the `predefined.stanza`.
5. Repeat for additional nodes in the `predefined.stanza` file based on the MTMS mapping.

In this example, the `predefined.stanzas` file now looks like the following:

```
# cat predefined.stanzas
cn01:
  objtype=node
  groups=all
  bmc=50.0.100.1
  cons=ipmi
  mgt=ipmi
  mtm=8247-22L
  serial=10112CA
  ip=10.0.101.1
```

4. Define the compute nodes into xCAT:

```
cat predefined.stanzas | mkdef -z
```

5. Set the chain table to run the `bmcsetup` script, this will set the BMC IP to static.

```
chdef cn01 chain="runcmd=bmcsetup"
```

6. **[Optional]** More operation plan to do after hardware discovery is done, `ondiscover` option can be used.

For example, configure console, copy SSH key for **OpenBMC**, then disable `powersupplyredundancy`

```
chdef cn01 -p chain=
  ↪ "ondiscover=makegocons|rspconfig:sshcfg|rspconfig:powersupplyredundancy=disabled"
```

**Note:** `|` is used to split commands, and `:` is used to split command with its option.

7. Set the target `osimage` into the chain table to automatically provision the operating system after the node discovery is complete.

```
chdef cn01 -p chain="osimage=<osimage_name>"
```

8. Change the BMC IP address

Set the BMC IP address to a different value for the **predefined** compute node definitions.

To change the dhcp obtained IP address of 50.0.100.1 to a static IP address of 50.0.101.1, run the following command:

```
chdef cn01 bmc=50.0.101.1
```

**[Optional]** If more configuration planned to be done on BMC, the following command is also needed.

```
chdef cn01 bmcvlangtag=<vlanid> # tag VLAN ID for BMC
chdef cn01 bmcusername=<desired_username>
chdef cn01 bmcpassword=<desired_password>
```

9. Add the compute node IP information to /etc/hosts:

```
makehosts cn01
```

10. Refresh the DNS configuration for the new hosts:

```
makedns -n
```

11. **[Optional]** Monitor the node discovery process using rcons

Configure the conserver for the **discovered** node to watch the discovery process using rcons:

```
makegocons node-8247-221-10112ca
```

In another terminal window, open the remote console:

```
rcons node-8247-221-10112ca
```

12. Start the discovery process by booting the **discovered** node definition:

```
rsetboot node-8247-221-10112ca net
rpower node-8247-221-10112ca on
```

13. The discovery process will network boot the machine into the diskless xCAT genesis kernel and perform the discovery process. When the discovery process is complete, doing `lsdef` on the compute nodes should show discovered attributes for the machine. The important mac information should be discovered, which is necessary for xCAT to perform OS provisioning.

## Set static BMC IP using dhcp provided IP address

The following example outlines the MTMS based hardware discovery for a single IPMI-based compute node.

Compute Node Information	Value
Model Type	8247-221
Serial Number	10112CA
Hostname	cn01
IP address	10.0.101.1



The BMC IP address is obtained by the open range dhcp server and the plan is to leave the IP address the same, except we want to change the IP address to be static in the BMC.

BMC Information	Value
IP address - dhcp	50.0.100.1
IP address - static	50.0.100.1

#### 1. Pre-define the compute nodes:

Use the `bmcdiscover` command to help discover the nodes over an IP range and easily create a starting file to define the compute nodes into xCAT.

To discover the compute nodes for the BMCs with an IP address of 50.0.100.1, use the command:

```
bmcdiscover --range 50.0.100.1 -z > predefined.stanzas
```

The discovered nodes have the naming convention: `node-<model-type>-<serial-number>`

```
# cat predefined.stanzas
node-8247-22l-10112ca:
  objtype=node
  groups=all
  bmc=50.0.100.1
  cons=ipmi
  mgt=ipmi
  mtm=8247-22L
  serial=10112CA
```

#### 2. Edit the `predefined.stanzas` file and change the discovered nodes to the intended hostname and IP address.

##### 1. Edit the `predefined.stanzas` file:

```
vi predefined.stanzas
```

##### 2. Rename the discovered object names to their intended compute node hostnames based on the MTMS mapping:

```
node-8247-22l-10112ca ==> cn01
```

##### 3. Add a `ip` attribute and give it the compute node IP address:

```
ip=10.0.101.1
```

##### 4. Repeat for additional nodes in the `predefined.stanza` file based on the MTMS mapping.

In this example, the `predefined.stanzas` file now looks like the following:

```
# cat predefined.stanzas
cn01:
  objtype=node
  groups=all
  bmc=50.0.100.1
  cons=ipmi
  mgt=ipmi
```

(continues on next page)

(continued from previous page)

```
mtm=8247-22L
serial=10112CA
ip=10.0.101.1
```

3. Define the compute nodes into xCAT:

```
cat predefined.stanzas | mkdef -z
```

4. Set the chain table to run the `bmcsetup` script, this will set the BMC IP to static.

```
chdef cn01 chain="runcmd=bmcsetup"
```

5. **[Optional]** More operation plan to do after hardware discovery is done, `ondiscover` option can be used.

For example, configure console, copy SSH key for **OpenBMC**, then disable `powersupplyredundancy`

```
chdef cn01 -p chain=
↪ "ondiscover=makegocons|rspconfig:sshcfg|rspconfig:powersupplyredundancy=disabled"
```

**Note:** `|` is used to split commands, and `:` is used to split command with its option.

6. Set the target `osimage` into the chain table to automatically provision the operating system after the node discovery is complete.

```
chdef cn01 -p chain="osimage=<osimage_name>"
```

7. Add the compute node IP information to `/etc/hosts`:

```
makehosts cn01
```

8. Refresh the DNS configuration for the new hosts:

```
makedns -n
```

9. **[Optional]** Monitor the node discovery process using `rcons`

Configure the `goconserver` for the **predefined** node to watch the discovery process using `rcons`:

```
makegocons cn01
```

In another terminal window, open the remote console:

```
rcons cn01
```

10. Start the discovery process by booting the **predefined** node definition:

```
rsetboot cn01 net
rpower cn01 on
```

11. The discovery process will network boot the machine into the diskless xCAT genesis kernel and perform the discovery process. When the discovery process is complete, doing `lsdef` on the compute nodes should show discovered attributes for the machine. The important `mac` information should be discovered, which is necessary for xCAT to perform OS provisioning.

## Switch-based Discovery

For switch based hardware discovery, the servers are identified through the switches and switchposts they are directly connected to.

In this document, the following configuration is used in the example

Management Node info:

```
MN Hostname: xcat1
MN NIC info for Management Network(Host network): eth1, 10.0.1.1/16
MN NIC info for Service Network(FSP/BMC network): eth2, 50.0.1.1/16
Dynamic IP range for Hosts: 10.0.100.1-10.0.100.100
Dynamic IP range for FSP/BMC: 50.0.100.1-50.0.100.100
```

Compute Node info:

```
CN Hostname: cn1
Machine type/model: 8247-22L
Serial: 10112CA
IP Address: 10.0.101.1
Root Password: cluster
Desired FSP/BMC IP Address: 50.0.101.1
DHCP assigned FSP/BMC IP Address: 50.0.100.1
FSP/BMC username: ADMIN
FSP/BMC Password: admin
```

Switch info:

```
Switch name: switch1
Switch username: xcat
Switch password: passw0rd
Switch IP Address: 10.0.201.1
Switch port for Compute Node: port0
```

## Configure xCAT

### Configure network table

Normally, there will be at least two entries for the two subnet on MN in `networks` table after xCAT is installed:

```
#tabdump networks
#netname,net,mask,mgtifname,gateway,dhcpserver,tftpserver,nameservers,ntpserver,
↳logserver,dynamicrange,staticrange,staticrangeincrement,nodehostname,ddnsdomain,
↳vlanid,domain,mtu,comments,disable
"10_0_0_0-255_255_0_0","10.0.0.0","255.255.0.0","eth1","<xcatmaster>",,"10.0.1.1",,,,,,
↳,,,,,
"50_0_0_0-255_255_0_0","50.0.0.0","255.255.0.0","eth2","<xcatmaster>",,"50.0.1.1",,,,,,
↳,,,,,
```

Run the following command to add networks in `networks` table if there are no entries in it:

```
makenetworks
```

## Setup DHCP

Set the correct NIC from which DHCP server provide service:

```
chdef -t site dhcpinterfaces=eth1,eth2
```

Add dynamic range in purpose of assigning temporary IP address for FSP/BMCs and hosts:

```
chdef -t network 10_0_0_0-255_255_0_0 dynamicrange="10.0.100.1-10.0.100.100"  
chdef -t network 50_0_0_0-255_255_0_0 dynamicrange="50.0.100.1-50.0.100.100"
```

Update DHCP configuration file:

```
makedhcp -n  
makedhcp -a
```

## Config passwd table

Set required passwords for xCAT to do hardware management and/or OS provisioning by adding entries to the xCAT passwd table:

```
# tabedit passwd  
# key,username,password,cryptmethod,authdomain,comments,disable
```

For hardware management with ipmi, add the following line:

```
"ipmi","ADMIN","admin",,,,
```

## Verify the genesis packages

The **xcat-genesis** packages should have been installed when xCAT was installed, but would cause problems if missing. **xcat-genesis** packages are required to create the genesis root image to do hardware discovery and the genesis kernel sits in `/tftpboot/xcat/`. Verify that the **genesis-scripts** and **genesis-base** packages are installed:

- [RHEL/SLES]: `rpm -qa | grep -i genesis`
- [Ubuntu]: `dpkg -l | grep -i genesis`

If missing, install them from the **xcat-deps** package and run `mknbn ppc64` to create the genesis network boot root image.

## Predefined Nodes

In order to differentiate one node from another, the admin needs to predefine node in xCAT database based on the switches information. This consists of two parts:

1. *Predefine Switches*
2. *Predefine Server Node*

### Predefine Switches

The predefined switches will represent devices that the physical servers are connected to. xCAT need to access those switches to get server related information through SNMP v3.

So the admin need to make sure those switches are configured correctly with SNMP v3 enabled. <TODO: The document that Configure Ethernet Switches>

Then, define switch info into xCAT:

```
nodeadd switch1 groups=switch,all
chdef switch1 ip=10.0.201.1
tabch switch=switch1 switches.snmpversion=3 switches.username=xcat switches.
↪password=passw0rd switches.auth=sha
```

Add switch into DNS using the following commands:

```
makehosts switch1
makedns -n
```

Predefine Server node

After switches are defined, the server node can be predefined with the following commands:

```
nodeadd cn1 groups=powerLE,all
chdef cn1 mgt=ipmi cons=ipmi ip=10.0.101.1 bmc=50.0.101.1 netboot=petitboot.
↪installnic=mac primarynic=mac
chdef cn1 switch=switch1 switchport=0
```

**[Optional]** If more configuration planed to be done on BMC, the following command is also needed.

```
chdef cn1 bmcvltag=<vlanid> # tag VLAN ID for BMC
chdef cn1 bmcusername=<desired_username>
chdef cn1 bmcpassw0rd=<desired_password>
```

In order to do BMC configuration during the discovery process, set `runcmd=bmcsetup`.

```
chdef cn1 chain="runcmd=bmcsetup"
```

**[Optional]** More operation plan to do after hardware discovery is done, `ondiscover` option can be used.

For example, configure console, copy SSH key for **OpenBMC**, then disable `powersupplyredundancy`

```
chdef cn01 -p chain=
↪"ondiscover=makegocons|rspconfig:sshcfg|rspconfig:powersupplyredundancy=disabled
↪"
```

**Note:** `|` is used to split commands, and `:` is used to split command with its option.

Set the target `osimage` into the chain table to automatically provision the operating system after the node discovery is complete.

```
chdef cn1 -p chain="osimage=<osimage_name>"
```

For more information about chain, refer to *Chain*

Add cn1 into DNS:

```
makehosts cn1
maekdns -n
```

## Discover server and define

After environment is ready, and the server is powered, we can start server discovery process. The first thing to do is discovering the FSP/BMC of the server. It is automatically powered on when the physical server is powered.

Use the `bmcdiscover` command to discover the BMCs responding over an IP range and write the output into the xCAT database. This discovered BMC node is used to control the physical server during hardware discovery and will be deleted after the correct server node object is matched to a pre-defined node. You **must** use the `-w` option to write the output into the xCAT database.

To discover the BMC with an IP address range of 50.0.100.1-100:

```
bmcdiscover --range 50.0.100.1-100 -z -w
```

The discovered nodes will be written to xCAT database. The discovered BMC nodes are in the form **node-model\_type-serial**. To view the discovered nodes:

```
lsdef /node-.*
```

**Note:** The `bmcdiscover` command will use the username/password from the `passwd` table corresponding to `key=ipmi`. To overwrite with a different username/password use the `-u` and `-p` option to `bmcdiscover`.

## Start discovery process

To start discovery process, just need to power on the PBMC node remotely with the following command, and the discovery process will start automatically after the host is powered on:

```
rpower node-8247-421-10112ca on
```

**[Optional]** If you'd like to monitor the discovery process, you can use:

```
makegocons node-8247-421-10112ca  
rcons node-8247-421-10112ca
```

## Verify node definition

The following is an example of the server node definition after hardware discovery:

```
#lsdef cn1  
Object name: cn1  
  arch=ppc64  
  bmc=50.0.101.1  
  cons=ipmi  
  cpucount=192  
  cputype=POWER8E (raw), altivec supported  
  groups=powerLE,all  
  installnic=mac  
  ip=10.0.101.1  
  mac=6c:ae:8b:02:12:50  
  memory=65118MB  
  mgt=ipmi  
  mtm=8247-22L
```

(continues on next page)

(continued from previous page)

```
netboot=petitboot
postbootscripts=otherpkgs
postscripts=syslog,remoteshell,syncfiles
primarynic=mac
serial=10112CA
supportedarchs=ppc64
switch=switch1
switchport=0
```

## Sequential-based Discovery

When the physical location of the server is not so important, sequential-based hardware discovery can be used to simplify the discovery work. The idea is: provided a node pool, each node in the pool will be assigned an IP address for host and an IP address for FSP/BMC, then the first physical server discovery request will be matched to the first free node in the node pool, and IP addresses for host and FSP/BMC will be assigned to that physical server.

In this document, the following configuration is used in the example

Management Node info:

```
MN Hostname: xcat1
MN NIC info for Management Network(Host network): eth1, 10.0.1.1/16
MN NIC info for Service Network(FSP/BMC network): eth2, 50.0.1.1/16
Dynamic IP range for Hosts: 10.0.100.1-10.0.100.100
Dynamic IP range for FSP/BMC: 50.0.100.1-50.0.100.100
```

Compute Node info:

```
CN Hostname: cn1
Machine type/model: 8247-22L
Serial: 10112CA
IP Address: 10.0.101.1
Root Password: cluster
Desired FSP/BMC IP Address: 50.0.101.1
DHCP assigned FSP/BMC IP Address: 50.0.100.1
FSP/BMC username: ADMIN
FSP/BMC Password: admin
```

## Configure xCAT

### Configure network table

Normally, there will be at least two entries for the two subnet on MN in `networks` table after xCAT is installed:

```
#tabdump networks
#netname,net,mask,mgtifname,gateway,dhcpserver,tftpserver,nameservers,ntpserver,
→logserver,dynamicrange,staticrange,staticrangeincrement,nodehostname,ddnsdomain,
→vlanid,domain,mtu,comments,disable
"10_0_0_0-255_255_0_0","10.0.0.0","255.255.0.0","eth1","<xcatmaster>","10.0.1.1",,,,,,
→,,,,,
```

(continues on next page)

(continued from previous page)

```
"50_0_0_0-255_255_0_0","50.0.0.0","255.255.0.0","eth2","<xcatmaster>",,"50.0.1.1",,,,,,  
→ , , , ,
```

Run the following command to add networks in `networks` table if there are no entries in it:

```
makenetworks
```

## Setup DHCP

Set the correct NIC from which DHCP server provide service:

```
chdef -t site dhcpinterfaces=eth1,eth2
```

Add dynamic range in purpose of assigning temporary IP address for FSP/BMCs and hosts:

```
chdef -t network 10_0_0_0-255_255_0_0 dynamicrange="10.0.100.1-10.0.100.100"  
chdef -t network 50_0_0_0-255_255_0_0 dynamicrange="50.0.100.1-50.0.100.100"
```

Update DHCP configuration file:

```
makedhcp -n  
makedhcp -a
```

## Config passwd table

Set required passwords for xCAT to do hardware management and/or OS provisioning by adding entries to the xCAT passwd table:

```
# tabedit passwd  
# key,username,password,cryptmethod,authdomain,comments,disable
```

For hardware management with ipmi, add the following line:

```
"ipmi","ADMIN","admin",,,,
```

## Verify the genesis packages

The **xcat-genesis** packages should have been installed when xCAT was installed, but would cause problems if missing. **xcat-genesis** packages are required to create the genesis root image to do hardware discovery and the genesis kernel sits in `/tftpboot/xcat/`. Verify that the **genesis-scripts** and **genesis-base** packages are installed:

- [RHEL/SLES]: `rpm -qa | grep -i genesis`
- [Ubuntu]: `dpkg -l | grep -i genesis`

If missing, install them from the **xcat-deps** package and run `mknb ppc64` to create the genesis network boot root image.



## Prepare node pool

To prepare the node pool, shall predefine nodes first, then initialize the discovery process with the predefined nodes.

### Predefine nodes

Predefine a group of nodes with desired IP address for host and IP address for FSP/BMC:

```
nodeadd cn1 groups=powerLE,all
chdef cn1 mgt=ipmi cons=ipmi ip=10.0.101.1 bmc=50.0.101.1 netboot=petitboot_
↪installnic=mac primarynic=mac
```

[Optional] If more configuration planed to be done on BMC, the following command is also needed.

```
chdef cn1 bmcvlangtag=<vlanid> # tag VLAN ID for BMC
chdef cn1 bmcusername=<desired_username>
chdef cn1 bmcpassword=<desired_password>
```

In order to do BMC configuration during the discovery process, set `runcmd=bmcsetup`.

```
chdef cn1 chain="runcmd=bmcsetup"
```

[Optional] More operation plan to do after hardware discovery is done, `ondiscover` option can be used.

For example, configure console, copy SSH key for **OpenBMC**, then disable `powersupplyredundancy`

```
chdef cn01 -p chain=
↪"ondiscover=makegocons|rspconfig:sshcfg|rspconfig:powersupplyredundancy=disabled
↪"
```

**Note:** `|` is used to split commands, and `:` is used to split command with its option.

Set the target `osimage` into the chain table to automatically provision the operating system after the node discovery is complete.

```
chdef cn1 -p chain="osimage=<osimage_name>"
```

For more information about chain, refer to [Chain](#)

### Initialize the discovery process

Specify the predefined nodes to the `nodediscoverstart` command to initialize the discovery process:

```
nodediscoverstart noderange=cn1
```

See `nodediscoverstart` for more information.

## Display information about the discovery process

There are additional *nodediscover\** commands you can run during the discovery process. See the man pages for more details.

Verify the status of discovery using *nodediscoverstatus*:

```
nodediscoverstatus
```

Show the nodes that have been discovered using *nodediscoverls*:

```
nodediscoverls -t seq -l
```

Stop the current sequential discovery process using: *nodediscoverstop*:

```
nodediscoverstop
```

**Note:** The sequential discovery process will stop automatically when all of the node names in the pool are consumed.

## Start discovery process

To start the discovery process, the system administrator needs to power on the servers one by one manually. Then the hardware discovery process will start automatically.

## Verify Node Definition

After discovery of the node, properties of the server will be added to the xCAT node definition.

Display the node definition and verify that the MAC address has been populated.

## Manually Define Nodes

**Manually Define Node** means the admin knows the detailed information of the physical server and manually defines it into xCAT database with *mkdef* commands.

In this document, the following configuration is used in the example

Management Node info:

```
MN Hostname: xcat1
MN NIC info for Management Network(Host network): eth1, 10.0.1.1/16
MN NIC info for Service Network(FSP/BMC network): eth2, 50.0.1.1/16
Dynamic IP range for Hosts: 10.0.100.1-10.0.100.100
Dynamic IP range for FSP/BMC: 50.0.100.1-50.0.100.100
```

Compute Node info:

```
CN Hostname: cn1
Machine type/model: 8247-22L
Serial: 10112CA
IP Address: 10.0.101.1
Root Password: cluster
Desired FSP/BMC IP Address: 50.0.101.1
```

(continues on next page)

(continued from previous page)

```
DHCP assigned FSP/BMC IP Address: 50.0.100.1
FSP/BMC username: ADMIN
FSP/BMC Password: admin
```

## Manually Define Node

Execute `mkdef` command to define the node:

```
mkdef -t node cn1 groups=powerLE,all mgt=ipmi cons=ipmi ip=10.0.101.1 netboot=petitboot
↪ bmc=50.0.101.1 bmcusername=ADMIN bmcpassword=admin installnic=mac primarynic=mac
↪ mac=6c:ae:8b:6a:d4:e4
```

The manually defined node will be like this:

```
Object name: cn1
  bmc=50.0.101.1
  bmcpassword=admin
  bmcusername=ADMIN
  cons=ipmi
  groups=powerLE,all
  installnic=mac
  ip=10.0.101.1
  mac=6c:ae:8b:6a:d4:e4
  mgt=ipmi
  netboot=petitboot
  postbootscripts=otherpkgs
  postscripts=syslog,remoteshell,syncfiles
  primarynic=mac
```

`mkdef --template` can be used to create node definitions easily from the typical node definition templates or existing node definitions, some examples:

- creating node definition “cn2” from an existing node definition “cn1”

```
mkdef -t node -o cn2 --template cn1 mac=66:55:44:33:22:11 ip=172.12.139.2 bmc=172.
↪ 11.139.2
```

except for the attributes specified (mac, ip and bmc), other attributes of the newly created node “cn2” inherit the values of template node “cn1”

- creating a node definition “cn2” with the template “ppc64le-openbmc-template” (openbmc controlled ppc64le node) shipped by xCAT

```
mkdef -t node -o cn2 --template ppc64le-openbmc-template mac=66:55:44:33:22:11
↪ ip=172.12.139.2 bmc=172.11.139.2 bmcusername=root bmcpassword=openBmc
```

the unspecified attributes of newly created node “cn2” will be assigned with the default values in the template to list all the node definition templates available in xCAT, run

```
lsdef -t node --template
```

to display the full definition of template “ppc64le-openbmc-template”, run

```
lsdef -t node --template ppc64le-openbmc-template
```



the mandatory attributes, which must be specified while creating definitions with templates, are denoted with the value `MANDATORY:<attribute description>` in template definition.

the optional attributes, which can be specified optionally, are denoted with the value `OPTIONAL:<attribute description>` in template definition

## Manually Discover Nodes

If you have a few nodes which were not discovered by automated hardware discovery process, you can find them in `discoverydata` table using the `nodediscoverls` command. The undiscovered nodes are those that have a discovery method value of 'undef' in the `discoverydata` table.

Display the undefined nodes with the `nodediscoverls` command:

```
#nodediscoverls -t undef
UUID                                NODE                                METHOD                                MTM                                
↳ SERIAL
fa2cec8a-b724-4840-82c7-3313811788cd  undef                                undef                                8247-22L 
↳ 10112CA
```

If you want to manually define an 'undefined' node to a specific free node name, use the `nodediscoverdef(TODO)` command.



Before doing that, a node with desired IP address for host and FSP/BMC must be defined first:

```
nodeadd cn1 groups=powerLE,all
chdef cn1 mgt=ipmi cons=ipmi ip=10.0.101.1 bmc=50.0.101.1 netboot=petitboot
↳ installnic=mac primarynic=mac
```

For example, if you want to assign the undefined node whose uuid is `fa2cec8a-b724-4840-82c7-3313811788cd` to `cn1`, run:

```
nodediscoverdef -u fa2cec8a-b724-4840-82c7-3313811788cd -n cn1
```

After manually defining it, the 'node name' and 'discovery method' attributes of the node will be changed. You can display the changed attributes using the `nodediscoverls` command:

```
#nodediscoverls
UUID                                NODE                                METHOD                                MTM                                
↳ SERIAL
fa2cec8a-b724-4840-82c7-3313811788cd  cn1                                manual                                8247-22L 
↳ 10112CA
```

**Note:** You can optionally add drivers or modules to the network boot image used during the discovery: *xCAT Genesis Base*

Following are the brief characteristics and adaptability of each method, you can select a proper one according to your cluster size and other consideration.

- **Manually Define Nodes**

Manually collect information for target servers and manually define them to xCAT **Node Object** through `mkdef` command.

This method is recommended for small cluster which has less than 10 nodes.

- pros

No specific configuration and procedure required and very easy to use.

- cons

It will take additional time to configure the SP (Management Modules like: BMC, FSP) and collect the server information like MTMS (Machine Type and Machine Serial) and Host MAC address for OS deployment ...

This method is inefficient and error-prone for a large number of servers.

- **MTMS-based Discovery**

**Step1: Automatically** search all the servers and collect server MTMS information.

**Step2:** Define the searched server to a **Node Object** automatically. In this case, the node name will be generated based on the **MTMS** string. The admin can rename the **Node Object** to a reasonable name like **r1u1** (It means the physical location is in Rack1 and Unit1).

**Step3:** Power on the nodes, xCAT discovery engine will update additional information like the **MAC for deployment** for the nodes.

This method is recommended for the medium scale of cluster which has less than 100 nodes.

- pros

With limited effort to get the automatic discovery benefit.

- cons

Compared to **Switch-based Discovery**, the admin needs to be involved to rename the automatically discovered node to a reasonable name (optional). It's hard to rename the node to a location-based name for a large number of server.

- **Switch-based Discovery**

**Step1: Pre-define** the **Node Object** for all the nodes in the cluster. The **Pre-defined** node must have the attributes **switch** and **switchport** defined to specify which **Switch and Port** this server connected to. xCAT will use this **Switch and Port** information to map a discovered node to certain **Pre-defined** node.

**Step2:** Power on the nodes, xCAT discovery engine will discover node attributes and update them to certain **Pre-defined** node.

- pros

The whole discovery process is totally automatic.

Since the node is physically identified by the **Switch and Port** that the server connected, if a node fail and replaced with a new one, xCAT will automatically discover the new one and assign it to the original node name since the **Switch and Port** does not change.

- cons

You need to plan the cluster with planned **Switch and Port** mapping for each server and switch. All the Switches need be configured with `snmpv3` accessible for xCAT management node.

- **Sequential-based Discovery**

**Step1: Pre-define** the **Node Object** for all the nodes in the cluster.

**Step2:** Manually power on the node one by one. The booted node will be discovered, each new discovered node will be assigned to one of the **Pre-defined** node in **Sequential**.

- pros

No special configuration required like **Switch-based Discovery**. No manual rename node step required like **MTMS-based Discovery**.

- cons

You have to strictly boot on the node in order if you want the node has the expected name. Generally you have to waiting for the discovery process finished before power on the next one.

## Hardware Management

### Basic Operations

#### rbeacon - Beacon Light

See *rbeacon manpage* for more information.

Most enterprise level servers have LEDs on their front and/or rear panels, one of which is a beacon light. If turned on, this light can assist the system administrator in locating one physical machine in the cluster.

Using xCAT, administrators can turn on and off the beacon light using: `rbeacon <node> on|off`

#### rpower - Remote Power Control

See *rpower manpage* for more information.

Use the `rpower` command to remotely power on and off a single server or a range of servers.

```
rpower <noderange> on
rpower <noderange> off
```

Other actions include:

- To get the current power state of a server: `rpower <noderange> state`
- To boot/reboot a server: `rpower <noderange> boot`
- To hardware reset a server: `rpower <noderange> reset`

#### rcons - Remote Console

See *rcons manpage* for more information.

Most enterprise servers do not have video adapters installed with the machine and often do not provide a method for attaching a physical monitor/keyboard/mouse to get the display output. For this purpose xCAT can assist the system administrator to view the console over a “Serial-over-LAN” (SOL) connection through the BMC.

Configure the correct console management by modifying the node definition:

- For OpenPOWER, **IPMI** managed server:

```
chdef -t node -o <noderange> cons=ipmi
```

- For OpenPOWER, **OpenBMC** managed servers:

```
chdef -t node -o <noderange> cons=openbmc
```

Open a console to compute1:

```
rcons compute1
```

**Note:** The keystroke `ctrl+e c .` will disconnect you from the console.

## Troubleshooting

### General

xCAT has been integrated with 3 kinds of console server service, they are

- [conserver](#) **[Deprecated]**
- [goconserver](#)
- [confluent](#)

`rcons` command relies on one of them. The `conserver` and `goconserver` packages should have been installed with xCAT as they are part of the xCAT dependency packages. If you want to try `confluent`, see [confluent server](#).

For systemd based systems, `goconserver` is used by default. If you are having problems seeing the console, try the following.

1. Make sure `goconserver` is configured by running `makegocons`.
2. Check if `goconserver` is up and running  
`systemctl status goconserver.service`
3. If `goconserver` is not running, start the service using:  
`systemctl start goconserver.service`
4. Try `makegocons -q [<node>]` to verify if the node has been registered.
5. Invoke the console again: `rcons <node>`

More details for `goconserver`, see [goconserver documentation](#).

**[Deprecated]** If `conserver` is used, try the following.

1. Make sure `conserver` is configured by running `makeconservercf`.
2. Check if `conserver` is up and running

```
[sysvinit] service consver status
[systemd] systemctl status consver.service
```

3. If `conserver` is not running, start the service using:

```
[sysvinit] service consver start
[systemd] systemctl start consver.service
```

4. Invoke the console again: `rcons <node>`

## Advanced Operations

### **rinv - Remote Hardware Inventory**

See *rinv manpage* for more information.

Use **rinv** command to remotely obtain inventory information of a physical machine. This will help to distinguish one machine from another and aid in mapping the model type and/or serial number of a machine with its host name.

To get all the hardware information for node **cn1**:

```
rinv cn1 all
```

To get just the firmware information for **cn1**:

```
rinv cn1 firm
```

### **rvitals - Remote Hardware Vitals**

See *rvitals manpage* for more information.

Collecting runtime information from a running physical machine is an important part of system administration. Data can be obtained from the service processor including temperature, voltage, cooling fans, etc.

Use the **rvitals** command to obtain this information.

```
rvitals <noderange> all
```

To only get the temperature information of machines in a particular noderange:

```
rvitals <noderange> temp
```

### **rflash - Remote Firmware Flashing**

See *rflash manpage* for more information.

### **IPMI Firmware Update**

The **rflash** command is provided to assist the system administrator in updating firmware.

To check the current firmware version on the node's BMC and the HPM file:

```
rflash <noderange> -c /firmware/8335_810.1543.20151021b_update.hpm
```

To update the firmware on the node's BMC to version in the HPM file:

```
rflash <noderange> /firmware/8335_810.1543.20151021b_update.hpm
```



## OpenBMC Firmware Update

### Manual Firmware Flash

The sequence of events that must happen to flash OpenBMC firmware is the following:

1. Power off the Host
2. Upload and Activate BMC
3. Reboot the BMC (applies BMC)
4. Upload and Activate Host
5. Power on the Host (applies Host)

### Power off Host

Use the `rpower` command to power off the host:

```
rpower <noderange> off
```

### Upload and Activate BMC Firmware

Use the `rflash` command to upload and activate the Host firmware:

```
rflash <noderange> -a /path/to/obmc-phosphor-image-witherspoon.ubi.mtd.tar
```

If running `rflash` in Hierarchy, the firmware files must be accessible on the Service Nodes.

**Note:** If a `.tar` file is provided, the `-a` option does an upload and activate in one step. If an ID is provided, the `-a` option just does activate the specified firmware. After firmware is activated, use the `rflash <noderange> -l` to view. The `rflash` command shows (\*) as the active firmware and (+) on the firmware that requires reboot to become effective.

### Reboot the BMC

Use the `rpower` command to reboot the BMC:

```
rpower <noderange> bmcreboot
```

The BMC will take 2-5 minutes to reboot, check the status using: `rpower <noderange> bmcstate` and wait for BMCReady to be returned.

**Known Issue:** On reboot, the first call to the BMC after reboot, xCAT will return Error: BMC did not respond within 10 seconds, retry the command.. Please retry.

## Upload and Activate Host Firmware

Use the `rflash` command to upload and activate the Host firmware:

```
rflash <noderange> -a /path/to/witherspoon.pnor.squashfs.tar
```

If running `rflash` in Hierarchy, the firmware files must be accessible on the Service Nodes.

**Note:** The `-a` option does an upload and activate in one step, after firmware is activated, use the `rflash <noderange> -l` to view. The `rflash` command shows (\*) as the active firmware and (+) on the firmware that requires reboot to become effective.

## Power on Host

Use the `rpower` command to power on the Host:

```
rpower <noderange> on
```

## Validation

Use one of the following commands to validate firmware levels are in sync:

- Use the `rinv` command to validate firmware level:

```
rinv <noderange> firm -V | grep -i ibm | grep "\*" | xcoll
```

- Use the `rflash` command to validate the firmware level:

```
rflash <noderange> -l | grep "\*" | xcoll
```

## Unattended Firmware Flash

Unattended flash of OpenBMC firmware will do the following events:

1. Upload both BMC firmware file and Host firmware file
2. Activate both BMC firmware and Host firmware
3. If BMC firmware becomes activate, reboot BMC to apply new BMC firmware, or else, `rflash` will exit
4. If BMC itself state is `NotReady`, `rflash` will exit
5. If BMC itself state is `Ready`, `rflash` will reboot the compute node to apply Host firmware

Use the following command to flash the firmware unattended:

```
rflash <noderange> -d /path/to/directory
```

If there are errors encountered during the flash process, take a look at the manual steps to continue flashing the BMC.

## Validation

Use one of the following commands to validate firmware levels are in sync:

- Use the `rinv` command to validate firmware level:

```
rinv <noderange> firm -V | grep -i ibm | grep "\*" | xcoll
```

- Use the `rflash` command to validate the firmware level:

```
rflash <noderange> -l | grep "\*" | xcoll
```

## rspconfig - Remote Configuration of Service Processors

See *rspconfig manpage* for more information.

The `rspconfig` command can be used to configure the service processor, or Baseboard Management Controller (BMC), of a physical machine.

For example, to turn on SNMP alerts for node `cn5`:

```
rspconfig cn5 alert=on
```

## reventlog - Remote Event Log of Service Processors

See *reventlog manpage* for more information.

The `reventlog` command can be used to display and clear event log information on the service processor, or Baseboard Management Controller (BMC), of a physical machine. OpenBMC based servers need the [IBM OpenBMC tool](#) to obtain more detailed logging messages.

For example, to display all event log entries for node `cn5`:

```
reventlog cn5
```

To clear all event log entries for node `cn5`:

```
reventlog cn5 clear
```

## Diskful Installation

### Select or Create an osimage Definition

Before creating an image on xCAT, the distro media should be prepared. That can be ISOs or DVDs.

XCAT uses `copycds` command to create an image which will be available to install nodes. `copycds` will copy all contents of Distribution DVDs/ISOs or Service Pack DVDs/ISOs to a destination directory, and create several relevant osimage definitions by default.

If using an ISO, copy it to (or NFS mount it on) the management node, and then run:

```
copycds <path>/<specific-distro>.iso
```

**Note:** While sle15 contains installer medium and packages medium, need copycds copy all contents of DVD1 of the installer medium and DVD1 of the packages medium, for example:

```
copycds SLE-15-Installer-DVD-ppc64le-GM-DVD1.iso SLE-15-Packages-ppc64le-GM-DVD1.iso
```

---

If using a DVD, put it in the DVD drive of the management node and run:

```
copycds /dev/<dvd-drive-name>
```

To see the list of osimages:

```
lsdef -t osimage
```

To see the attributes of a particular osimage:

```
lsdef -t osimage <osimage-name>
```

Initially, some attributes of osimage are assigned default values by xCAT - they all can work correctly because the files or templates invoked by those attributes are shipped with xCAT by default. If you need to customize those attributes, refer to the next section *Customize osimage*

Below is an example of osimage definitions created by copycds:

```
# lsdef -t osimage
rhels7.2-ppc64le-install-compute (osimage)
rhels7.2-ppc64le-install-service (osimage)
rhels7.2-ppc64le-netboot-compute (osimage)
rhels7.2-ppc64le-stateful-mgmtnode (osimage)
```

In these osimage definitions shown above

- **<os>-<arch>-install-compute** is the default osimage definition used for diskful installation
- **<os>-<arch>-netboot-compute** is the default osimage definition used for diskless installation
- **<os>-<arch>-install-service** is the default osimage definition used for service node deployment which shall be used in hierarchical environment

---

**Note:** Additional steps are needed for **ubuntu ppc64le** osimages:

---

For pre-16.04.02 version of Ubuntu for ppc64el, the `initrd.gz` shipped with the ISO does not support network booting. In order to install Ubuntu with xCAT, you need to follow the steps to complete the osimage definition.

- Download `mini.iso` from
  - [ubuntu 14.04.1]: <http://xcat.org/files/netboot/ubuntu14.04.1/ppc64el/mini.iso>
  - [ubuntu 14.04.2]: <http://xcat.org/files/netboot/ubuntu14.04.2/ppc64el/mini.iso>
  - [ubuntu 14.04.3]: <http://xcat.org/files/netboot/ubuntu14.04.3/ppc64el/mini.iso>
  - [ubuntu 14.04.4]: <http://xcat.org/files/netboot/ubuntu14.04.4/ppc64el/mini.iso>
  - [ubuntu 16.04]: <http://xcat.org/files/netboot/ubuntu16.04/ppc64el/mini.iso>
  - [ubuntu 16.04.1]: <http://xcat.org/files/netboot/ubuntu16.04.1/ppc64el/mini.iso>
- Mount `mini.iso`

```
mkdir /tmp/iso
mount -o loop mini.iso /tmp/iso
```

- Copy the netboot initrd.gz to osimage

```
mkdir -p /install/<ubuntu-version>/ppc64el/install/netboot
cp /tmp/iso/install/initrd.gz /install/<ubuntu-version>/ppc64el/install/netboot
```

### [Tips 1]

If this is the same distro version as what your management node uses, create a `.repo` file in `/etc/yum.repos.d` with contents similar to:

```
[local-<os>-<arch>]
name=xCAT local <os> <version>
baseurl=file:/install/<os>/<arch>
enabled=1
gpgcheck=0
```

This way, if you need to install some additional RPMs into your MN later, you can simply install them with `yum`. Or if you are installing a software on your MN that depends some RPMs from this distro, those RPMs will be found and installed automatically.

### [Tips 2]

You can create/modify an osimage definition easily with any existing osimage definition, the command is

```
mkdef -t osimage -o <new osimage> --template <existing osimage> [<attribute>=<value>, ...
↪]
```

Except the specified attributes `<attribute>`, the attributes of `<new osimage>` will inherit the values of template osimage `<existing osimage>`.

As an example, the following command creates a new osimage `myosimage.rh7.compute.netboot` based on the existing osimage `rhels7.4-ppc64le-netboot-compute` with some customized attributes

```
mkdef -t osimage -o myosimage.rh7.compute.netboot --template rhels7.4-ppc64le-netboot-
↪compute synclists=/tmp/synclist otherpkgdir=/install/custom/osimage/myosimage.rh7.
↪compute.netboot/3rdpkgs/ otherpkglist=/install/custom/osimage/myosimage.rh7.compute.
↪netboot/3rd.pkglist
```

## Customize osimage (Optional)

Optional means all the subitems in this page are not necessary to finish an OS deployment. If you are new to xCAT, you can just jump to *Initialize the Compute for Deployment*.

## Configure RAID before deploying the OS

### Overview

xCAT provides an user interface *linuximage.partitionfile* to specify the customized partition script for diskful provision, and provides some default partition scripts.

### Deploy Diskful Nodes with RAID1 Setup on RedHat

xCAT provides a partition script *raid1\_rh.sh* which configures RAID1 across 2 disks on RHEL 7.x operating systems.

In most scenarios, the sample partitioning script is sufficient to create a basic RAID1 across two disks and is provided as a sample to build upon.

1. Obtain the partition script:

```
mkdir -p /install/custom/partition/
wget https://raw.githubusercontent.com/xcat2/xcat-extensions/master/partition/raid1_
↪rh.sh \
    -O /install/custom/partition/raid1_rh.sh
```

2. Associate the partition script to the osimage:

```
chdef -t osimage -o rhels7.3-ppc64le-install-compute \
    partitionfile="s:/install/custom/partition/raid1_rh.sh"
```

3. Provision the node:

```
rinstall cn1 osimage=rhels7.3-ppc64le-install-compute
```

After the diskful nodes are up and running, you can check the RAID1 settings with the following process:

`mount` command shows the `/dev/mdx` devices are mounted to various file systems, the `/dev/mdx` indicates that the RAID is being used on this node.

```
# mount
...
/dev/md1 on / type xfs (rw,relatime,attr2,inode64,noquota)
/dev/md0 on /boot type xfs (rw,relatime,attr2,inode64,noquota)
/dev/md2 on /var type xfs (rw,relatime,attr2,inode64,noquota)
```

The file `/proc/mdstat` includes the RAID devices status on the system, here is an example of `/proc/mdstat` in the non-multipath environment:

```
# cat /proc/mdstat
Personalities : [raid1]
md2 : active raid1 sdk2[0] sdj2[1]
      1047552 blocks super 1.2 [2/2] [UU]
      resync=DELAYED
      bitmap: 1/1 pages [64KB], 65536KB chunk

md3 : active raid1 sdk3[0] sdj3[1]
      1047552 blocks super 1.2 [2/2] [UU]
      resync=DELAYED
```

(continues on next page)

(continued from previous page)

```
md0 : active raid1 sdk5[0] sdj5[1]
      524224 blocks super 1.0 [2/2] [UU]
      bitmap: 0/1 pages [0KB], 65536KB chunk

md1 : active raid1 sdk6[0] sdj6[1]
      973998080 blocks super 1.2 [2/2] [UU]
      [==>.....] resync = 12.8% (125356224/973998080) finish=138.1min
      ↪speed=102389K/sec
      bitmap: 1/1 pages [64KB], 65536KB chunk

unused devices: <none>
```

On the system with multipath configuration, the `/proc/mdstat` looks like:

```
# cat /proc/mdstat
Personalities : [raid1]
md2 : active raid1 dm-11[0] dm-6[1]
      291703676 blocks super 1.1 [2/2] [UU]
      bitmap: 1/1 pages [64KB], 65536KB chunk

md1 : active raid1 dm-8[0] dm-3[1]
      1048568 blocks super 1.1 [2/2] [UU]

md0 : active raid1 dm-9[0] dm-4[1]
      204788 blocks super 1.0 [2/2] [UU]

unused devices: <none>
```

The command `mdadm` can query the detailed configuration for the RAID partitions:

```
mdadm --detail /dev/md2
```

## Deploy Diskful Nodes with RAID1 Setup on SLES

xCAT provides one sample autoyast template files with the RAID1 settings `/opt/xcat/share/xcat/install/sles/service.raid1.sles11.tmpl`. You can customize the template file and put it under `/install/custom/install/<platform>/` if the default one does not match your requirements.

Here is the RAID1 partitioning section in `service.raid1.sles11.tmpl`:

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <partitions config:type="list">
      <partition>
        <format config:type="boolean">>false</format>
        <partition_id config:type="integer">65</partition_id>
        <partition_nr config:type="integer">1</partition_nr>
        <partition_type>primary</partition_type>
        <size>24M</size>
      </partition>
```

(continues on next page)

(continued from previous page)

```

<partition>
  <format config:type="boolean">false</format>
  <partition_id config:type="integer">253</partition_id>
  <partition_nr config:type="integer">2</partition_nr>
  <raid_name>/dev/md0</raid_name>
  <raid_type>raid</raid_type>
  <size>2G</size>
</partition>
<partition>
  <format config:type="boolean">false</format>
  <partition_id config:type="integer">253</partition_id>
  <partition_nr config:type="integer">3</partition_nr>
  <raid_name>/dev/md1</raid_name>
  <raid_type>raid</raid_type>
  <size>max</size>
</partition>
</partitions>
<use>all</use>
</drive>
<drive>
  <device>/dev/sdb</device>
  <partitions config:type="list">
    <partition>
      <format config:type="boolean">false</format>
      <partition_id config:type="integer">131</partition_id>
      <partition_nr config:type="integer">1</partition_nr>
      <partition_type>primary</partition_type>
      <size>24M</size>
    </partition>
    <partition>
      <format config:type="boolean">false</format>
      <partition_id config:type="integer">253</partition_id>
      <partition_nr config:type="integer">2</partition_nr>
      <raid_name>/dev/md0</raid_name>
      <raid_type>raid</raid_type>
      <size>2G</size>
    </partition>
    <partition>
      <format config:type="boolean">false</format>
      <partition_id config:type="integer">253</partition_id>
      <partition_nr config:type="integer">3</partition_nr>
      <raid_name>/dev/md1</raid_name>
      <raid_type>raid</raid_type>
      <size>max</size>
    </partition>
  </partitions>
  <use>all</use>
</drive>
<drive>
  <device>/dev/md</device>
  <partitions config:type="list">
    <partition>

```

(continues on next page)



(continued from previous page)

```
<filesystem config:type="symbol">reiser</filesystem>
<format config:type="boolean">true</format>
<mount>swap</mount>
<partition_id config:type="integer">131</partition_id>
<partition_nr config:type="integer">0</partition_nr>
<raid_options>
  <chunk_size>4</chunk_size>
  <parity_algorithm>left-asymmetric</parity_algorithm>
  <raid_type>raid1</raid_type>
</raid_options>
</partition>
<partition>
  <filesystem config:type="symbol">reiser</filesystem>
  <format config:type="boolean">true</format>
  <mount>/</mount>
  <partition_id config:type="integer">131</partition_id>
  <partition_nr config:type="integer">1</partition_nr>
  <raid_options>
    <chunk_size>4</chunk_size>
    <parity_algorithm>left-asymmetric</parity_algorithm>
    <raid_type>raid1</raid_type>
  </raid_options>
</partition>
</partitions>
<use>all</use>
</drive>
</partitioning>
```

The samples above created one 24MB PRéP partition on each disk, one 2GB mirrored swap partition and one mirrored / partition uses all the disk space. If you want to use different partitioning scheme in your cluster, modify this RAID1 section in the autoyast template file accordingly.

Since the PRéP partition can not be mirrored between the two disks, some additional postinstall commands should be run to make the second disk bootable, here the commands needed to make the second disk bootable:

```
# Set the second disk to be bootable for RAID1 setup
parted -s /dev/sdb mkfs 1 fat32
parted /dev/sdb set 1 type 6
parted /dev/sdb set 1 boot on
dd if=/dev/sda1 of=/dev/sdb1
bootlist -m normal sda sdb
```

The procedure listed above has been added to the file /opt/xcats/share/xcats/install/scripts/post.sles11.raid1 to make it be automated. The autoyast template file service.raid1.sles11.tmpl will include the content of post.sles11.raid1, so no manual steps are needed here.

After the diskful nodes are up and running, you can check the RAID1 settings with the following commands:

Mount command shows the /dev/mdx devices are mounted to various file systems, the /dev/mdx indicates that the RAID is being used on this node.

```
server:~ # mount
/dev/md1 on / type reiserfs (rw)
proc on /proc type proc (rw)
```

(continues on next page)

(continued from previous page)

```
sysfs on /sys type sysfs (rw)
debugfs on /sys/kernel/debug type debugfs (rw)
devtmpfs on /dev type devtmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,mode=1777)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
```

The file `/proc/mdstat` includes the RAID devices status on the system, here is an example of `/proc/mdstat`:

```
server:~ # cat /proc/mdstat
Personalities : [raid1] [raid0] [raid10] [raid6] [raid5] [raid4]
md0 : active (auto-read-only) raid1 sda2[0] sdb2[1]
      2104500 blocks super 1.0 [2/2] [UU]
      bitmap: 0/1 pages [0KB], 128KB chunk

md1 : active raid1 sda3[0] sdb3[1]
      18828108 blocks super 1.0 [2/2] [UU]
      bitmap: 0/9 pages [0KB], 64KB chunk

unused devices: <none>
```

The command `mdadm` can query the detailed configuration for the RAID partitions:

```
mdadm --detail /dev/md1
```

## Disk Replacement Procedure

If any one disk fails in the RAID1 array, do not panic. Follow the procedure listed below to replace the failed disk.

Faulty disks should appear marked with an (F) if you look at `/proc/mdstat`:

```
# cat /proc/mdstat
Personalities : [raid1]
md2 : active raid1 dm-11[0](F) dm-6[1]
      291703676 blocks super 1.1 [2/1] [_U]
      bitmap: 1/1 pages [64KB], 65536KB chunk

md1 : active raid1 dm-8[0](F) dm-3[1]
      1048568 blocks super 1.1 [2/1] [_U]

md0 : active raid1 dm-9[0](F) dm-4[1]
      204788 blocks super 1.0 [2/1] [_U]

unused devices: <none>
```

We can see that the first disk is broken because all the RAID partitions on this disk are marked as (F).

## Remove the failed disk from RAID array

mdadm is the command that can be used to query and manage the RAID arrays on Linux. To remove the failed disk from RAID array, use the command:

```
mdadm --manage /dev/mdx --remove /dev/xxx
```

Where the /dev/mdx are the RAID partitions listed in /proc/mdstat file, such as md0, md1 and md2; the /dev/xxx are the backend devices like dm-11, dm-8 and dm-9 in the multipath configuration and sda5, sda3 and sda2 in the non-multipath configuration.

Here is the example of removing failed disk from the RAID1 array in the non-multipath configuration:

```
mdadm --manage /dev/md0 --remove /dev/sda3
mdadm --manage /dev/md1 --remove /dev/sda2
mdadm --manage /dev/md2 --remove /dev/sda5
```

Here is the example of removing failed disk from the RAID1 array in the multipath configuration:

```
mdadm --manage /dev/md0 --remove /dev/dm-9
mdadm --manage /dev/md1 --remove /dev/dm-8
mdadm --manage /dev/md2 --remove /dev/dm-11
```

After the failed disk is removed from the RAID1 array, the partitions on the failed disk will be removed from /proc/mdstat and the mdadm --detail output also.

```
# cat /proc/mdstat
Personalities : [raid1]
md2 : active raid1 dm-6[1]
      291703676 blocks super 1.1 [2/1] [_U]
      bitmap: 1/1 pages [64KB], 65536KB chunk

md1 : active raid1 dm-3[1]
      1048568 blocks super 1.1 [2/1] [_U]

md0 : active raid1 dm-4[1]
      204788 blocks super 1.0 [2/1] [_U]

unused devices: <none>

# mdadm --detail /dev/md0
/dev/md0:
   Version : 1.0
  Creation Time : Tue Jul 19 02:39:03 2011
    Raid Level : raid1
    Array Size : 204788 (200.02 MiB 209.70 MB)
  Used Dev Size : 204788 (200.02 MiB 209.70 MB)
    Raid Devices : 2
   Total Devices : 1
 Persistence : Superblock is persistent

   Update Time : Wed Jul 20 02:00:04 2011
     State : clean, degraded
  Active Devices : 1
```

(continues on next page)

(continued from previous page)

```
Working Devices : 1
Failed Devices : 0
Spare Devices : 0

    Name : c250f17c01ap01:0 (local to host c250f17c01ap01)
    UUID : eba4d8ad:8f08f231:3c60e20f:1f929144
    Events : 26

    Number   Major   Minor   RaidDevice State
    0         0       0       0         removed
    1        253       4       1         active sync  /dev/dm-4
```

## Replace the disk

Depends on the hot swap capability, you may simply unplug the disk and replace with a new one if the hot swap is supported; otherwise, you will need to power off the machine and replace the disk and the power on the machine. Create partitions on the new disk

The first thing we must do now is to create the exact same partitioning as on the new disk. We can do this with one simple command:

```
sfdisk -d /dev/<good_disk> | sfdisk /dev/<new_disk>
```

For the non-multipath configuration, here is an example:

```
sfdisk -d /dev/sdb | sfdisk /dev/sda
```

For the multipath configuration, here is an example:

```
sfdisk -d /dev/dm-1 | sfdisk /dev/dm-0
```

If you got error message “sfdisk: I don’t like these partitions - nothing changed.”, you can add `--force` option to the sfdisk command:

```
sfdisk -d /dev/sdb | sfdisk /dev/sda --force
```

You can run:

```
fdisk -l
```

To check if both hard drives have the same partitioning now.

## Add the new disk into the RAID1 array

After the partitions are created on the new disk, you can use command:

```
mdadm --manage /dev/mdx --add /dev/xxx
```

To add the new disk to the RAID1 array. Where the `/dev/mdx` are the RAID partitions like md0, md1 and md2; the `/dev/xxx` are the backend devices like dm-11, dm-8 and dm-9 in the multipath configuration and sda5, sda3 and sda2 in the non-multipath configuration.

Here is an example for the non-multipath configuration:

```
mdadm --manage /dev/md0 --add /dev/sda3
mdadm --manage /dev/md1 --add /dev/sda2
mdadm --manage /dev/md2 --add /dev/sda5
```

Here is an example for the multipath configuration:

```
mdadm --manage /dev/md0 --add /dev/dm-9
mdadm --manage /dev/md1 --add /dev/dm-8
mdadm --manage /dev/md2 --add /dev/dm-11
```

All done! You can have a cup of coffee to watch the fully automatic reconstruction running...

While the RAID1 array is reconstructing, you will see some progress information in /proc/mdstat:

```
# cat /proc/mdstat
Personalities : [raid1]
md2 : active raid1 dm-11[0] dm-6[1]
      291703676 blocks super 1.1 [2/1] [_U]
      [=.....] recovery = 0.7% (2103744/291703676) finish=86.2min
↳ speed=55960K/sec
      bitmap: 1/1 pages [64KB], 65536KB chunk

md1 : active raid1 dm-8[0] dm-3[1]
      1048568 blocks super 1.1 [2/1] [_U]
      [=====>.....] recovery = 65.1% (683904/1048568) finish=0.1min
↳ speed=48850K/sec

md0 : active raid1 dm-9[0] dm-4[1]
      204788 blocks super 1.0 [2/1] [_U]
      [=====>.] recovery = 96.5% (198016/204788) finish=0.0min
↳ speed=14144K/sec

unused devices: <none>
```

After the reconstruction is done, the /proc/mdstat becomes like:

```
# cat /proc/mdstat
Personalities : [raid1]
md2 : active raid1 dm-11[0] dm-6[1]
      291703676 blocks super 1.1 [2/2] [UU]
      bitmap: 1/1 pages [64KB], 65536KB chunk

md1 : active raid1 dm-8[0] dm-3[1]
      1048568 blocks super 1.1 [2/2] [UU]

md0 : active raid1 dm-9[0] dm-4[1]
      204788 blocks super 1.0 [2/2] [UU]

unused devices: <none>
```

## Make the new disk bootable

If the new disk does not have a PReP partition or the PReP partition has some problem, it will not be bootable, here is an example on how to make the new disk bootable, you may need to substitute the device name with your own values.

- [RHEL]:

```
mkofboot .b /dev/sda
bootlist -m normal sda sdb
```

- [SLES]:

```
parted -s /dev/sda mkfs 1 fat32
parted /dev/sda set 1 type 6
parted /dev/sda set 1 boot on
dd if=/dev/sdb1 of=/dev/sda1
bootlist -m normal sda sdb
```

## Load Additional Drivers

### Overview

During the installing or netbooting of a node, the drivers in the initrd will be used to drive the devices like network cards and IO devices to perform the installation/netbooting tasks. But sometimes the drivers for the new devices were not included in the default initrd shipped by Red Hat or Suse. A solution is to inject the new drivers into the initrd to drive the new device during the installation/netbooting process.

Generally there are two approaches to inject the new drivers: **Driver Update Disk** and **Driver RPM package**.

A “**Driver Update Disk**” is media which contains the drivers, firmware and related configuration files for certain devices. The driver update disk is always supplied by the vendor of the device. One driver update disk can contain multiple drivers for different OS releases and different hardware architectures. Red Hat and Suse have different driver update disk formats.

The ‘**Driver RPM Package**’ is the rpm package which includes the drivers and firmware for the specific devices. The Driver RPM is the rpm package which is shipped by the Vendor of the device for a new device or a new kernel version.

xCAT supports both. But for ‘**Driver RPM Package**’ is only supported in xCAT 2.8 and later.

No matter which approach chosen, there are two steps to make new drivers work. one is locate the new driver’s path, another is inject the new drivers into the initrd.

### Locate the New Drivers

#### For Driver Update Disk

There are two approaches for xCAT to find the driver disk (pick one):

1. Specify the location of the driver disk in the osimage object (*This is ONLY supported in xCAT 2.8 and later*)

The value for the ‘driverupdatesrc’ attribute is a comma separated driver disk list. The tag ‘dud’ must be specified before the full path of ‘driver update disk’ to specify the type of the file:

```
chdef -t osimage <osimagename> driverupdatesrc=dud:<full path of driver disk>
```

1. Put the driver update disk in the directory <installroot>/driverdisk/<os>/<arch> (example: /install/driverdisk/sles11.1/x86\_64).

During the running of the `genimage`, `geninitrd`, or `nodeset` commands, xCAT will look for driver update disks in the directory <installroot>/driverdisk/<os>/<arch>.

## For Driver RPM Packages

The Driver RPM packages must be specified in the `osimage` object.

Three attributes of `osimage` object can be used to specify the Driver RPM location and Driver names. If you want to load new drivers in the `initrd`, the **'netdrivers'** attribute must be set. And one or both of the **'driverupdatesrc'** and **'osupdatename'** attributes must be set. If both of **'driverupdatesrc'** and **'osupdatename'** are set, the drivers in the **'driverupdatesrc'** have higher priority.

- `netdrivers` - comma separated driver names that need to be injected into the `initrd`. The postfix `'.ko'` can be ignored.

The `'netdrivers'` attribute must be set to specify the new driver list. If you want to load all the drivers from the driver rpms, use the keyword `allupdate`. Another keyword for the `netdrivers` attribute is `updateonly`, which means only the drivers located in the original `initrd` will be added to the newly built `initrd` from the driver rpms. This is useful to reduce the size of the new built `initrd` when the distro is updated, since there are many more drivers in the new kernel rpm than in the original `initrd`. Examples:

```
chdef -t osimage <osimagename> netdrivers=megaraid_sas.ko,igb.ko
chdef -t osimage <osimagename> netdrivers=allupdate
chdef -t osimage <osimagename> netdrivers=updateonly,igb.ko,new.ko
```

- `driverupdatesrc` - comma separated driver rpm packages (full path should be specified)

A tag named `'rpm'` can be specified before the full path of the rpm to specify the file type. The tag is optional since the default format is `'rpm'` if no tag is specified. Example:

```
chdef -t osimage <osimagename> driverupdatesrc=rpm:<full path of driver disk1>,rpm:<full_
↳path of driver disk2>
```

- `osupdatename` - comma separated `'osdistroudate'` objects. Each `'osdistroudate'` object specifies a Linux distro update.

When `geninitrd` is run, `kernel-*.rpm` will be searched in the `osdistroudate.dirpath` to get all the rpm packages and then those rpms will be searched for drivers. Example:

```
mkdef -t osdistroudate update1 dirpath=/install/<os>/<arch>
chdef -t osimage <osimagename> osupdatename=update1
```

If `'osupdatename'` is specified, the kernel shipped with the `'osupdatename'` will be used to load the newly built `initrd`, then only the drivers matching the new kernel will be kept in the newly built `initrd`. If trying to use the `'osupdatename'`, the `'allupdate'` or `'updateonly'` should be added in the `'netdrivers'` attribute, or all the necessary driver names for the new kernel need to be added in the `'netdrivers'` attribute. Otherwise the new drivers for the new kernel will be missed in newly built `initrd`. ..

## Inject the Drivers into the initrd

### For Driver Update Disk

- If specifying the driver disk location in the osimage, there are two ways to inject drivers:

1. Using nodeset command only:

```
nodeset <noderange> osimage=<osimagename>
```

2. Using geninitrd with nodeset command:

```
geninitrd <osimagename>  
nodeset <noderange> osimage=<osimagename> --noupdateinitrd
```

---

**Note:** ‘geninitrd’ + ‘nodeset –noupdateinitrd’ is useful when you need to run nodeset frequently for a diskful node. ‘geninitrd’ only needs be run once to rebuild the initrd and ‘nodeset –noupdateinitrd’ will not touch the initrd and kernel in /tftpboot/xcat/osimage/<osimage name>/.

---

- If putting the driver disk in <installroot>/driverdisk/<os>/<arch>:

Running ‘nodeset <noderange>’ in anyway will load the driver disk

### For Driver RPM Packages

There are two ways to inject drivers:

1. Using nodeset command only:

```
nodeset <noderange> osimage=<osimagename> [--ignorekernelchk]
```

2. Using geninitrd with nodeset command:

```
geninitrd <osimagename> [--ignorekernelchk]  
nodeset <noderange> osimage=<osimagename> --noupdateinitrd
```

---

**Note:** ‘geninitrd’ + ‘nodeset –noupdateinitrd’ is useful when you need to run nodeset frequently for diskful nodes. ‘geninitrd’ only needs to be run once to rebuild the initrd and ‘nodeset –noupdateinitrd’ will not touch the initrd and kernel in /tftpboot/xcat/osimage/<osimage name>/.

---

The option ‘–ignorekernelchk’ is used to skip the kernel version checking when injecting drivers from osimage.driverupdatesrc. To use this flag, you should make sure the drivers in the driver rpms are usable for the target kernel. ..



## Notes

- If the drivers from the driver disk or driver rpm are not already part of the installed or booted system, it's necessary to add the rpm packages for the drivers to the .pkglist or .otherpkglist of the osimage object to install them in the system.
- If a driver rpm needs to be loaded, the osimage object must be used for the 'nodeset' and 'genimage' command, instead of the older style profile approach.
- Both a Driver disk and a Driver rpm can be loaded in one 'nodeset' or 'genimage' invocation.

## Configure Disk Partition

By default, xCAT will attempt to determine the first physical disk and use a generic default partition scheme for the operating system. You may require a more customized disk partitioning scheme and can accomplish this in one of the following methods:

- partition definition file
- partition definition script

**Note:** **partition definition file** can be used for RedHat, SLES, and Ubuntu. However, disk configuration for Ubuntu is different from RedHat/SLES, there may be some special sections required for Ubuntu.

**Warning:** **partition definition script** has only been verified on RedHat and Ubuntu, use at your own risk for SLES.

## Partition Definition File

The following steps are required for this method:

1. Create a partition file
2. Associate the partition file with an xCAT osimage

The `nodeset` command will then insert the contents of this partition file into the generated autoinst config file that will be used by the operation system installer.

## Create Partition File

The partition file must follow the partitioning syntax of the respective installer

- Redhat: [Kickstart documentation](#)
  - The file `/root/anaconda-ks.cfg` is a sample kickstart file created by RedHat installing during the installation process based on the options that you selected.
  - `system-config-kickstart` is a tool with graphical interface for creating kickstart files
- SLES: [Autoyast documentation](#)
  - Use `yast2 autoyast` in GUI or CLI mode to customize the installation options and create autoyast file

- Use yast2 clone\_system to create autoyast configuration file /root/autoinst.xml to clone an existing system
- Ubuntu: [Preseed documentation](#)
  - For detailed information see the files partman-auto-recipe.txt and partman-auto-raid-recipe.txt included in the debian-installer package. Both files are also available from the debian-installer source repository.

---

**Note:** Supported functionality may change between releases of the Operating System, always refer to the latest documentation provided by the operating system.

---

Here is partition definition file example for Ubuntu standard partition in ppc64le machines

```
ubuntu-boot ::
8 1 1 prep
    $primary{ } $bootable{ } method{ prep }
.
500 10000 10000000000 ext4
    method{ format } format{ } use_filesystem{ } filesystem{ ext4 } mountpoint{ / }
.
2048 512 300% linux-swap
    method{ swap } format{ }
.
```

### Associate Partition File with Osmage

If your custom partition file is located at: /install/custom/my-partitions, run the following command to associate the partition file with an osimage:

```
chdef -t osimage <osimagename> partitionfile=/install/custom/my-partitions
```

To generate the configuration, run the nodeset command:

```
nodeset <nodename> osimage=<osimagename>
```

---

**Note: RedHat:** Running nodeset will generate the /install/autoinst file for the node. It will replace the #XCAT\_PARTITION\_START# and #XCAT\_PARTITION\_END# directives with the contents of your custom partition file.

---

---

**Note: SLES:** Running nodeset will generate the /install/autoinst file for the node. It will replace the #XCAT-PARTITION-START# and #XCAT-PARTITION-END# directives with the contents of your custom partition file. Do not include <partitioning config:type="list"> and </partitioning> tags, they will be added by xCAT.

---

---

**Note: Ubuntu:** Running nodeset will generate the /install/autoinst file for the node. It will write the partition file to /tmp/partitionfile and replace the #XCA\_PARTMAN\_RECIPE\_SCRIPT# directive in /install/autoinst/<node>.pre with the contents of your custom partition file.

---

## Partitioning disk file(For Ubuntu only)

The disk file contains the name of the disks to partition in traditional, non-devfs format and delimited with space “ ”, for example :

```
/dev/sda /dev/sdb
```

If not specified, the default value will be used.

### Associate partition disk file with osimage

```
chdef -t osimage <osimagename> -p partitionfile='d:/install/custom/partitiondisk'
nodeset <nodename> osimage=<osimage>
```

- the d: preceding the filename tells nodeset that this is a partition disk file.
- For Ubuntu, when nodeset runs and generates the /install/autoinst file for a node, it will generate a script to write the content of the partition disk file to /tmp/install\_disk, this context to run the script will replace the #XCA\_PARTMAN\_DISK\_SCRIPT# directive in /install/autoinst/<node>.pre.

## Additional preseed configuration file(For Ubuntu only)

To support other specific partition methods such as RAID or LVM in Ubuntu, some additional preseed configuration entries should be specified.

If using file way, c:<the absolute path of the additional preseed config file>, the additional preseed config file contains the additional preseed entries in d-i ... syntax. When nodeset, the #XCA\_PARTMAN\_ADDITIONAL\_CFG# directive in /install/autoinst/<node> will be replaced with content of the config file. For example:

```
d-i partman-auto/method string raid
d-i partman-md/confirm boolean true
```

If not specified, the default value will be used. ..

## Partition Definition Script

Create a shell script that will be run on the node during the install process to dynamically create the disk partitioning definition. This script will be run during the OS installer %pre script on RedHat or preseed/early\_command on Unbuntu execution and must write the correct partitioning definition into the file /tmp/partitionfile on the node

### Create Partition Script

The purpose of the partition script is to create the /tmp/partionfile that will be inserted into the kick-start/autoyast/preseed template, the script could include complex logic like select which disk to install and even configure RAID, etc

**Note:** the partition script feature is not thoroughly tested on SLES, there might be problems, use this feature on SLES at your own risk.

Here is an example of the partition script on RedHat and SLES, the partitioning script is /install/custom/my-partitions.sh:

```

instdisk="/dev/sda"

modprobe ext4 >& /dev/null
modprobe ext4dev >& /dev/null
if grep ext4dev /proc/filesystems > /dev/null; then
    FSTYPE=ext3
elif grep ext4 /proc/filesystems > /dev/null; then
    FSTYPE=ext4
else
    FSTYPE=ext3
fi
BOOTFSTYPE=ext4
EFIFSTYPE=vfat
if uname -r|grep ^3.*el7 > /dev/null; then
    FSTYPE=xfs
    BOOTFSTYPE=xfs
    EFIFSTYPE=efi
fi

if [ `uname -m` = "ppc64" ]; then
    echo 'part None --fstype "PPC PReP Boot" --ondisk '$instdisk' --size 8' >> /tmp/
↪partitionfile
fi
if [ -d /sys/firmware/efi ]; then
    echo 'bootloader --driveorder='$instdisk >> /tmp/partitionfile
    echo 'part /boot/efi --size 50 --ondisk '$instdisk' --fstype $EFIFSTYPE' >> /tmp/
↪partitionfile
else
    echo 'bootloader' >> /tmp/partitionfile
fi

echo "part /boot --size 512 --fstype $BOOTFSTYPE --ondisk $instdisk" >> /tmp/
↪partitionfile
echo "part swap --recommended --ondisk $instdisk" >> /tmp/partitionfile
echo "part / --size 1 --grow --ondisk $instdisk --fstype $FSTYPE" >> /tmp/partitionfile

```

The following is an example of the partition script on Ubuntu, the partitioning script is /install/custom/my-partitions.sh:

```

if [ -d /sys/firmware/efi ]; then
    echo "ubuntu-efi ::" > /tmp/partitionfile
    echo "    512 512 1024 fat32" >> /tmp/partitionfile
    echo '    $iflabel{ gpt } $reusemethod{ } method{ efi } format{ }' >> /tmp/
↪partitionfile
    echo "    ." >> /tmp/partitionfile
else
    echo "ubuntu-boot ::" > /tmp/partitionfile
    echo "100 50 100 ext4" >> /tmp/partitionfile
    echo '    $primary{ } $bootable{ } method{ format } format{ } use_filesystem{ }_
↪filesystem{ ext4 } mountpoint{ /boot }' >> /tmp/partitionfile
    echo "    ." >> /tmp/partitionfile
fi
echo "500 10000 10000000000 ext4" >> /tmp/partitionfile

```

(continues on next page)

(continued from previous page)

```
echo "    method{ format } format{ } use_filesystem{ } filesystem{ ext4 } mountpoint{ / }
↪" >> /tmp/partitionfile
echo "    ." >> /tmp/partitionfile
echo "2048 512 300% linux-swaps" >> /tmp/partitionfile
echo "    method{ swap } format{ }" >> /tmp/partitionfile
echo "    ." >> /tmp/partitionfile
```

## Associate partition script with osimage

Run below commands to associate partition script with osimage:

```
chdef -t osimage <osimagename> partitionfile='s:/install/custom/my-partitions.sh'
nodeset <nodename> osimage=<osimage>
```

- The `s:` preceding the filename tells nodeset that this is a script.
- For RedHat, when nodeset runs and generates the `/install/autoinst` file for a node, it will add the execution of the contents of this script to the `%pre` section of that file. The nodeset command will then replace the `#XCAT_PARTITION_START#...#XCAT_PARTITION_END#` directives from the osimage template file with `%include /tmp/partitionfile` to dynamically include the tmp definition file your script created.
- For Ubuntu, when nodeset runs and generates the `/install/autoinst` file for a node, it will replace the `#XCA_PARTMAN_RECIPE_SCRIPT#` directive and add the execution of the contents of this script to the `/install/autoinst/<node>.pre`, the `/install/autoinst/<node>.pre` script will be run in the `preseed/early_command`.

## Partitioning disk script (For Ubuntu only)

The disk script contains a script to generate a partitioning disk file named `/tmp/install_disk`. for example:

```
rm /tmp/devs-with-boot 2>/dev/null || true;
for d in $(list-devices partition); do
    mkdir -p /tmp/mymount;
    rc=0;
    mount $d /tmp/mymount || rc=$?;
    if [[ $rc -eq 0 ]]; then
        [[ -d /tmp/mymount/boot ]] && echo $d >>/tmp/devs-with-boot;
        umount /tmp/mymount;
    fi
done;
if [[ -e /tmp/devs-with-boot ]]; then
    head -n1 /tmp/devs-with-boot | egrep -o '\S+[^0-9]' > /tmp/install_disk;
    rm /tmp/devs-with-boot 2>/dev/null || true;
else
    DEV=`ls /dev/disk/by-path/* -l | egrep -o '/dev.*[s|h|v]d[^0-9]$' | sort -t : -k 1 -
↪k 2 -k 3 -k 4 -k 5 -k 6 -k 7 -k 8 -g | head -n1 | egrep -o '[s|h|v]d.*$`';
    if [[ "$DEV" == "" ]]; then DEV="sda"; fi;
    echo "/dev/$DEV" > /tmp/install_disk;
fi;
```

If not specified, the default value will be used.

### Associate partition disk script with osimage

```
chdef -t osimage <osimagename> -p partitionfile='s:d:/install/custom/partitiondiskscript'
nodeset <nodename> osimage=<osimage>
```

- the `s:` prefix tells `nodeset` that is a script, the `s:d:` preceding the filename tells `nodeset` that this is a script to generate the partition disk file.
- For Ubuntu, when `nodeset` runs and generates the `/install/autoinst` file for a node, this context to run the script will replace the `#XCA_PARTMAN_DISK_SCRIPT#` directive in `/install/autoinst/<node>.pre`.

### Additional preseed configuration script (For Ubuntu only)

To support other specific partition methods such as RAID or LVM in Ubuntu, some additional preseed configuration entries should be specified.

If using script way, `'s:c:<the absolute path of the additional preseed config script>'`, the additional preseed config script is a script to set the preseed values with `"debconf-set"`. When `"nodeset"`, the `#XCA_PARTMAN_ADDITIONAL_CONFIG_SCRIPT#` directive in `/install/autoinst/<node>.pre` will be replaced with the content of the script. For example:

```
debconf-set partman-auto/method string raid
debconf-set partman-md/confirm boolean true
```

If not specified, the default value will be used. ..

## Prescripts and Postscripts

### Using Prescript

The prescript table will allow you to run scripts before the install process. This can be helpful for performing advanced actions such as manipulating system services or configurations before beginning to install a node, or to prepare application servers for the addition of new nodes. Check the man page for more information.

`man prescripts`

The scripts will be run as root on the MASTER for the node. If there is a service node for the node, then the scripts will be run on the service node.

Identify the scripts to be run for each node by adding entries to the prescripts table:

```
tabedit prescripts
Or:
chdef -t node -o <noderange> prescripts-begin=<beginscripts> prescripts-end=<endscripts>
Or:
chdef -t group -o <nodegroup> prescripts-begin=<beginscripts> prescripts-end=<endscripts>

tabdump prescripts
#node,begin,end,comments,disable

begin or prescripts-begin - This attribute lists the scripts to be run at the beginning
↳ of the nodeset.
end or prescripts-end - This attribute lists the scripts to be run at the end of the
↳ nodeset.
```

## Format for naming prescripts

The general format for the prescripts-begin or prescripts-end attribute is:

```
[action1:]s1,s2...[|action2:s3,s4,s5...]
```

where:

- action1 **and** action2 are the nodeset actions ( 'install', 'netboot',etc) specified **in** the command .
- s1 **and** s2 are the scripts to run **for** \_action1\_ **in** order.
- s3, s4, **and** s5 are the scripts to run **for** action2.

If actions are omitted, the scripts apply to all actions.

Examples:

- myscript1,myscript2 - run scripts for all supported commands
- install:myscript1,myscript2|netboot:myscript3 - Run scripts myscript1 and myscript2 for nodeset(install), runs myscript3 for nodeset(netboot).

All the scripts should be copied to /install/prescripts directory and made executable for root and world readable for mounting. If you have service nodes in your cluster with a local /install directory (i.e. /install is not mounted from the xCAT management node to the service nodes), you will need to synchronize your /install/prescripts directory to your service node anytime you create new scripts or make changes to existing scripts.

The following two environment variables will be passed to each script:

- NODES - a comma separated list of node names on which to run the script
- ACTION - current nodeset action.

By default, the script will be invoked once for all nodes. However, if #xCAT setting:MAX\_INSTANCE=<number> is specified in the script, the script will be invoked for each node in parallel, but no more than number of instances specified in <number> will be invoked at a time.

## Exit values for prescripts

If there is no error, a prescript should return with 0. If an error occurs, it should put the error message on the stdout and exit with 1 or any non zero values. The command (nodeset for example) that runs prescripts can be divided into 3 sections.

1. run begin prescripts
2. run other code
3. run end prescripts

If one of the prescripts returns 1, the command will finish the rest of the prescripts in that section and then exit out with value 1. For example, a node has three begin prescripts s1,s2 and s3, three end prescripts s4,s5,s6. If s2 returns 1, the prescript s3 will be executed, but other code and the end prescripts will not be executed by the command.

If one of the prescripts returns 2 or greater, then the command will exit out immediately. This only applies to the scripts that do not have #xCAT setting:MAX\_INSTANCE=<number>.

## Using Postscript

### Postscript Execution Order Summary

Diskful Stage	Scripts	Execute Order	
N/A	postinstall	Does not execute for diskfull install	
Install/Create	postscripts (execute before reboot)	1	postscripts.xcatdefaults
		2	osimage
		3	node
Boot/Reboot	postbootscripts	4	postscripts.xcatdefaults
		5	osimage
		6	node

xCAT automatically runs a few postscripts and postbootscripts that are delivered with xCAT to set up the nodes. You can also add your own scripts to further customize the nodes.

### Types of scripts

There are two types of scripts in the postscripts table ( postscripts and postbootscripts). The types are based on when in the install process they will be executed. Run the following for more information:

```
man postscripts
```

- **postscripts attribute** - List of scripts that should be run on this node after diskful installation or diskless boot.
  - **[RHEL]**  
 Postscripts will be run before the reboot.
  - **[SLES]**  
 Postscripts will be run after the reboot but before the `init.d` process. For Linux diskless deployment, the postscripts will be run at the `init.d` time, and xCAT will automatically add the list of postscripts from the postbootscripts attribute to run after postscripts list.
- **postbootscripts attribute** - list of postbootscripts that should be run on this Linux node at the `init.d` time after diskful installation reboot or diskless boot
- **xCAT**, by default, for diskful installs only runs the postbootscripts on the install and not on reboot. In xCAT a site table attribute `runbootscripts` is available to change this default behavior. If set to `yes` or `y` or `1`, then the postbootscripts will be run on install and on reboot. To avoid reinstallation, run `updatenode <nodes> -P setuppostbootscripts` command to update the value of `runbootscripts` attribute on the compute or service nodes.

---

**Note:** xCAT automatically adds the postscripts from the `xcatdefaults.postscripts` attribute of the table to run first on the nodes after install or diskless boot.

---



## Adding your own postscripts

To add your own script, place it in `/install/postscripts` on the management node. Make sure it is executable and world readable. Then add it to the `postscripts` table for the group of nodes you want it to be run on (or the `all` group if you want it run on all nodes).

To check what scripts will be run on your node during installation:

```
lsdef node1 | grep scripts
postbootscripts=otherpkgs
postscripts=syslog,remoteshell,syncfiles
```

You can pass parameters to the postscripts. For example:

```
script1 p1 p2,script2,....
```

`p1 p2` are the parameters to `script1`.

Postscripts could be placed in the subdirectories in `/install/postscripts` on management node, and specify `subdir/postscriptname` in the `postscripts` table to run the postscripts in the subdirectories. This feature could be used to categorize the postscripts for different purposes. For example:

```
mkdir -p /install/postscripts/subdir1
mkdir -p /install/postscripts/subdir2
cp postscript1 /install/postscripts/subdir1/
cp postscript2 /install/postscripts/subdir2/
chdef node1 -p postscripts=subdir1/postscript1,subdir2/postscript2
updatenode node1 -P
```

If some of your postscripts will affect the network communication between the management node and compute node, like restarting network or configuring bond, the postscripts execution might not be able to be finished successfully because of the network connection problems. Even if we put this postscript be the last postscript in the list, xCAT still may not be able to update the node status to be booted. The recommendation is to use the Linux `at` mechanism to schedule this network-killing postscript to be run at a later time. For example:

The user needs to add a postscript to customize the nics bonding setup, the nics bonding setup will break the network between the management node and compute node. User could use `at` to run this nic bonding postscripts after all the postscripts processes have been finished.

Write a script, `/install/postscripts/nicbondscript`, the `nicbondscript` simply calls the `confignicsbond` using `at`:

```
[root@xcatmn ~]#cat /install/postscripts/nicbondscript
#!/bin/bash
at -f ./confignicsbond now + 1 minute
[root@xcatmn ~]#
```

Then

```
chdef <nodename> -p postbootscripts=nicbondscript
```

## Recommended Postscript design

- Postscripts that you want to run anywhere on Linux, should be written in shell. This should be available on all OS's. If only on the service nodes, you can use Perl .
- Postscripts should log errors using the following command (local4 is the default xCAT syslog class). `logger -t xCAT -p local4.info "your info message"`.
- Postscripts should have good and error exit codes (i.e 0 and 1).
- Postscripts should be well documented. At the top of the script, the first few lines should describe the function and inputs and output. You should have comments throughout the script. This is especially important if using `regex`.

## PostScript/PostbootScript execution

When your script is executed on the node, all the attributes in the `site` table are exported as variables for your scripts to use. You can add extra attributes for yourself. See the sample `mypostscript` file below.

To run the postscripts, a script is built, so the above exported variables can be input. You can usually find that script in `/xcatpost` on the node and in the Linux case it is call `mypostscript`. A good way to debug problems is to go to the node and just run `mypostscript` and see errors. You can also check the `syslog` on the Management Node for errors.

When writing you postscripts, it is good to follow the example of the current postscripts and write errors to `syslog` and in shell. See Suggestions for writing scripts.

All attributes in the `site` table are exported and available to the postscript/postbootscript during execution. See the `mypostscript` file, which is generated and executed on the nodes to run the postscripts.

Example of `mypostscript`

```
#subroutine used to run postscripts
run_ps () {
logdir="/var/log/xcat"
mkdir -p $logdir
logfile="/var/log/xcat/xcat.log"
if [_-f_$1_]; then
  echo "Running postscript: $@" | tee -a $logfile
  ./$_ 2>&1 | tee -a $logfile
else
  echo "Postscript $1 does NOT exist." | tee -a $logfile
fi
}
# subroutine end
AUDITSKIPCMDS='tabdump,nodefs'
export AUDITSKIPCMDS
TEST='test'
export TEST
NAMESERVERS='7.114.8.1'
export NAMESERVERS
NTPSERVERS='7.113.47.250'
export NTPSERVERS
INSTALLLOC='/install'
export INSTALLLOC
DEFSERIALPORT='0'
export DEFSERIALPORT
```

(continues on next page)

(continued from previous page)

```

DEFSERIALSPEED='19200'
export DEFSERIALSPEED
DHCPINTERFACES="'xcat20RRmn|eth0;rra000-m|eth1'"
export DHCPINTERFACES
FORWARDERS='7.113.8.1,7.114.8.2'
export FORWARDERS
NAMESERVER='7.113.8.1,7.114.47.250'
export NAMESERVER
DB='postg'
export DB
BLADEMAXP='64'
export BLADEMAXP
FSPTIMEOUT='0'
export FSPTIMEOUT
INSTALLDIR='/install'
export INSTALLDIR
IPMIMAXP='64'
export IPMIMAXP
IPMIRETRIES='3'
export IPMIRETRIES
IPMITIMEOUT='2'
export IPMITIMEOUT
CONSOLEONDEMAND='no'
export CONSOLEONDEMAND
SITEMASTER=7.113.47.250
export SITEMASTER
MASTER=7.113.47.250
export MASTER
MAXSSH='8'
export MAXSSH
PPCMAXP='64'
export PPCMAXP
PPCRETRY='3'
export PPCRETRY
PPCTIMEOUT='0'
export PPCTIMEOUT
SHAREDFTTP='1'
export SHAREDFTTP
SNSYNCFILEDIR='/var/xcat/syncfiles'
export SNSYNCFILEDIR
TFTPDIR='/tftpboot'
export TFTPDIR
XCATDPORT='3001'
export XCATDPORT
XCATIPORT='3002'
export XCATIPORT
XCATCONFDIR='/etc/xcat'
export XCATCONFDIR
TIMEZONE='America/New_York'
export TIMEZONE
USENMAPFROMMN='no'
export USENMAPFROMMN

```

(continues on next page)

(continued from previous page)

```

DOMAIN='cluster.net'
export DOMAIN
USESSHONAIX='no'
export USESSHONAIX
NODE=rra000-m
export NODE
NFSSERVER=7.113.47.250
export NFSSERVER
INSTALLNIC=eth0
export INSTALLNIC
PRIMARYNIC=eth1
OSVER=fedora9
export OSVER
ARCH=x86_64
export ARCH
PROFILE=service
export PROFILE
PATH=`dirname $0`: $PATH
export PATH
NODESETSTATE='netboot'
export NODESETSTATE
UPDATENODE=1
export UPDATENODE
NTYPE=service
export NTYPE
MACADDRESS='00:14:5E:5B:51:FA'
export MACADDRESS
MONSERVER=7.113.47.250
export MONSERVER
MONMASTER=7.113.47.250
export MONMASTER
OSPKGS=bash,openssl,dhclient,kernel,openssh-server,openssh-clients,busybox-anaconda,vim-
minimal,rpm,bind,bind-utils,ksh,nfs-utils,dhcp,bzip2,rootfiles,vixie-cron,wget,vsftpd,
↪ ntp,rsync
OTHERPKGS1=xCATsn,xCAT-rmc,rsct/rsct.core,rsct/rsct.core.utils,rsct/src,yaboot-xcat
export OTHERPKGS1
OTHERPKGS_INDEX=1
export OTHERPKGS_INDEX
export NOSYNCFILES
# postscripts-start-here\n
run_ps ospkgs
run_ps script1 p1 p2
run_ps script2
# postscripts-end-here\n

```

The mypostscript file is generated according to the mypostscript.tmpl file.

## Using the mypostscript template

### Using the mypostscript template

xCAT provides a way for the admin to customize the information that will be provided to the postscripts/postbootscripts when they run on the node. This is done by editing the `mypostscript.tmpl` file. The attributes that are provided in the shipped `mypostscript.tmpl` file should not be removed. They are needed by the default xCAT postscripts.

The `mypostscript.tmpl`, is shipped in the `/opt/xcat/share/xcat/mypostscript` directory.

If the admin customizes the `mypostscript.tmpl`, they should copy the `mypostscript.tmpl` to `/install/postscripts/mypostscript.tmpl`, and then edit it. The `mypostscript` for each node will be named `mypostscript.<nodename>`. The generated `mypostscript.<nodename>` will be put in the `/tftpboot/mypostscripts` directory.

### site table precreatemyposcripts attribute

If the site table `precreatemyposcripts` attribute is set to 1 or yes, it will instruct xCAT at `nodeset` and `updatenode` time to query the db once for all of the nodes passed into the command and create the `mypostscript` file for each node and put them in a directory in `$TFTPDIR` (for example `/tftpboot`). The created `mypostscript.<nodename>` file in the `/tftpboot/mypostscripts` directory will not be regenerated unless another `nodeset` or `updatenode` command is run to that node. This should be used when the system definition has stabilized. It saves time on the `updatenode` or reboot by not regenerating the `mypostscript` file.

If the `precreatemyposcripts` attribute is yes, and a database change is made or xCAT code is upgraded, then you should run a new `nodeset` or `updatenode` to regenerate the `/tftpboot/mypostscript/mypostscript.<nodename>` file to pick up the latest database setting. The default for `precreatemyposcripts` is no/0.

When you run `nodeset` or `updatenode`, it will search the `/install/postscripts/mypostscript.tmpl` first. If the `/install/postscripts/mypostscript.tmpl` exists, it will use that template to generate the `mypostscript` for each node. Otherwise, it will use `/opt/xcat/share/xcat/mypostscript/mypostscript.tmpl`.

### Content of the template for mypostscript

**Note:** The attributes that are defined in the shipped `mypostscript.tmpl` file should not be removed. The xCAT default postscripts rely on that information to run successfully.

The following will explain the entries in the `mypostscript.tmpl` file.

The `SITE_TABLE_ALL_ATTRIBS_EXPORT` line in the file directs the code to export all attributes defined in the site table. The attributes are not always defined exactly as in the site table to avoid conflict with other table attributes of the same name. For example, the site table master attribute is named `SITEMASTER` in the generated `mypostscript` file.

```
#SITE_TABLE_ALL_ATTRIBS_EXPORT#
```

The following line exports `ENABLESSHBEETWEENNODES` by running the internal xCAT routine (`enablenessshbetweennodes`).

```
ENABLESSHBEETWEENNODES=#Subroutine:xCAT::Template::enablenessshbetweennodes:$NODE#
export ENABLESSHBEETWEENNODES
```

`tabdump(<TABLENAME>)` is used to get all the information in the `<TABLENAME>` table

```
tabdump(networks)
```

These line export the node name based on its definition in the database.

```
NODE=$NODE
export NODE
```

These lines get a comma separated list of the groups to which the node belongs.

```
GROUP=#TABLE:nodelist:$NODE:groups#
export GROUP
```

These lines reads the nodesres table, the given attributes (nfsserver, installnic, primarynic, xcatmaster, routenames) for the node (\$NODE), and exports it.

```
NFSSERVER=#TABLE:noderes:$NODE:nfsserver#
export NFSSERVER
INSTALLNIC=#TABLE:noderes:$NODE:installnic#
export INSTALLNIC
PRIMARYNIC=#TABLE:noderes:$NODE:primarynic#
export PRIMARYNIC
MASTER=#TABLE:noderes:$NODE:xcatmaster#
export MASTER
NODEROUTENAMES=#TABLE:noderes:$NODE:routenames#
export NODEROUTENAMES
```

The following entry exports multiple variables from the routes table. Not always set.

```
#ROUTES_VARS_EXPORT#
```

The following lines export nodetype table attributes.

```
OSVER=#TABLE:nodetype:$NODE:os#
export OSVER
ARCH=#TABLE:nodetype:$NODE:arch#
export ARCH
PROFILE=#TABLE:nodetype:$NODE:profile#
export PROFILE
PROVMETHOD=#TABLE:nodetype:$NODE:provmethod#
export PROVMETHOD
```

The following adds the current directory to the path for the postscripts.

```
PATH=`dirname $0`: $PATH
export PATH
```

The following sets the NODESETSTATE by running the internal xCAT getnodesetstate script.

```
NODESETSTATE=#Subroutine:xCAT::Postage::getnodesetstate:$NODE#
export NODESETSTATE
```

The following says the postscripts are not being run as a result of updatenode. (This is changed =1, when updatenode runs).

```
UPDATENODE=0
export UPDATENODE
```

The following sets the NTYPE to compute, service or MN.

```
NTYPE=$NTYPE
export NTYPE
```

The following sets the mac address.

```
MACADDRESS=#TABLE:mac:$NODE:mac#
export MACADDRESS
```

If vlan is setup, then the #VLAN\_VARS\_EXPORT# line will provide the following exports:

```
VMNODE='YES'
export VMNODE
VLANID=vlan1...
export VLANID
VLANHOSTNAME=..
..
#VLAN_VARS_EXPORT#
```

If monitoring is setup, then the #MONITORING\_VARS\_EXPORT# line will provide:

```
MONSERVER=11.10.34.108
export MONSERVER
MONMASTER=11.10.34.108
export MONMASTER
#MONITORING_VARS_EXPORT#
```

The #OSIMAGE\_VARS\_EXPORT# line will provide, for example:

```
OSPKGDIR=/install/<os>/<arch>
export OSPKGDIR
OSPKGS='bash,nfs-utils,openssl,dhclient,kernel,openssh-server,openssh-clients,busybox,
↪ wget,rsyslog,dash,vim-minimal,ntp,rsyslog,rpm,rsync,
  ppc64-utils,iputils,dracut,dracut-network,e2fsprogs,bc,lsnvd,irqbalance,procps,yum'
export OSPKGS
#OSIMAGE_VARS_EXPORT#
```

THE #NETWORK\_FOR\_DISKLESS\_EXPORT# line will provide diskless networks information, if defined.

```
NETMASK=255.255.255.0
export NETMASK
GATEWAY=8.112.34.108
export GATEWAY
..
#NETWORK_FOR_DISKLESS_EXPORT#
```

**Note:** The #INCLUDE\_POSTSCRIPTS\_LIST# and the #INCLUDE\_POSTBOOTSCRIPTS\_LIST# sections in /tftpboot/mypostscript(mypostbootscripts) on the Management Node will contain all the postscripts and postbootscripts

defined for the node. When running an `updatenode` command for only some of the scripts, you will see in the `/xcatpost/mypostscript` file on the node, the list has been redefined during the execution of `updatenode` to only run the requested scripts. For example, if you run `updatenode <nodename> -P syslog`.

The `#INCLUDE_POSTSCRIPTS_LIST#` flag provides a list of postscripts defined for this `$NODE`.

```
#INCLUDE_POSTSCRIPTS_LIST#
```

For example, you will see in the generated file the following stanzas:

```
# postscripts-start-here
# defaults-postscripts-start-here
syslog
remoteshell
# defaults-postscripts-end-here
# node-postscripts-start-here
syncfiles
# node-postscripts-end-here
```

The `#INCLUDE_POSTBOOTSCRIPTS_LIST#` provides a list of postbootscripts defined for this `$NODE`.

```
#INCLUDE_POSTBOOTSCRIPTS_LIST#
```

For example, you will see in the generated file the following stanzas:

```
# postbootscripts-start-here
# defaults-postbootscripts-start-here
otherpkgs
# defaults-postbootscripts-end-here
# node-postbootscripts-end-here
# postbootscripts-end-here
```

## Kinds of variables in the template

**Type 1:** For the simple variable, the syntax is as follows. The `mypostscript.tmpl` has several examples of this. `$NODE` is filled in by the code. `UPDATENODE` is changed to 1, when the postscripts are run by `updatenode`. `$NTYPE` is filled in as either `compute`, `service` or `MN`.

```
NODE=$NODE
export NODE
UPDATENODE=0
export UPDATENODE
NTYPE=$NTYPE
export NTYPE
```

**Type 2:** This is the syntax to get the value of one attribute from the `<tablename>` and its key is `$NODE`. It does not support tables with two keys. Some of the tables with two keys are `litefile`, `prodkey`, `deps`, `monsetting`, `mpa`, `networks`. It does not support tables with keys other than `$NODE`. Some of the tables that do not use `$NODE` as the key, are `passwd`, `rack`, `token`

```
VARNAME=#TABLE:tablename:$NODE:attribute#
```

For example, to get the new `updatestatus` attribute from the `nodelist` table:



```
UPDATESTATUS=#TABLE:nodelist:$NODE:updatestatus#
export UPDATESTATUS
```

**Type 3:** The syntax is as follows:

```
VARNAME=#Subroutine:modulename::subroutine:$NODE#
or
VARNAME=#Subroutine:modulename::subroutine#
```

Examples in the `mypostscript.tmpl` are the following:

```
NODESETSTATE=#Subroutine:xCAT::Postage::getnodesetstate:$NODE#
export NODESETSTATE
ENABLESSHBEETWEENNODES=#Subroutine:xCAT::Template::enablesshbetweennodes:$NODE#
export ENABLESSHBEETWEENNODES
```

**Note:** Type 3 is not an open interface to add extensions to the template.

**Type 4:** The syntax is `#FLAG#`. When parsing the template, the code generates all entries defined by `#FLAG#`, if they are defined in the database. For example: To export all values of all attributes from the `site` table. The tag is

```
#SITE_TABLE_ALL_ATTRIBS_EXPORT#
```

For the `#SITE_TABLE_ALL_ATTRIBS_EXPORT#` flag, the related subroutine will get the attributes' values and deal with the special case. such as: the `site.master` should be exported as "SITEMASTER". And if the `noderes.xcatmaster` exists, the `noderes.xcatmaster` should be exported as "MASTER", otherwise, we also should export `site.master` as the "MASTER".

Other examples are:

```
#VLAN_VARS_EXPORT# - gets all vlan related items
#MONITORING_VARS_EXPORT# - gets all monitoring configuration and setup data
#OSIMAGE_VARS_EXPORT# - get osimage related variables, such as ospkgdir, ospkgs ...
#NETWORK_FOR_DISKLESS_EXPORT# - gets diskless network information
#INCLUDE_POSTSCRIPTS_LIST# - includes the list of all postscripts for the node
#INCLUDE_POSTBOOTSCRIPTS_LIST# - includes the list of all postbootscripts for the node
```

**Note:** Type4 is not an open interface to add extensions to the template.

**Type 5:** Get all the data from the specified table. The `<TABLENAME>` should not be a node table, like `nodelist`. This should be handles with TYPE 2 syntax to get specific attributes for the `$NODE`. `tabdump` would result in too much data for a `nodetype` table. Also the `auditlog`, `eventlog` should not be in `tabdump` for the same reason. `site` table should not be specified, it is already provided with the `#SITE_TABLE_ALL_ATTRIBS_EXPORT#` flag. It can be used to get the data from the two key tables (like `switch`). The syntax is:

```
tabdump(<TABLENAME>)
```

## Edit mypostscript.tpl

### Add new attributes into mypostscript.tpl

When you add new attributes into the template, you should edit the `/install/postscripts/mypostscript.tpl` which you created by copying `/opt/xcat/share/xcat/mypostscript/mypostscript.tpl`. Make all additions before the `# postscripts-start-here` section. xCAT will first look in `/install/postscripts/mypostscript.tpl` for a file and then, if not found, will use the one in `/opt/xcat/share/xcat/mypostscript/mypostscript.tpl`.

For example:

```
UPDATESTATUS=#TABLE:nodelist:$NODE:updatestatus#
export UPDATESTATUS

...
# postscripts-start-here
#INCLUDE_POSTSCRIPTS_LIST#
## The following flag postscripts-end-here must not be deleted.
# postscripts-end-here
```

**Note:** If you have a hierarchical cluster, you must copy your new `mypostscript.tpl` to `/install/postscripts/mypostscript.tpl` on the service nodes, unless `/install/postscripts` directory is mounted from the MN to the service node.

### Remove attribute from mypostscript.tpl

If you want to remove an attribute that you have added, you should remove all the related lines or comment them out with `##`. For example, comment out the added lines.

```
##UPDATESTATUS=#TABLE:nodelist:$NODE:updatestatus#
##export UPDATESTATUS
```

## Test the new template

There are two quick ways to test the template.

1. If the node is up

```
updatenode <nodename> -P syslog
```

Check your generated mypostscript on the compute node:

```
vi /xcatpost/mypostscript
```

2. Set the `precreatemypostscripts` option

```
chdef -t site -o clustersite precreatemypostscripts=1
```

Then run

```
nodeset <nodename> ....
```

Check your generated mypostscript

```
vi /tftpboot/mypostscripts/mypostscript.<nodename>
```

### Sample /xcatpost/mypostscript

This is an example of the generated postscript for a servicenode install. It is found in /xcatpost/mypostscript on the node.

```
# global value to store the running status of the postbootscripts,the value
#is non-zero if one postbootscript failed
return_value=0
# subroutine used to run postscripts
run_ps () {
    local ret_local=0
    logdir="/var/log/xcat"
    mkdir -p $logdir
    logfile="/var/log/xcat/xcat.log"
    if [ -f $1 ]; then
        echo "`date` Running postscript: $@" | tee -a $logfile
        #./$@ 2>&1 1> /tmp/tmp4xcatlog
        #cat /tmp/tmp4xcatlog | tee -a $logfile
        ./$@ 2>&1 | tee -a $logfile
        ret_local=${PIPESTATUS[0]}
        if [ "$ret_local" -ne "0" ]; then
            return_value=$ret_local
        fi
        echo "Postscript: $@ exited with code $ret_local"
    else
        echo "`date` Postscript $1 does NOT exist." | tee -a $logfile
        return_value=-1
    fi
    return 0
}
# subroutine end
SHAREDFTFTP='1'
export SHAREDFTFTP
TFTPDIR='/tftpboot'
export TFTPDIR
CONSOLEONDEMAND='yes'
export CONSOLEONDEMAND
PPCTIMEOUT='300'
export PPCTIMEOUT
VSFTP='y'
export VSFTP
DOMAIN='cluster.com'
export DOMAIN
XCATIPORT='3002'
export XCATIPORT
DHCPINTERFACES="'xcatmn2|eth1;service|eth1'"
export DHCPINTERFACES
MAXSSH='10'
export MAXSSH
```

(continues on next page)

(continued from previous page)

```

SITEMASTER=10.2.0.100
export SITEMASTER
TIMEZONE='America/New_York'
export TIMEZONE
INSTALLDIR='/install'
export INSTALLDIR
NTPSERVERS='xcatmn2'
export NTPSERVERS
EA_PRIMARY_HMC='c76v2hmc01'
export EA_PRIMARY_HMC
NAMESERVERS='10.2.0.100'
export NAMESERVERS
SNSYNCFILEDIR='/var/xcat/syncfiles'
export SNSYNCFILEDIR
DISJOINTDHCP='0'
export DISJOINTDHCP
FORWARDERS='8.112.8.1,8.112.8.2'
export FORWARDERS
VLANNETS='|(\d+)|10.10.($1+0).0|'
export VLANNETS
XCATDPORT='3001'
export XCATDPORT
USENMAPFROMMN='no'
export USENMAPFROMMN
DNSHANDLER='ddns'
export DNSHANDLER
ROUTENAMES='r1,r2'
export ROUTENAMES
INSTALLLOC='/install'
export INSTALLLOC
ENABLESSHBETWEENNODES=YES
export ENABLESSHBETWEENNODES
NETWORKS_LINES=4
export NETWORKS_LINES
NETWORKS_LINE1='netname=public_net|net=8.112.154.64|mask=255.255.255.
↪192|mgtifname=eth0|gateway=8.112.154.126|dhcpserver=|tftpserver=8.112.154.
↪69|nameservers=8.112.8.
↪1|ntpserver=|logserver=|dynamicrange=|staticrange=|staticrangeincrement=|nodehostname=|ddnsd
↪'
export NETWORKS_LINE2
NETWORKS_LINE3='netname=sn21_net|net=10.2.1.0|mask=255.255.255.
↪0|mgtifname=eth1|gateway=<xcatmaster>|dhcpserver=|tftpserver=|nameservers=10.2.1.
↪100,10.2.1.
↪101|ntpserver=|logserver=|dynamicrange=|staticrange=|staticrangeincrement=|nodehostname=|ddns
↪'
export NETWORKS_LINE3
NETWORKS_LINE4='netname=sn22_net|net=10.2.2.0|mask=255.255.255.
↪0|mgtifname=eth1|gateway=10.2.2.100|dhcpserver=10.2.2.100|tftpserver=10.2.2.
↪100|nameservers=10.2.2.100|ntpserver=|logserver=|dynamicrange=10.2.2.120-10.2.2.
↪250|staticrange=|staticrangeincrement=|nodehostname=|ddnsdomain=|vlanid=|domain=|mtu=|disab
↪'
export NETWORKS_LINE4

```

(continues on next page)

(continued from previous page)

```

NODE=xcatsn23
export NODE
NFSSERVER=10.2.0.100
export NFSSERVER
INSTALLNIC=eth0
export INSTALLNIC
PRIMARYNIC=eth0
export PRIMARYNIC
MASTER=10.2.0.100
export MASTER
OSVER=sles11
export OSVER
ARCH=ppc64
export ARCH
PROFILE=service-xcattest
export PROFILE
PROVMETHOD=netboot
export PROVMETHOD
PATH=`dirname $0`: $PATH
export PATH
NODESETSTATE=netboot
export NODESETSTATE
UPDATENODE=1
export UPDATENODE
NTYPE=service
export NTYPE
MACADDRESS=16:3d:05:fa:4a:02
export MACADDRESS
NODEID=EA163d05fa4a02EA
export NODEID
MONSERVER=8.112.154.69
export MONSERVER
MONMASTER=10.2.0.100
export MONMASTER
MS_NODEID=0360238fe61815e6
export MS_NODEID
OSPKGS='kernel-ppc64,udev,sysconfig,aaa_base,klogd,device-mapper,bash,openssl,nfs- utils,
↪ksh,syslog-ng,openssh,openssh-askpass,busybox,vim,rpm,bind,bind-utils,dhcp,dhcpd,dhcp-
↪server,dhcp-client,dhcp-relay,bzip2,cron,wget,vsftpd,util-linux,module-init-tools,
↪mkinitrd,apache2,apache2-prefork,perl-Bootloader,psmisc,procps,dbus-1,hal,timezone,
↪rsync,powerpc-utils,bc,iputils,uuid-runtime,unixODBC,gcc,zypper,tar'
export OSPKGS
OTHERPKGS1='xcat/xcat-core/xCAT-rmc,xcat/xcat-core/xCATsn,xcat/xcat-dep/sles11/ppc64/
↪conserver,perl-DBD-mysql,nagios/nagios-nsca-client,nagios/nagios,nagios/nagios-plugins-
↪nrpe,nagios/nagios-nrpe'
export OTHERPKGS1
OTHERPKGS_INDEX=1
export OTHERPKGS_INDEX
## get the diskless networks information. There may be no information.
NETMASK=255.255.255.0
export NETMASK
GATEWAY=10.2.0.100

```

(continues on next page)

(continued from previous page)

```

export GATEWAY
# NIC related attributes for the node for confignetwork postscript
NICIPS=""
export NICIPS
NICHOSTNAMESUFFIXES=""
export NICHOSTNAMESUFFIXES
NICTYPES=""
export NICTYPES
NICCUSTOMSCRIPTS=""
export NICCUSTOMSCRIPTS
NICNETWORKS=""
export NICNETWORKS
NICCOMMENTS=
export NICCOMMENTS
# postscripts-start-here
# defaults-postscripts-start-here
run_ps test1
run_ps syslog
run_ps remoteshell
run_ps syncfiles
run_ps confNagios
run_ps configrmcnode
# defaults-postscripts-end-here
# node-postscripts-start-here
run_ps servicenode
run_ps configeth_new
# node-postscripts-end-here
run_ps setbootfromnet
# postscripts-end-here
# postbootscripts-start-here
# defaults-postbootscripts-start-here
run_ps otherpkgs
# defaults-postbootscripts-end-here
# node-postbootscripts-start-here
run_ps test
# The following line node-postbootscripts-end-here must not be deleted.
# node-postbootscripts-end-here
# postbootscripts-end-here
exit $return_value

```

## Suggestions

### For writing scripts

- Some compute node profiles exclude perl to keep the image as small as possible. If this is your case, your postscripts should obviously be written in another shell language, e.g. **bash**, **ksh**.
- If a postscript is specific for an os, name your postscript mypostscript.osname.
- Add logger statements to send errors back to the Management Node. By default, xCAT configures the syslog service on compute nodes to forward all syslog messages to the Management Node. This will help debug.

## Using Hierarchical Clusters

If you are running a hierarchical cluster, one with Service Nodes. If your `/install/postscripts` directory is not mounted on the Service Node. You are going to need to sync or copy the postscripts that you added or changed in the **/install/postscripts** on the MN to the SN, before running them on the compute nodes. To do this easily, use the `xdcp` command and just copy the entire **/install/postscripts** directory to the servicenodes ( usually in `/xcatpost` ).

```
xdcp service -R /install/postscripts/* /xcatpost
or
prsync /install/postscripts service:/xcatpost
```

If your **/install/postscripts** is not mounted on the Service Node, you should also:

```
xdcp service -R /install/postscripts/* /install
or
prsync /install/postscripts service:/install
```

## Synchronizing Files

### Overview

Synchronizing (sync) files to the nodes is a feature of xCAT used to distribute specific files from the management node to the newly-deploying or deployed nodes.

This function is supported for diskful or RAMdisk-based diskless nodes. Generally, the specific files are usually the system configuration files for the nodes in the `/etc` directory, like `/etc/hosts`, `/etc/resolve.conf`; it also could be the application programs configuration files for the nodes. The advantages of this function are: it can parallel sync files to the nodes or nodegroup for the installed nodes; it can automatically sync files to the newly-installing node after the installation. Additionally, this feature also supports the flexible format to define the files to be synced in a configuration file, called “synclist”.

The synclist file can be a common one for a group of nodes using the same profile or osimage, or can be the unique for each particular node. Since the location of the synclist file will be used to find the synclist file, the common synclist should be put in a given location for Linux nodes or specified by the osimage.

`xdcp` command supplies the basic Syncing File function. If the **-F synclist** option is specified in the `xdcp` command, it syncs files configured in the synclist to the nodes. If the **-i <install image path>** option is specified with **-F synclist**, it syncs files to the root image located in the `<install image path>` directory.

---

**Note:** The **-i <install image path>** option is only supported for Linux nodes

---

`xdcp` supports hierarchy where service nodes are used. If a node is serviced by a service node, `xdcp` will sync the files to the service node first, then sync the files from service node to the compute node. The files are placed in an intermediate directory on the service node defined by the **SNsyncfiledir** attribute in the **site** table. The default is `/var/xcat/syncfiles`.

Since `updatenode -F` calls the `xdcp` to handle the Syncing File function, the `updatenode -F` also supports the hierarchy.

For a new-installing nodes, the Syncing File action will be triggered when running the postscripts for the nodes. A special postscript named **syncfiles** is used to initiate the Syncing File process.

The postscript **syncfiles** is located in the `/install/postscripts/`. When running, it sends a message to the **xcatd** on the management node or service node, then the **xcatd** figures out the corresponding synclist file for the node and calls the `xdcp` command to sync files in the synclist to the node.

If installing nodes in a hierarchical configuration, you must sync the service nodes first to make sure they are updated. The compute nodes will be synced from their service nodes. You can use the `updatenode <computenodes> -f` command to sync all the service nodes for range of compute nodes provided.

For an installed nodes, the Syncing File action happens when performing the `updatenode -F` or `xdcp -F synclist` command to update a nodes. While performing the `updatenode -F`, it figures out the location of the synclist files for all the nodes and groups the nodes which will be using the same synclist file and then calls the `xdcp -F synclist` to sync files to the nodes.

## The synclist file

### The Format of synclist file

The synclist file contains the configuration entries that specify where the files should be synced to. In the synclist file, each line is an entry which describes the location of the source files and the destination location of files on the target node.

The basic entry format looks like following:

```
path_of_src_file1 -> path_of_dst_file1
path_of_src_file1 -> path_of_dst_directory
path_of_src_file1 path_of_src_file2 ... -> path_of_dst_directory
```

The `path_of_src_file*` should be the full path of the source file on the Management Node.

The `path_of_dst_file*` should be the full path of the destination file on target node. Make sure `path_of_dst_file*` is not a existing directory on target node, otherwise, the file sync with `updatenode -r /usr/bin/scp` or `xdcp -r /usr/bin/scp` will fail.

The `path_of_dst_directory` should be the full path of the destination directory. Make sure `path_of_dst_directory` is not a existing file on target node, otherwise, the file sync with `updatenode -r /usr/bin/scp` or `xdcp -r /usr/bin/scp` will fail.

If no target node is specified, the files will be synced to all nodes in the cluster. See “Support nodes in synclist file” below for how to specify a noderange.

The following synclist formats are supported:

sync file **/etc/file2** to the file **/etc/file2** on the node (with same file name)

```
/etc/file2 -> /etc/file2
```

sync file **/etc/file2** to the file **/etc/file3** on the node (with different file name)

```
/etc/file2 -> /etc/file3
```

sync file **/etc/file4** to the file **/etc/tmp/file5** on the node (different file name and directory). The directory will be automatically created for you.

```
/etc/file4 -> /etc/tmp/file5
```

sync the multiple files **/etc/file1**, **/etc/file2**, **/etc/file3**, ... to the directory **/tmp/etc** (**/tmp/etc** must be a directory when multiple files are synced at one time). If the directory does not exist, it will be created.

```
/etc/file1 /etc/file2 /etc/file3 -> /tmp/etc
```

sync file **/etc/file2** to the file **/etc/file2** on the node



```
/etc/file2 -> /etc/
```

sync all files, including subdirectories, in **/home/mikev** to directory **/home/mikev** on the node

```
/home/mikev/* -> /home/mikev/
or
/home/mikev -> /home/mikev/
```

**Note:** Don't try to sync files to the read only directory on the target node.

### An example of synclist file

Sync the file **/etc/common\_hosts** to the two places on the target node: put one to the **/etc/hosts**, the other to the **/tmp/etc/hosts**. Following configuration entries should be added

```
/etc/common_hosts -> /etc/hosts
/etc/common_hosts -> /tmp/etc/hosts
```

Sync files in the directory **/tmp/prog1** to the directory **/prog1** on the target node, and the postfix **.tmpl** needs to be removed on the target node. (directory **/tmp/prog1/** contains two files: **conf1.tmpl** and **conf2.tmpl**) Following configuration entries should be added

```
/tmp/prog1/conf1.tmpl -> /prog1/conf1
/tmp/prog1/conf2.tmpl -> /prog1/conf2
```

Sync the files in the directory **/tmp/prog2** to the directory **/prog2** with same name on the target node. (directory **/tmp/prog2** contains two files: **conf1** and **conf2**) Following configuration entries should be added:

```
/tmp/prog2/conf1 /tmp/prog2/conf2 -> /prog2
```

Sample synclist file

```
/etc/common_hosts -> /etc/hosts
/etc/common_hosts -> /tmp/etc/hosts
/tmp/prog1/conf1.tmpl -> /prog1/conf1
/tmp/prog1/conf2.tmpl -> /prog1/conf2
/tmp/prog2/conf1 /tmp/prog2/conf2 -> /prog2
/tmp/* -> /tmp/
/etc/testfile -> /etc/
```

If the above synclist file is used by the **updatenode/xdcp** commands, or used in a node installation process, the following files will exist on the target node with the following contents.

```
/etc/hosts(It has the same content with /etc/common_hosts on the MN)
/tmp/etc/hosts(It has the same content with /etc/common_hosts on the MN)
/prog1/conf1(It has the same content with /tmp/prog1/conf1.tmpl on the MN)
/prog1/conf2(It has the same content with /tmp/prog1/conf2.tmpl on the MN)
/prog2/conf1(It has the same content with /tmp/prog2/conf1 on the MN)
/prog2/conf2(It has the same content with /tmp/prog2/conf2 on the MN)
```

## Support nodes in synclist file

Starting with xCAT 2.9.2 on AIX and with xCAT 2.12 on Linux, xCAT supports a new format for syncfile. The new format is

```
file -> (noderange for permitted nodes) file
```

The noderange can have several formats. Following examples show that **/etc/hosts** file is synced to the nodes which are specified before the file name

```
/etc/hosts -> (node1,node2) /etc/hosts      # The /etc/hosts file is synced to
↪node1 and node2
/etc/hosts -> (node1-node4) /etc/hosts      # The /etc/hosts file is synced to
↪node1,node2,node3 and node4
/etc/hosts -> (node[1-4]) /etc/hosts        # The /etc/hosts file is synced to
↪node1, node2, node3 and node4
/etc/hosts -> (node1,node[2-3],node4) /etc/hosts # The /etc/hosts file is synced to
↪node1, node2, node3 and node4
/etc/hosts -> (group1) /etc/hosts           # The /etc/hosts file is synced to
↪nodes in group1
/etc/hosts -> (group1,group2) /etc/hosts     # The /etc/hosts file is synced to
↪nodes in group1 and group2
```

## Advanced synclist file features

### EXECUTE

The **EXECUTE** clause is used to list all the postsync scripts (<filename>.post) you would like to run after the files are synced, only if the file <filename> is updated. For hierarchical clusters, the postsync files in this list must also be added to the list of files to sync. It is optional for non-hierarchical clusters. If noderange is used in the synclist for the file listed in the **EXECUTE** clause, the postsync script will only be executed on the nodes in that noderange. The **EXECUTE** clause is not supported oif **-r /usr/bin/scp** option is used with **xdcp** or **updatenode** command.

### EXECUTEALWAYS

The **EXECUTEALWAYS** clause is used to list all the postsync scripts you would like to run after the files are synced, whether or not any file is actually updated. The files in this list must be added to the list of files to sync. If noderange is used in the synclist for the file listed in the **EXECUTEALWAYS** clause, the script will only be executed on the nodes in that noderange.

---

**Note:** The path to the file to **EXECUTE** or **EXECUTEALWAYS**, is the location of the file on the MN.

---

For example, your syncfile may look like this.:

```
/tmp/share/file2 -> /tmp/file2
/tmp/share/file2.post -> /tmp/file2.post (required for hierarchical clusters)
/tmp/share/file3 -> /tmp/filex
/tmp/share/file3.post -> /tmp/file3.post (required for hierarchical clusters)
/tmp/myscript1 -> /tmp/myscript1
/tmp/myscript2 -> /tmp/myscript2
# Postscripts
EXECUTE:
```

(continues on next page)

(continued from previous page)

```
/tmp/share/file2.post
/tmp/share/file3.post
EXECUTEALWAYS:
/tmp/myscript1
/tmp/myscript2
```

If **/tmp/file2** is updated on the node in **/tmp/file2**, then **/tmp/file2.post** is automatically executed on that node. If **/tmp/file3** is updated on the node in **/tmp/filex**, then **/tmp/file3.post** is automatically executed on that node.

## APPEND

The **APPEND** clause is used to append the contents of the input file to an existing file on the node. The file to be appended must already exist on the node and not be part of the synclist that contains the **APPEND** clause.

For example, your synclist file may look like this:

```
/tmp/share/file2 -> /tmp/file2
/tmp/share/file2.post -> /tmp/file2.post
/tmp/share/file3 -> /tmp/filex
/tmp/share/file3.post -> /tmp/file3.post
/tmp/myscript -> /tmp/myscript
# Postscripts
EXECUTE:
/tmp/share/file2.post
/tmp/share/file3.post
EXECUTEALWAYS:
/tmp/myscript
APPEND:
/etc/myappenddir/appendfile -> /etc/mysetup/setup
/etc/myappenddir/appendfile2 -> /etc/mysetup/setup2
```

When you use the **APPEND** clause, the source file to the left of the arrow is appended to the file to the right of the arrow. In this example, **/etc/myappenddir/appendfile** is appended to **/etc/mysetup/setup** file, which must already exist on the node. The **/opt/xcats/share/xcats/scripts/xdcpappend.sh** is used to accomplish this.

The script creates a backup of the original file on the node in the directory defined by the **site** table **nodesyncfiledir** attribute, which is **/var/xcats/node/syncfiles** by default. To update the original file when using the function, you need to sync a new original file to the node, removed the old original from the **/var/xcats/node/syncfiles/org** directory. If you want to cleanup all the files for the append function on the node, you can use **xdsh -c** command. See man page for **xdsh**.

**MERGE** (supported on Linux only).

The **MERGE** clause is used to append the contents of the input file to either the **/etc/passwd**, **/etc/shadow** or **/etc/group** files. They are the only supported files. You must not put the **/etc/passwd**, **/etc/shadow**, **/etc/group** files in an **APPEND** clause if using a **MERGE** clause. For these three files you should use the **MERGE** clause. The **APPEND** will add the information to the end of the file. The **MERGE** will add or replace the information and ensure that there are no duplicate entries in these files.

For example, your synclist file may look like this

```
/tmp/share/file2 -> /tmp/file2
/tmp/share/file2.post -> /tmp/file2.post
/tmp/share/file3 -> /tmp/filex
/tmp/share/file3.post -> /tmp/file3.post
/tmp/myscript -> /tmp/myscript
```

(continues on next page)

(continued from previous page)

```
# Postscripts
EXECUTE:
/tmp/share/file2.post
/tmp/share/file3.post
EXECUTEALWAYS:
/tmp/myscript
MERGE:
/etc/mydir/mergepasswd -> /etc/passwd
/etc/mydir/mergeshadow -> /etc/shadow
/etc/mydir/mergegroup -> /etc/group
```

When you use the **MERGE** clause, the source file to the left of the arrow is merged into the file to the right of the arrow. It will replace any common userids found in those files and add new userids. The `/opt/xcat/share/xcat/scripts/xdcpmerge.sh` is used to accomplish this.

**Note:** no order of execution may be assumed by the order of **EXECUTE**, **EXECUTEALWAYS**, **APPEND** and **MERGE** clauses in the synclist file.

## The location of synclist file for updatenode and install process

In the installation process or **updatenode** process, xCAT needs to figure out the location of the synclist file automatically, so the synclist should be put into the specified place with the proper name.

If the provisioning method for the node is an osimage name, then the path to the synclist will be read from the osimage definition **synclists** attribute. You can display this information by running the following command, supplying your osimage name.

```
lsdef -t osimage -l <os>-<arch>-netboot-compute

Object name: <os>-<arch>-netboot-compute
exlist=/opt/xcat/share/xcat/netboot/<os>/compute.exlist
imagetype=linux
osarch=<arch>
osname=Linux
osvers=<os>
otherpkgdir=/install/post/otherpkgs/<os>/<arch>
pkgdir=/install/<os>/<arch>
pkglist=/opt/xcat/share/xcat/netboot/<os>/compute.pkglist
profile=compute
provmethod=netboot
rootimgdir=/install/netboot/<os>/<arch>/compute
**synclists=/install/custom/netboot/compute.synclist**
```

You can set the synclist path using the following command

```
chdef -t osimage -o <os>-<arch>-netboot-compute synclists="/install/custom/netboot/
↪compute.synclist"
```

If the provisioning method for the node is *install*, or *netboot* then the path to the synclist should be in the following format

```
/install/custom/<inst_type>/<distro>/<profile>.<os>.<arch>.synclist
```

```
<inst_type>: "install", "netboot"
<distro>:    "rh", "centos", "fedora", "sles"
<profile>, <os> and <arch> are what you set for the node
```

For example: The location of synclist file for the diskful installation of RedHat 7.5 with **compute** as the profile

```
/install/custom/install/rh/compute.rhels7.5.synclist
```

The location of synclist file for the diskless netboot of SLES 12.3 with **service** as the profile

```
/install/custom/netboot/sles/service.sles12.3.synclist
```

## Run xdc command to perform Syncing File action

xdcp command supplies three options **-F**, **-s**, and **-i** to support the Syncing File function.

- **-F**—File <synclist input file>

Specifies the full path to the synclist file

- **-s**

Specifies to sync to the service nodes only for the input compute noderange.

- **-i** | **-rootimg** <install image for Linux>

Specifies the full path to the install image on the local node.

By default, if the **-F** option is specified, the **rsync** command is used to perform the syncing file function. Only the **ssh** remote shell is supported for **rsync**. **xdcp** uses the **-Lpotz** as the default flags to call the **rsync** command. More flags for **rsync** command can be specified by adding **-o** flag to the call to **xdcp**.

For example you can use **xdcp -F** option to sync files which are listed in the **/install/custom/commonsynfiles/<profile>.synclist** file to the node group named **compute**. If the node group **compute** is serviced by servicenodes, then the files will be automatically staged to the correct service nodes, and then synced to the compute nodes from those service nodes. The files will be stored in **/var/xcat/synfiles** directory on the service nodes by default, or in the directory indicated in the **site.SNsynfiledir** attribute. See **-s** option below.

```
xdcp compute -F /install/custom/commonsynfiles/<profile>.synclist
```

For Linux nodes, you can use **xdcp** command **-i** option with **-F** to sync files specified in the **/install/custom/<inst\_type>/<os>/<profile>.synclist** file to the osimage in the directory **/install/<inst\_type>/<os>/<arch>/<profile>/rootimg**:

```
xdcp -i /install/<inst_type>/<os>/<arch>/<profile>/rootimg -F /install/custom/<inst_type>
↪/<os>/<profile>.synclist
```

You can use the **xdcp -s** option to sync the files only to the service nodes for the node group named **compute**. The files will be placed in the default **/var/xcat/synfiles** directory or in the directory as indicated in the **site.SNsynfiledir** attribute. If you want the files synced to the same directory on the service node that they come from on the Management Node, set **site.SNsynfiledir=/** attribute. This can be setup before a node install, to have the files available to be synced during the install:

```
xdcp compute -s -F /install/custom/<inst_type>/<os>/<profile>.synclist
```

## Synchronizing Files during the installation process

The **policy** table must have the entry to allow **syncfiles** postscript to access the Management Node. Make sure this entry is in your **policy** table:

```
#priority,name,host,commands,noderange,parameters,time,rule,comments,disable
.
.
"4.6",,,,"syncfiles",,,,"allow",,
.
.
```

## Hierarchy and Service Nodes

If using Service nodes to manage you nodes, you should make sure that the service nodes have been synchronized with the latest files from the Management Node before installing. If you have a group of compute nodes **compute** that are going to be installed that are serviced by SN1, then run the following before the install to sync the current files to SN1.:

```
updatenode compute -f
```

---

**Note:** updatenode will figure out which service nodes need updating.

---

## Diskful installation

The **syncfiles** postscript is in the defaults section of the **postscripts** table. To enable the **syncfiles** postscript to sync files to the nodes during install the user need to do the following:

- Create the synclist file with the entries indicating which files should be synced. (refer to *The Format of synclist file* )
- Put the synclist into the proper location for the node type (refer to *The location of synclist file for updatenode and install process*)

Make sure your **postscripts** table has the syncfiles postscript listed:

```
#node,postscripts,postbootscripts,comments,disable
"xcatdefaults","syslog,remoteshell,syncfiles","otherpkgs",,
```

## Diskless Installation

The diskless boot is similar with the diskful installation for the synchronizing files operation, except that the **packimage** command will sync files to the root directories of image during the creating image process.

Creating the synclist file as the steps in Diskful installation section, then the synced files will be synced to the os image during the **packimage** and **mkdisklnode** commands running.

Also the files will always be re-synced during the booting up of the diskless node.

## Run the Sync'ing File action in the creating diskless image process

Different approaches are used to create the diskless image. The **Sync'ing** File action is also different.

The **packimage** command is used to prepare the root image files and package the root image. The Syncing File action is performed here.

Steps to make the Sync'ing File working in the **packimage** command:

1. Prepare the synclist file and put it into the appropriate location as describe above in (refer *The location of synclist file for updatenode and install process*)
2. Run **packimage** as is normally done.

## Run the Syncing File action in the updatenode process

Running **updatenode** command with **-F** option, will sync files configured in the synclist file to the nodes. **updatenode** does not sync images, use **xdcp -i -F** command to sync images.

**updatenode** can be used to sync files to to diskful or diskless nodes. **updatenode** cannot be used to sync files to statelite nodes.

Steps to make the Syncing File working in the **updatenode -F** command:

1. Create the synclist file with the entries indicating which files should be synced. (refer to *The Format of synclist file*)
2. Put the synclist into the proper location (refer to *The location of synclist file for updatenode and install process*).
3. Run the **updatenode <noderange> -F** command to initiate the Syncing File action.

**Note:** Since Syncing File action can be initiated by the **updatenode -F**, the **updatenode -P** does NOT support to re-run the **syncfiles** postscript, even if you specify the **syncfiles** postscript on the **updatenode** command line or set the **syncfiles** in the **postscripts.postscripts** attribute.

## Run the Syncing File action periodically

If the admins want to run the Syncing File action automatically or periodically, the **xdcp -F**, **xdcp -i -F** and **updatenode -F** commands can be used in the script, crontab or FAM directly.

For example:

Use the **cron** daemon to sync files in the **/install/custom/<inst\_type>/<distro>/<profile>.<os>.synclist** to the node-group **compute** every 10 minutes with the **xdcp** command by adding this to **crontab**. :

```
*/10 * * * * root /opt/xcat/bin/xdcp compute -F /install/custom/<inst_type>/<distro>/
-><profile>.<distro>.synclist
```

Use the **cron** daemon to sync files for the nodegroup **compute** every 10 minutes with **updatenode** command.

```
*/10 * * * * root /opt/xcat/bin/updatenode compute -F
```

## Add Additional Software Packages

### Overview

The name of the packages that will be installed on the node are stored in the packages list files. There are two kinds of package list files:

- The package list file contains the names of the packages that comes from the os distro. They are stored in .pkglist file.
- The other package list file contains the names of the packages that do NOT come from the os distro. They are stored in .otherpkgs.pkglist file.

The path to the package lists will be read from the osimage definition. Which osimage a node is using is specified by the provmethod attribute. To display this value for a node:

```
lsdef node1 -i provmethod
Object name: node
provmethod=<osimagenam>
```

You can display this details of this osimage by running the following command, supplying your osimage name:

```
lsdef -t osimage <osimagenam>
Object name: <osimagenam>
exlist=/opt/xcat/share/xcat/<inst_type>/<os>/<profile>.exlist
imagetype=linux
osarch=<arch>
osname=Linux
osvers=<os>
otherpkgdir=/install/post/otherpkgs/<os>/<arch>
otherpkglist=/install/custom/<inst_type>/<distro>/<profile>.otherpkgs.pkglist
pkgdir=/install/<os>/<arch>
pkglist=/opt/xcat/share/xcat/<inst_type>/<os>/<profile>.pkglist
postinstall=/opt/xcat/share/xcat/<inst_type>/<distro>/<profile>.<os>.<arch>.postinstall
profile=<profile>
provmethod=<profile>
rootingdir=/install/<inst_type>/<os>/<arch>/<profile>
synclist=/install/custom/<inst_type>/<profile>.synclist
```

You can set the pkglist and otherpkglist using the following command:

```
chdef -t osimage <osimagenam> pkglist=/opt/xcat/share/xcat/<inst_type>/<distro>/
-><profile>.pkglist\
                                otherpkglist=/install/custom/<inst_type>/
-><distro>/my.otherpkgs.pkglist
```



## Install Additional OS Packages for RHEL and SLES

### Install Additional Packages using OS Packages steps

For rpms from the OS distro, add the new rpm names (without the version number) in the .pkglist file. For example, file `/install/custom/<inst_type>/<os>/<profile>.pkglist` will look like this after adding perl-DBI:

```
bash
nfs-utils
openssl
dhcpcd
kernel-smp
openssh
procps
psmisc
resmgr
wget
rsync
timezone
perl-DBI
```

For the format of the .pkglist file, see [File Format for .ospkgs.pkglist File](#)

If you have newer updates to some of your operating system packages that you would like to apply to your OS image, you can place them in another directory, and add that directory to your osimage pkgdir attribute. For example, with the osimage defined above, if you have a new openssl package that you need to update for security fixes, you could place it in a directory, create repository data, and add that directory to your pkgdir:

```
mkdir -p /install/osupdates/<os>/<arch>
cd /install/osupdates/<os>/<arch>
cp <your new openssl rpm> .
createrepo .
chdef -t osimage <os>-<arch>-<inst_type>-<profile> pkgdir=/install/<os>/<arch>,/install/
↳ osupdates/<os>/<arch>
```

Note: If the objective node is not installed by xCAT, make sure the correct osimage pkgdir attribute so that you could get the correct repository data.

### File Format for .ospkgs.pkglist File

The .pkglist file is used to specify the rpm and the group/pattern names from os distro that will be installed on the nodes. It can contain the following types of entries:

- \* rpm name without version numbers
- \* group/pattern name marked **with** a '@' (**for** full install only)
- \* rpms to removed after the installation marked **with** a "-" (**for** full install only)

These are described in more details in the following sections.

## RPM Names

A simple .pkglist file just contains the name of the rpm file without the version numbers.

For example

```
openssl
xntp
rsync
glibc-devel.i686
```

## Include pkglist Files

The **#INCLUDE** statement is supported in the pkglist file.

You can group some rpms in a file and include that file in the pkglist file using **#INCLUDE:<file>#** format.

```
openssl
xntp
rsync
glibc-devel.1686
#INCLUDE:/install/post/custom/<distro>/myotherlist#
```

where **/install/post/custom/<distro>/myotherlist** is another package list file that follows the same format.

Note: the trailing “#” character at the end of the line. It is important to specify this character for correct pkglist parsing.

## Module/Group/Pattern Names

---

**Note:** On SLES pattern names are not supported for diskless deployment

---

On Linux, groups of rpms can be packaged together into one package. It can be a module or a group on RedHat, CentOS, Fedora and Scientific Linux. To get the list of available groups, run

- [RHEL]

```
yum group list
yum module list
```

- [SLES]

```
zypper se -t pattern
```

You can specify module/group/pattern names by adding a ‘@’ before the module/group/pattern names. For example:

```
@base
@Security Tools
@ruby:2.6
```

## Remove RPMs After Installing

You can specify that certain rpms to be removed after installing the new software. This is done by adding ‘-’ before the rpm names you want to remove. For example:

```
-ntp
-@ruby:2.6
```

## Install Additional Other Packages for RHEL and SLES

### Install Additional Other Packages Steps

If you have additional rpms (rpms **not** in the distro) that you also want installed, make a directory to hold them, create a list of the rpms you want installed, and add that information to the osimage definition:

- Create a directory to hold the additional rpms:

```
mkdir -p /install/post/otherpkgs/<distro>/<arch>
cd /install/post/otherpkgs/<distro>/<arch>
cp /myrpms/* .
createrepo .
```

- Create a file that lists the additional rpms that should be installed. For example, in `/install/custom/<inst_type>/<distro>/<profile>.otherpkgs.pkglist` put:

```
myrpm1
myrpm2
myrpm3
```

- Add both the directory and the file to the osimage definition:

```
chdef -t osimage mycomputeimage otherpkgdir=/install/post/otherpkgs/<os>/<arch>
->otherpkglist=/install/custom/<inst_type>/<os>/<profile>.otherpkgs.pkglist
```

If you add more rpms at a later time, you must run `createrepo` again. The `createrepo` command is in the `createrepo` rpm, which for RHEL is in the 1st DVD, but for SLES is in the SDK DVD.

If you have **multiple sets of rpms** that you want to **keep separate** to keep them organized, you can put them in separate sub-directories in the `otherpkgdir`. If you do this, you need to do the following extra things, in addition to the steps above:

- Run `createrepo` in each sub-directory
- In your `otherpkgs.pkglist`, list at least 1 file from each sub-directory. (During installation, xCAT will define a yum or zypper repository for each directory you reference in your `otherpkgs.pkglist`.) For example:

```
xcat/xcat-core/xCATsn
xcat/xcat-dep/<os>/<arch>/conserver-xcat
```

There are some examples of `otherpkgs.pkglist` in `/opt/xcat/share/xcat/<inst_type>/<distro>/<profile>.*.otherpkgs.pkglist` that show the format.

Note: the `otherpkgs` postbootscript should by default be associated with every node. Use `lsdef` to check:

```
lsdef node1 -i postbootscripts
```

If it is not, you need to add it. For example, add it for all of the nodes in the “**compute**” group:

```
chdef -p -t group compute postbootscripts=otherpkgs
```

For the format of the .Otherpkgs file, see *File Format for .otherpkgs.pkglist File*

## File Format for .otherpkgs.pkglist File

The otherpkgs.pkglist file can contain the following types of entries:

```
* rpm name without version numbers
* otherpkgs subdirectory plus rpm name
* blank lines
* comment lines starting with #
* #INCLUDE: <full file path># to include other pkglist files
* #NEW_INSTALL_LIST# to signify that the following rpms will be installed with a new rpm.
↳ install command (zypper, yum, or rpm as determined by the function using this file)
* #ENV:<variable list># to specify environment variable(s) for a sparate rpm install.
↳ command
* rpms to remove before installing marked with a "-"
* rpms to remove after installing marked with a "--"
```

These are described in more details in the following sections.

## RPM Names

A simple otherpkgs.pkglist file just contains the name of the rpm file without the version numbers.

For example, if you put the following three rpms under **/install/post/otherpkgs/<os>/<arch>/** directory,

```
rsct.core-2.5.3.1-09120.ppc.rpm
rsct.core.utils-2.5.3.1-09118.ppc.rpm
src-1.3.0.4-09118.ppc.rpm
```

The otherpkgs.pkglist file will be like this:

```
src
rsct.core
rsct.core.utils
```

## RPM Names with otherpkgs Subdirectories

If you create a subdirectory under **/install/post/otherpkgs/<os>/<arch>/**, say **rsct**, the otherpkgs.pkglist file will be like this:

```
rsct/src
rsct/rsct.core
rsct/rsct.core.utils
```

## Include Other pkglist Files

You can group some rpms in a file and include that file in the otherpkgs.pkglist file using **#INCLUDE:<file>#** format.

```
rsct/src
rsct/rsct.core
rsct/rsct.core.utils
#INCLUDE:/install/post/otherpkgs/myotherlist#
```

where **/install/post/otherpkgs/myotherlist** is another package list file that follows the same format.

Note the trailing “#” character at the end of the line. It is important to specify this character for correct pkglist parsing.

## Multiple Install Lists

You can specify that separate calls should be made to the rpm install program (**zypper, yum, rpm**) for groups of rpms by specifying the entry **#NEW\_INSTALL\_LIST#** on a line by itself as a separator in your pkglist file. All rpms listed up to this separator will be installed together. You can have as many separators as you wish in your pkglist file, and each sublist will be installed separately in the order they appear in the file.

For example:

```
compilers/vacpp.rte
compilers/vac.lib
compilers/vacpp.lib
compilers/vacpp.rte.lnk
#NEW_INSTALL_LIST#
pe/IBM_pe_license
```

## Environment Variable List

You can specify environment variable(s) for each rpm install call by entry “**#ENV:<variable list>#**”. The environment variables also apply to rpm(s) remove call if there is rpm(s) needed to be removed in the sublist.

For example:

```
#ENV:INUCLIENTS=1 INUBOSTYPE=1#
rsct/rsct.core
rsct/rsct.core.utils
rsct/src
```

Be same as,

```
#ENV:INUCLIENTS=1#
#ENV:INUBOSTYPE=1#
rsct/rsct.core
rsct/rsct.core.utils
rsct/src
```

## Remove RPMs Before Installing

You can also specify in this file that certain rpms to be removed before installing the new software. This is done by adding '-' before the rpm names you want to remove. For example:

```
rsct/src
rsct/rsct.core
rsct/rsct.core.utils
#INCLUDE:/install/post/otherpkgs/myotherlist#
-perl-doc
```

If you have #NEW\_INSTALL\_LIST# separators in your pkglist file, the rpms will be removed before the install of the sublist that the "-<rpmname>" appears in.

## Remove RPMs After Installing

You can also specify in this file that certain rpms to be removed after installing the new software. This is done by adding -- before the rpm names you want to remove. For example:

```
pe/IBM_pe_license
--ibm-java2-ppc64-jre
```

If you have #NEW\_INSTALL\_LIST# separators in your pkglist file, the rpms will be removed after the install of the sublist that the "--<rpmname>" appears in.

## Install Additional Other Packages with Ubuntu official mirror

The Ubuntu ISO used to install the compute nodes only include packages to run a minimal base operating system, it is likely that users will want to install additional Ubuntu packages from the internet Ubuntu repositories or local repositories, this section describes how to install additional Ubuntu packages.

### Compute nodes can access the internet

1. Specify the repository

Define the **otherpkgdir** attribute in osimage to use the internet repository directly.:

```
chdef -t osimage <osimage name> otherpkgdir="http://us.archive.ubuntu.com/ubuntu/ \
$(lsb_release -sc) main,http://us.archive.ubuntu.com/ubuntu/ $(lsb_release -sc)-
↪update main"
```

2. Define the otherpkglist file

create an **otherpkglist** file: /install/custom/install/ubuntu/compute.otherpkgs.pkglist, add the package names into this file, and modify the otherpkglist attribute in the osimage.

```
chdef -t osimage <osimage name> otherpkglist=/install/custom/install/ubuntu/compute.
↪otherpkgs.pkglist
```

3. Run updatenode <noderange> -S or updatenode <noderange> -P otherpkgs

Run updatenode -S to **install/update** the packages on the compute nodes

```
updatenode <noderange> -S
```

Run `updatenode -P otherpkgs` to **install/update** the packages on the compute nodes

```
updatenode <noderange> -P otherpkgs
```

## Compute nodes can not access the internet

If compute nodes cannot access the internet, there are two ways to install additional packages

- **Use local mirror**  
Please refer the Ubuntu document below for how to set up your own local Ubuntu mirror. <https://help.ubuntu.com/community/Rsyncmirror>
- **Use apt-proxy**  
Please refer the Ubuntu document below for how to setup a apt-proxy server. <https://help.ubuntu.com/community/AptProxy>
- **Setting up apt-get to use a http-proxy.**  
Please refer the Ubuntu document below for how to do set up it. [https://help.ubuntu.com/community/AptGet/Howto#Setting\\_up\\_apt-get\\_to\\_use\\_a\\_http-proxy](https://help.ubuntu.com/community/AptGet/Howto#Setting_up_apt-get_to_use_a_http-proxy)

## Use new kernel patch

This procedure assumes there are kernel RPM in /tmp, we take the osimage **rhels7.3-ppc64le-install-compute** as an example. The RPM names below are only examples, substitute your specific level and architecture.

- **[RHEL]**
  1. The RPM kernel package is usually named: kernel-<kernelver>.rpm. Append new kernel packages directory to osimage pkgdir

```
mkdir -p /install/kernels/<kernelver>
cp /tmp/kernel-*.rpm /install/kernels/<kernelver>
createrepo /install/kernels/<kernelver>/
chdef -t osimage rhels7.3-ppc64le-install-compute -p pkgdir=/install/kernels/
↳<kernelver>
```

2. Inject the drivers from the new kernel RPMs into the initrd

```
mkdef -t osdistroudate kernelupdate dirpath=/install/kernels/<kernelver>/
chdef -t osimage rhels7.3-ppc64le-install-compute osupdatename=kernelupdate
chdef -t osimage rhels7.3-ppc64le-install-compute netdrivers=updateonly
genitrd rhels7.3-ppc64le-install-compute --ignorekernelchk
nodeset <CN> osimage=rhels7.3-ppc64le-install-compute --noupdateinitrd
```

3. Boot CN from net normally.

## Customize network adapter

This section describes how to configure network adapters with persistent configuration using xCAT. The `confignetwork` postscript can be used to configure the network interfaces on the compute nodes to support Ethernet adapters, VLAN, BONDS, and BRIDGES.

### Configure Additional Network Interfaces - `confignetwork`

The `confignetwork` postscript can be used to configure the network interfaces on the compute nodes to support Ethernet adapters, VLAN, BONDS, and BRIDGES. `confignetwork` can be used in postscripts during OS provisioning, it can also be executed in `updatenode`. The way the `confignetwork` postscript decides what IP address to give the secondary adapter is by checking the `nics` table, in which the nic configuration information is stored. In order for the `confignetwork` postscript to run successfully, the following attributes must be configured for the node in the `nics` table:

- `nicips`
- `nictypes`
- `nicnetworks`

If configuring VLAN, BOND, or BRIDGES, `nicdevices` in `nics` table must be configured. VLAN, BOND or BRIDGES is only supported on RHEL.

- `nicdevices` - resolves the relationship among the physical network interface devices

The following scenarios are examples to configure Ethernet adapters/BOND/VLAN/Bridge.

1. Configure static install or application Ethernet adapters:
  - Scenario 1: Configure Ethernet Network Interface
2. Configure BOND [RHEL]:
  - Scenario 2: Configure Bond using two Ethernet Adapters
3. Configure VLAN [RHEL]:
  - Scenario 3: Configure VLAN Based on Ethernet Adapter
  - Scenario 4: Configure VLAN Based on Bond Adapters
4. Configure Bridge [RHEL]:
  - Scenario 5: Configure Bridge Based On Ethernet NIC
  - Scenario 6: Configure Bridge Based on Bond Adapters
  - Scenario 7: Configure Bridge Based on VLAN
  - Scenario 8: Configure Bridge Based on VLAN,VLAN use BOND adapter
5. Advanced topics:
  - Use Customized Scripts To Configure NIC
  - Use Extra Parameters In NIC Configuration File
  - Configure Aliases



## Configure routes

There are 2 ways to configure OS route in xCAT:

- **makeroutes**: command to add or delete routes on the management node or any given nodes.
- **setroute**: script to replace/add the routes to the node, it can be used in postscripts/postbootscripts.

**makeroutes** or **setroute** will modify OS temporary route, it also modifies persistent route in `/etc/sysconfig/static-routes` file.

Before using **makeroutes** or **setroute** to configure OS route, details of the routes data such as routename, subnet, net mask and gateway should be stored in routes table.

**Notes:** the gateway in the **networks** table assigns gateway from DHCP to compute node, so if use **makeroutes** or **setroute** to configure OS static route for compute node, make sure there is no gateway for the specific network in **networks** table.

## Configure routes table

1. Store default route data in routes table:

```
chdef -t route defaultroute net=default mask=255.0.0.0 gateway=10.0.0.101
```

2. Store additional route data in routes table:

```
chdef -t route 20net net=20.0.0.0 mask=255.0.0.0 gateway=0.0.0.0 ifname=eth1
```

3. Check data in routes table:

```
tabdump routes
#routename,net,mask,gateway,ifname,comments,disable
"30net","30.0.0.0","255.0.0.0","0.0.0.0","eth2",,
"20net","20.0.0.0","255.0.0.0","0.0.0.0","eth1",,
"defaultroute","default","255.0.0.0","10.0.0.101",,,
```

## Use makeroutes to configure OS route on xCAT management node

1. define the names of the routes to be setup on the management node in site table:

```
chdef -t site mnroutenames="defaultroute,20net"
lsdef -t site clustersite -i mnroutenames
Object name: clustersite
        mnroutenames=defaultroute,20net
```

2. add all routes from the mnroutenames to the OS route table for the management node:

```
makeroutes
```

3. add route 20net and 30net to the OS route table for the management node:

```
makeroutes -r 20net,30net
```

4. delete route 20net from the OS route table for the management node:

```
makeroutes -d -r 20net
```

### Use `makeroutes` to configure OS route for compute node

1. define the names of the routes to be setup on the compute node:

```
chdef -t cn1 routenames="defaultroute,20net"
```

2. add all routes from the `routenames` to the OS route table for the compute node:

```
makeroutes cn1
```

3. add route `20net` and `30net` to the OS route table for the compute node:

```
makeroutes cn1 -r 20net,30net
```

4. delete route `20net` from the OS route table for the compute node:

```
makeroutes cn1,cn2 -d -r 20net
```

### Use `setroute` to configure OS route for compute node

1. define the names of the routes to be setup on the compute node:

```
chdef -t cn1 routenames="defaultroute,20net"
```

2. If adding `setroute [replace | add]` into the node's `postscripts` list, `setroute` will be executed during OS deployment on compute node to replace/add routes from `routenames`:

```
chdef cn1 -p postscripts="setroute replace"
```

3. Or if the compute node is already running, use `updatenode` command to run `setroute [replace | add]` postscript:

```
updatenode cn1 -P "setroute replace"
```

### Check result

1. Use `route` command in xCAT management node to check OS route table.
2. Use `xdsh cn1 route` to check compute node OS route table.

## Initialize the Compute for Deployment

XCAT use **'nodeset'** command to associate a specific image to a node which will be installed with this image.

```
nodeset <nodename> osimage=<osimage>
```

There are more attributes of nodeset used for some specific purpose or specific machines, for example:

- **runimage:** If you would like to run a task after deployment, you can define that task with this attribute.
- **runcmd:** This instructs the node to boot to the xCAT nbfs environment and proceed to configure BMC for basic remote access. This causes the IP, netmask, gateway, username, and password to be programmed according to the configuration table.
- **shell:** This instructs the node to boot to the xCAT genesis environment, and present a shell prompt on console. The node will also be able to be sshed into and have utilities such as wget, tftp, scp, nfs, and cifs. It will have storage drivers available for many common systems.

Choose such additional attribute of nodeset according to your requirement, if want to get more information about nodeset, refer to nodeset's man page.

## Start the OS Deployment

Start the deployment involves two key operations. First specify the boot device of the next boot to be network, then reboot the node:

For **Power servers**, those two operations can be completed by one command **rnetboot**:

```
rnetboot <node>
```

For **x86\_64 servers**, those two operations need two independent commands.

1. set the next boot device to be from the "network"

```
rsetboot <node> net
```

2. Reboot the xSeries server: ::

```
rpower <node> reset
```

## Diskless Installation

### Select or Create an osimage Definition

Before creating an image on xCAT, the distro media should be prepared. That can be ISOs or DVDs.

XCAT uses **copycds** command to create an image which will be available to install nodes. **copycds** will copy all contents of Distribution DVDs/ISOs or Service Pack DVDs/ISOs to a destination directory, and create several relevant osimage definitions by default.

If using an ISO, copy it to (or NFS mount it on) the management node, and then run:

```
copycds <path>/<specific-distro>.iso
```

**Note:** While sle15 contains installer medium and packages medium, need copycds copy all contents of DVD1 of the installer medium and DVD1 of the packages medium, for example:

```
copycds SLE-15-Installer-DVD-ppc64le-GM-DVD1.iso SLE-15-Packages-ppc64le-GM-DVD1.iso
```

---

If using a DVD, put it in the DVD drive of the management node and run:

```
copycds /dev/<dvd-drive-name>
```

To see the list of osimages:

```
lsdef -t osimage
```

To see the attributes of a particular osimage:

```
lsdef -t osimage <osimage-name>
```

Initially, some attributes of osimage are assigned default values by xCAT - they all can work correctly because the files or templates invoked by those attributes are shipped with xCAT by default. If you need to customize those attributes, refer to the next section *Customize osimage*

Below is an example of osimage definitions created by copycds:

```
# lsdef -t osimage
rhels7.2-ppc64le-install-compute (osimage)
rhels7.2-ppc64le-install-service (osimage)
rhels7.2-ppc64le-netboot-compute (osimage)
rhels7.2-ppc64le-stateful-mgmtnode (osimage)
```

In these osimage definitions shown above

- **<os>-<arch>-install-compute** is the default osimage definition used for diskful installation
- **<os>-<arch>-netboot-compute** is the default osimage definition used for diskless installation
- **<os>-<arch>-install-service** is the default osimage definition used for service node deployment which shall be used in hierarchical environment

---

**Note:** Additional steps are needed for **ubuntu ppc64le** osimages:

---

For pre-16.04.02 version of Ubuntu for ppc64el, the `initrd.gz` shipped with the ISO does not support network booting. In order to install Ubuntu with xCAT, you need to follow the steps to complete the osimage definition.

- Download `mini.iso` from
  - [ubuntu 14.04.1]: <http://xcat.org/files/netboot/ubuntu14.04.1/ppc64el/mini.iso>
  - [ubuntu 14.04.2]: <http://xcat.org/files/netboot/ubuntu14.04.2/ppc64el/mini.iso>
  - [ubuntu 14.04.3]: <http://xcat.org/files/netboot/ubuntu14.04.3/ppc64el/mini.iso>
  - [ubuntu 14.04.4]: <http://xcat.org/files/netboot/ubuntu14.04.4/ppc64el/mini.iso>
  - [ubuntu 16.04]: <http://xcat.org/files/netboot/ubuntu16.04/ppc64el/mini.iso>
  - [ubuntu 16.04.1]: <http://xcat.org/files/netboot/ubuntu16.04.1/ppc64el/mini.iso>
- Mount `mini.iso`

```
mkdir /tmp/iso
mount -o loop mini.iso /tmp/iso
```

- Copy the netboot initrd.gz to osimage

```
mkdir -p /install/<ubuntu-version>/ppc64el/install/netboot
cp /tmp/iso/install/initrd.gz /install/<ubuntu-version>/ppc64el/install/netboot
```

### [Tips 1]

If this is the same distro version as what your management node uses, create a `.repo` file in `/etc/yum.repos.d` with contents similar to:

```
[local-<os>-<arch>]
name=xCAT local <os> <version>
baseurl=file:/install/<os>/<arch>
enabled=1
gpgcheck=0
```

This way, if you need to install some additional RPMs into your MN later, you can simply install them with `yum`. Or if you are installing a software on your MN that depends some RPMs from this distro, those RPMs will be found and installed automatically.

### [Tips 2]

You can create/modify an osimage definition easily with any existing osimage definition, the command is

```
mkdef -t osimage -o <new osimage> --template <existing osimage> [<attribute>=<value>, ...
↪]
```

Except the specified attributes `<attribute>`, the attributes of `<new osimage>` will inherit the values of template osimage `<existing osimage>`.

As an example, the following command creates a new osimage `myosimage.rh7.compute.netboot` based on the existing osimage `rhels7.4-ppc64le-netboot-compute` with some customized attributes

```
mkdef -t osimage -o myosimage.rh7.compute.netboot --template rhels7.4-ppc64le-netboot-
↪compute synclists=/tmp/synclist otherpkgdir=/install/custom/osimage/myosimage.rh7.
↪compute.netboot/3rdpkgs/ otherpkglist=/install/custom/osimage/myosimage.rh7.compute.
↪netboot/3rd.pkglist
```

## Customize osimage (Optional)

Optional means all the subitems in this page are not necessary to finish an OS deployment. If you are new to xCAT, you can just jump to *Initialize the Compute for Deployment*.

## Load Additional Drivers

### Overview

During the installing or netbooting of a node, the drivers in the initrd will be used to drive the devices like network cards and IO devices to perform the installation/netbooting tasks. But sometimes the drivers for the new devices were not included in the default initrd shipped by Red Hat or Suse. A solution is to inject the new drivers into the initrd to drive the new device during the installation/netbooting process.

Generally there are two approaches to inject the new drivers: **Driver Update Disk** and **Driver RPM package**.

A “**Driver Update Disk**” is media which contains the drivers, firmware and related configuration files for certain devices. The driver update disk is always supplied by the vendor of the device. One driver update disk can contain multiple drivers for different OS releases and different hardware architectures. Red Hat and Suse have different driver update disk formats.

The ‘**Driver RPM Package**’ is the rpm package which includes the drivers and firmware for the specific devices. The Driver RPM is the rpm package which is shipped by the Vendor of the device for a new device or a new kernel version.

xCAT supports both. But for ‘**Driver RPM Package**’ is only supported in xCAT 2.8 and later.

No matter which approach chosen, there are two steps to make new drivers work. one is locate the new driver’s path, another is inject the new drivers into the initrd.

### Locate the New Drivers

#### For Driver Update Disk

There are two approaches for xCAT to find the driver disk (pick one):

1. Specify the location of the driver disk in the osimage object (*This is ONLY supported in xCAT 2.8 and later*)

The value for the ‘driverupdatesrc’ attribute is a comma separated driver disk list. The tag ‘dud’ must be specified before the full path of ‘driver update disk’ to specify the type of the file:

```
chdef -t osimage <osimagename> driverupdatesrc=dud:<full path of driver disk>
```

1. Put the driver update disk in the directory <installroot>/driverdisk/<os>/<arch> (example: /install/driverdisk/sles11.1/x86\_64).

During the running of the `genimage`, `geninitrd`, or `nodeset` commands, xCAT will look for driver update disks in the directory <installroot>/driverdisk/<os>/<arch>.

#### For Driver RPM Packages

The Driver RPM packages must be specified in the osimage object.

Three attributes of osimage object can be used to specify the Driver RPM location and Driver names. If you want to load new drivers in the initrd, the ‘**netdrivers**’ attribute must be set. And one or both of the ‘**driverupdatesrc**’ and ‘**osupdatename**’ attributes must be set. If both of ‘driverupdatesrc’ and ‘osupdatename’ are set, the drivers in the ‘driverupdatesrc’ have higher priority.

- netdrivers - comma separated driver names that need to be injected into the initrd. The postfix ‘.ko’ can be ignored.

The ‘netdrivers’ attribute must be set to specify the new driver list. If you want to load all the drivers from the driver rpms, use the keyword `allupdate`. Another keyword for the `netdrivers` attribute is `updateonly`, which means only the drivers located in the original `initrd` will be added to the newly built `initrd` from the driver rpms. This is useful to reduce the size of the new built `initrd` when the distro is updated, since there are many more drivers in the new kernel rpm than in the original `initrd`. Examples:

```
chdef -t osimage <osimagename> netdrivers=megaraid_sas.ko,igb.ko
chdef -t osimage <osimagename> netdrivers=allupdate
chdef -t osimage <osimagename> netdrivers=updateonly,igb.ko,new.ko
```

- `driverupdatesrc` - comma separated driver rpm packages (full path should be specified)

A tag named ‘rpm’ can be specified before the full path of the rpm to specify the file type. The tag is optional since the default format is ‘rpm’ if no tag is specified. Example:

```
chdef -t osimage <osimagename> driverupdatesrc=rpm:<full path of driver disk1>,rpm:<full_
↳path of driver disk2>
```

- `osupdatename` - comma separated ‘osdistroudate’ objects. Each ‘osdistroudate’ object specifies a Linux distro update.

When `geninitrd` is run, `kernel-*.rpm` will be searched in the `osdistroudate.dirpath` to get all the rpm packages and then those rpms will be searched for drivers. Example:

```
mkdef -t osdistroudate update1 dirpath=/install/<os>/<arch>
chdef -t osimage <osimagename> osupdatename=update1
```

If ‘osupdatename’ is specified, the kernel shipped with the ‘osupdatename’ will be used to load the newly built `initrd`, then only the drivers matching the new kernel will be kept in the newly built `initrd`. If trying to use the ‘osupdatename’, the ‘allupdate’ or ‘updateonly’ should be added in the ‘netdrivers’ attribute, or all the necessary driver names for the new kernel need to be added in the ‘netdrivers’ attribute. Otherwise the new drivers for the new kernel will be missed in newly built `initrd`. ..

## Inject the Drivers into the initrd

### For Driver Update Disk

- If specifying the driver disk location in the `osimage`

Run the following command:

```
genimage <osimagename>
```

- If putting the driver disk in `<installroot>/driverdisk/<os>/<arch>`:

Running ‘genimage’ in anyway will load the driver disk ..

## For Driver RPM Packages

Run the following command:

```
genimage <osimagename> [--ignorekernelchk]
```

The option ‘--ignorekernelchk’ is used to skip the kernel version checking when injecting drivers from osimage.driverupdatesrc. To use this flag, you should make sure the drivers in the driver rpms are usable for the target kernel. ..

## Notes

- If the drivers from the driver disk or driver rpm are not already part of the installed or booted system, it’s necessary to add the rpm packages for the drivers to the .pkglist or .otherpkglist of the osimage object to install them in the system.
- If a driver rpm needs to be loaded, the osimage object must be used for the ‘nodeset’ and ‘genimage’ command, instead of the older style profile approach.
- Both a Driver disk and a Driver rpm can be loaded in one ‘nodeset’ or ‘genimage’ invocation.

## Prescripts and Postscripts

### Using Postscript

### Postscript Execution Order Summary

Diskless Stage	Scripts	Execute Order
Install/Create	postinstall	genimage, after packages are installed
Boot/Reboot	postscripts	1 postscripts.xcatdefaults
		2 osimage
		3 node
	postbootscripts	4 postscripts.xcatdefaults
		5 osimage
		6 node

xCAT automatically runs a few postscripts and postbootscripts that are delivered with xCAT to set up the nodes. You can also add your own scripts to further customize the nodes.



## Types of scripts

There are two types of scripts in the postscripts table ( postscripts and postbootscripts). The types are based on when in the install process they will be executed. Run the following for more information:

```
man postscripts
```

- **postscripts attribute** - List of scripts that should be run on this node after diskful installation or diskless boot.
  - [RHEL]
 

Postscripts will be run before the reboot.
  - [SLES]
 

Postscripts will be run after the reboot but before the `init.d` process. For Linux diskless deployment, the postscripts will be run at the `init.d` time, and xCAT will automatically add the list of postscripts from the postbootscripts attribute to run after postscripts list.
- **postbootscripts attribute** - list of postbootscripts that should be run on this Linux node at the `init.d` time after diskful installation reboot or diskless boot
- **xCAT**, by default, for diskful installs only runs the postbootscripts on the install and not on reboot. In xCAT a `site` table attribute `runbootscripts` is available to change this default behavior. If set to `yes` or `y` or `1`, then the postbootscripts will be run on install and on reboot. To avoid reinstallation, run `updatenode <nodes> -P setuppostbootscripts` command to update the value of `runbootscripts` attribute on the compute or service nodes.

**Note:** xCAT automatically adds the postscripts from the `xcatdefaults.postscripts` attribute of the table to run first on the nodes after install or diskless boot.

## Adding your own postscripts

To add your own script, place it in `/install/postscripts` on the management node. Make sure it is executable and world readable. Then add it to the `postscripts` table for the group of nodes you want it to be run on (or the `all` group if you want it run on all nodes).

To check what scripts will be run on your node during installation:

```
lsdef node1 | grep scripts
postbootscripts=otherpkgs
postscripts=syslog,remoteshell,syncfiles
```

You can pass parameters to the postscripts. For example:

```
script1 p1 p2,script2,....
```

`p1 p2` are the parameters to `script1`.

Postscripts could be placed in the subdirectories in `/install/postscripts` on management node, and specify `subdir/postscriptname` in the `postscripts` table to run the postscripts in the subdirectories. This feature could be used to categorize the postscripts for different purposes. For example:

```
mkdir -p /install/postscripts/subdir1
mkdir -p /install/postscripts/subdir2
```

(continues on next page)

(continued from previous page)

```
cp postscript1 /install/postscripts/subdir1/
cp postscript2 /install/postscripts/subdir2/
chdef node1 -p postscripts=subdir1/postscript1,subdir2/postscript2
updatenode node1 -P
```

If some of your postscripts will affect the network communication between the management node and compute node, like restarting network or configuring bond, the postscripts execution might not be able to be finished successfully because of the network connection problems. Even if we put this postscript be the last postscript in the list, xCAT still may not be able to update the node status to be booted. The recommendation is to use the Linux `at` mechanism to schedule this network-killing postscript to be run at a later time. For example:

The user needs to add a postscript to customize the nics bonding setup, the nics bonding setup will break the network between the management node and compute node. User could use `at` to run this nic bonding postscripts after all the postscripts processes have been finished.

Write a script, `/install/postscripts/nicbondscript`, the `nicbondscript` simply calls the `confignicsbond` using `at`:

```
[root@xcatmn ~]#cat /install/postscripts/nicbondscript
#!/bin/bash
at -f ./confignicsbond now + 1 minute
[root@xcatmn ~]#
```

Then

```
chdef <nodename> -p postbootscripts=nicbondscript
```

## Recommended Postscript design

- Postscripts that you want to run anywhere on Linux, should be written in shell. This should be available on all OS's. If only on the service nodes, you can use Perl .
- Postscripts should log errors using the following command (`local4` is the default xCAT syslog class). `logger -t xCAT -p local4.info "your info message"`.
- Postscripts should have good and error exit codes (i.e 0 and 1).
- Postscripts should be well documented. At the top of the script, the first few lines should describe the function and inputs and output. You should have comments throughout the script. This is especially important if using `regex`.

## PostScript/PostbootScript execution

When your script is executed on the node, all the attributes in the `site` table are exported as variables for your scripts to use. You can add extra attributes for yourself. See the sample `mypostscript` file below.

To run the postscripts, a script is built, so the above exported variables can be input. You can usually find that script in `/xcatpost` on the node and in the Linux case it is call `mypostscript`. A good way to debug problems is to go to the node and just run `mypostscript` and see errors. You can also check the `syslog` on the Management Node for errors.

When writing you postscripts, it is good to follow the example of the current postscripts and write errors to `syslog` and in shell. See Suggestions for writing scripts.

All attributes in the `site` table are exported and available to the postscript/postbootscript during execution. See the `mypostscript` file, which is generated and executed on the nodes to run the postscripts.

Example of mypostscript

```
#subroutine used to run postscripts
run_ps () {
logdir="/var/log/xcat"
mkdir -p $logdir
logfile="/var/log/xcat/xcat.log"
if [_-f_$1_]; then
    echo "Running postscript: $" | tee -a $logfile
    ./$_ 2>&1 | tee -a $logfile
else
    echo "Postscript $1 does NOT exist." | tee -a $logfile
fi
}
# subroutine end
AUDITSKIPCMDS='tabdump,nodeids'
export AUDITSKIPCMDS
TEST='test'
export TEST
NAMESERVERS='7.114.8.1'
export NAMESERVERS
NTPSERVERS='7.113.47.250'
export NTPSERVERS
INSTALLLOC='/install'
export INSTALLLOC
DEFSERIALPORT='0'
export DEFSERIALPORT
DEFSERIALSPEED='19200'
export DEFSERIALSPEED
DHCPINTERFACES="'xcat20RRmn|eth0;rra000-m|eth1'"
export DHCPINTERFACES
FORWARDERS='7.113.8.1,7.114.8.2'
export FORWARDERS
NAMESERVER='7.113.8.1,7.114.47.250'
export NAMESERVER
DB='postg'
export DB
BLADEMAXP='64'
export BLADEMAXP
FSPTIMEOUT='0'
export FSPTIMEOUT
INSTALLDIR='/install'
export INSTALLDIR
IPMIMAXP='64'
export IPMIMAXP
IPMIRETRIES='3'
export IPMIRETRIES
IPMITIMEOUT='2'
export IPMITIMEOUT
CONSOLEONDEMAND='no'
export CONSOLEONDEMAND
SITEMASTER=7.113.47.250
export SITEMASTER
```

(continues on next page)

(continued from previous page)

```

MASTER=7.113.47.250
export MASTER
MAXSSH='8'
export MAXSSH
PPCMAXP='64'
export PPCMAXP
PPCRETRY='3'
export PPCRETRY
PPCTIMEOUT='0'
export PPCTIMEOUT
SHAREDFTTP='1'
export SHAREDFTTP
SNSYNCFILEDIR='/var/xcat/syncfiles'
export SNSYNCFILEDIR
TFTPDIR='/tftpboot'
export TFTPDIR
XCATDPORT='3001'
export XCATDPORT
XCATIPORT='3002'
export XCATIPORT
XCATCONFDIR='/etc/xcat'
export XCATCONFDIR
TIMEZONE='America/New_York'
export TIMEZONE
USENMAPFROMMN='no'
export USENMAPFROMMN
DOMAIN='cluster.net'
export DOMAIN
USESSHONAIIX='no'
export USESSHONAIIX
NODE=rra000-m
export NODE
NFSSERVER=7.113.47.250
export NFSSERVER
INSTALLNIC=eth0
export INSTALLNIC
PRIMARYNIC=eth1
OSVER=fedora9
export OSVER
ARCH=x86_64
export ARCH
PROFILE=service
export PROFILE
PATH=`dirname $0`: $PATH
export PATH
NODESETSTATE='netboot'
export NODESETSTATE
UPDATENODE=1
export UPDATENODE
NTYPE=service
export NTYPE
MACADDRESS='00:14:5E:5B:51:FA'

```

(continues on next page)

(continued from previous page)

```
export MACADDRESS
MONSERVER=7.113.47.250
export MONSERVER
MONMASTER=7.113.47.250
export MONMASTER
OSPKGS=bash,openssl,dhclient,kernel,openssh-server,openssh-clients,busybox-anaconda,vim-
minimal,rpm,bind,bind-utils,ksh,nfs-utils,dhcp,bzip2,rootfiles,vixie-cron,wget,vsftpd,
↪ntp,rsync
OTHERPKGS1=xCATsn,xCAT-rmc,rsct/rsct.core,rsct/rsct.core.utils,rsct/src,yaboot-xcat
export OTHERPKGS1
OTHERPKGS_INDEX=1
export OTHERPKGS_INDEX
export NOSYNCFILES
# postscripts-start-here\n
run_ps ospkgs
run_ps script1 p1 p2
run_ps script2
# postscripts-end-here\n
```

The mypostscript file is generated according to the mypostscript.tmpl file.

## Using the mypostscript template

### Using the mypostscript template

xCAT provides a way for the admin to customize the information that will be provided to the postscripts/postbootscripts when they run on the node. This is done by editing the mypostscript.tmpl file. The attributes that are provided in the shipped mypostscript.tmpl file should not be removed. They are needed by the default xCAT postscripts.

The mypostscript.tmpl, is shipped in the /opt/xcat/share/xcat/mypostscript directory.

If the admin customizes the mypostscript.tmpl, they should copy the mypostscript.tmpl to /install/postscripts/mypostscript.tmpl, and then edit it. The mypostscript for each node will be named mypostscript.<nodename>. The generated mypostscript.<nodename>. will be put in the /tftpboot/mypostscripts directory.

### site table precreatemypostscripts attribute

If the site table precreatemypostscripts attribute is set to 1 or yes, it will instruct xCAT at nodeset and updatenode time to query the db once for all of the nodes passed into the command and create the mypostscript file for each node and put them in a directory in \$TFTPDIR (for example /tftpboot). The created mypostscript.<nodename>. file in the /tftpboot/mypostscripts directory will not be regenerated unless another nodeset or updatenode command is run to that node. This should be used when the system definition has stabilized. It saves time on the updatenode or reboot by not regenerating the mypostscript file.

If the precreatemyposcripts attribute is yes, and a database change is made or xCAT code is upgraded, then you should run a new nodeset or updatenode to regenerate the /tftpboot/mypostscript/mypostscript.<nodename> file to pick up the latest database setting. The default for precreatemypostscripts is no/0.

When you run nodeset or updatenode, it will search the /install/postscripts/mypostscript.tmpl first. If the /install/postscripts/mypostscript.tmpl exists, it will use that template to generate the mypostscript for each node. Otherwise, it will use /opt/xcat/share/xcat/mypostscript/mypostscript.tmpl.

## Content of the template for mypostscript

**Note:** The attributes that are defined in the shipped mypostscript.tpl file should not be removed. The xCAT default postscripts rely on that information to run successfully.

---

The following will explain the entries in the mypostscript.tpl file.

The SITE\_TABLE\_ALL\_ATTRIBS\_EXPORT line in the file directs the code to export all attributes defined in the site table. The attributes are not always defined exactly as in the site table to avoid conflict with other table attributes of the same name. For example, the site table master attribute is named SITEMASTER in the generated mypostscript file.

```
#SITE_TABLE_ALL_ATTRIBS_EXPORT#
```

The following line exports ENABLESSHBEWEENNODES by running the internal xCAT routine (enablenesshbetweennodes).

```
ENABLESSHBEWEENNODES=#Subroutine:xCAT::Template::enablenesshbetweennodes:$NODE#  
export ENABLESSHBEWEENNODES
```

tabdump(<TABLENAME>) is used to get all the information in the <TABLENAME> table

```
tabdump(networks)
```

These line export the node name based on its definition in the database.

```
NODE=$NODE  
export NODE
```

These lines get a comma separated list of the groups to which the node belongs.

```
GROUP=#TABLE:nodelist:$NODE:groups#  
export GROUP
```

These lines reads the nodesres table, the given attributes (nfssserver, installnic, primarynic, xcatmaster, routenames) for the node (\$NODE), and exports it.

```
NFSSERVER=#TABLE:noderes:$NODE:nfssserver#  
export NFSSERVER  
INSTALLNIC=#TABLE:noderes:$NODE:installnic#  
export INSTALLNIC  
PRIMARYNIC=#TABLE:noderes:$NODE:primarynic#  
export PRIMARYNIC  
MASTER=#TABLE:noderes:$NODE:xcatmaster#  
export MASTER  
NODEROUTENAMES=#TABLE:noderes:$NODE:routenames#  
export NODEROUTENAMES
```

The following entry exports multiple variables from the routes table. Not always set.

```
#ROUTES_VARS_EXPORT#
```

The following lines export nodetype table attributes.

```
OSVER=#TABLE:nodetype:$NODE:os#
export OSVER
ARCH=#TABLE:nodetype:$NODE:arch#
export ARCH
PROFILE=#TABLE:nodetype:$NODE:profile#
export PROFILE
PROVMETHOD=#TABLE:nodetype:$NODE:provmethod#
export PROVMETHOD
```

The following adds the current directory to the path for the postscripts.

```
PATH=`dirname $0`: $PATH
export PATH
```

The following sets the NODESETSTATE by running the internal xCAT getnodesetstate script.

```
NODESETSTATE=#Subroutine:xCAT::Postage::getnodesetstate:$NODE#
export NODESETSTATE
```

The following says the postscripts are not being run as a result of updatenode. (This is changed =1, when updatenode runs).

```
UPDATENODE=0
export UPDATENODE
```

The following sets the NTYPE to compute, service or MN.

```
NTYPE=$NTYPE
export NTYPE
```

The following sets the mac address.

```
MACADDRESS=#TABLE:mac:$NODE:mac#
export MACADDRESS
```

If vlan is setup, then the #VLAN\_VARS\_EXPORT# line will provide the following exports:

```
VMNODE='YES'
export VMNODE
VLANID=vlan1...
export VLANID
VLANHOSTNAME=. .
. .
#VLAN_VARS_EXPORT#
```

If monitoring is setup, then the #MONITORING\_VARS\_EXPORT# line will provide:

```
MONSERVER=11.10.34.108
export MONSERVER
MONMASTER=11.10.34.108
export MONMASTER
#MONITORING_VARS_EXPORT#
```

The #OSIMAGE\_VARS\_EXPORT# line will provide, for example:

```
OSPKGDIR=/install/<os>/<arch>
export OSPKGDIR
OSPKGS='bash,nfs-utils,openssl,dhclient,kernel,openssh-server,openssh-clients,busybox,
↪ wget,rsyslog,dash,vim-minimal,ntp,rsyslog,rpm,rsync,
  ppc64-utils,iputils,dracut,dracut-network,e2fsprogs,bc,lsnbd,irqbalance,procps,yum'
export OSPKGS

#OSIMAGE_VARS_EXPORT#
```

THE #NETWORK\_FOR\_DISKLESS\_EXPORT# line will provide diskless networks information, if defined.

```
NETMASK=255.255.255.0
export NETMASK
GATEWAY=8.112.34.108
export GATEWAY
. .
#NETWORK_FOR_DISKLESS_EXPORT#
```

---

**Note:** The #INCLUDE\_POSTSCRIPTS\_LIST# and the #INCLUDE\_POSTBOOTSCRIPTS\_LIST# sections in /tftpboot/mypostscript(mypostbootscripts) on the Management Node will contain all the postscripts and postbootscripts defined for the node. When running an updatenode command for only some of the scripts, you will see in the /xcatpost/mypostscript file on the node, the list has been redefined during the execution of updatenode to only run the requested scripts. For example, if you run updatenode <nodename> -P syslog.

---

The #INCLUDE\_POSTSCRIPTS\_LIST# flag provides a list of postscripts defined for this \$NODE.

```
#INCLUDE_POSTSCRIPTS_LIST#
```

For example, you will see in the generated file the following stanzas:

```
# postscripts-start-here
# defaults-postscripts-start-here
syslog
remoteshell
# defaults-postscripts-end-here
# node-postscripts-start-here
syncfiles
# node-postscripts-end-here
```

The #INCLUDE\_POSTBOOTSCRIPTS\_LIST# provides a list of postbootscripts defined for this \$NODE.

```
#INCLUDE_POSTBOOTSCRIPTS_LIST#
```

For example, you will see in the generated file the following stanzas:

```
# postbootscripts-start-here
# defaults-postbootscripts-start-here
otherpkgs
# defaults-postbootscripts-end-here
# node-postbootscripts-end-here
# postbootscripts-end-here
```



## Kinds of variables in the template

**Type 1:** For the simple variable, the syntax is as follows. The `mypostscript.tpl` has several examples of this. `$NODE` is filled in by the code. `UPDATENODE` is changed to 1, when the postscripts are run by `updatenode`. `$NTYPE` is filled in as either `compute`, `service` or `MN`.

```
NODE=$NODE
export NODE
UPDATENODE=0
export UPDATENODE
NTYPE=$NTYPE
export NTYPE
```

**Type 2:** This is the syntax to get the value of one attribute from the `<tablename>` and its key is `$NODE`. It does not support tables with two keys. Some of the tables with two keys are `litefile`, `prodkey`, `deps`, `monsetting`, `mpa`, `networks`. It does not support tables with keys other than `$NODE`. Some of the tables that do not use `$NODE` as the key, are `passwd`, `rack`, `token`

```
VARNAME=#TABLE:tablename:$NODE:attribute#
```

For example, to get the new `updatestatus` attribute from the `nodelist` table:

```
UPDATESTATUS=#TABLE:nodelist:$NODE:updatestatus#
export UPDATESTATUS
```

**Type 3:** The syntax is as follows:

```
VARNAME=#Subroutine:modulename::subroutinename:$NODE#
or
VARNAME=#Subroutine:modulename::subroutinename#
```

Examples in the `mypostscript.tpl` are the following:

```
NODESETSTATE=#Subroutine:xCAT::Postage::getnodesetstate:$NODE#
export NODESETSTATE
ENABLESSHBEETWEENNODES=#Subroutine:xCAT::Template::enablesshbetweennodes:$NODE#
export ENABLESSHBEETWEENNODES
```

**Note:** Type 3 is not an open interface to add extensions to the template.

**Type 4:** The syntax is `#FLAG#`. When parsing the template, the code generates all entries defined by `#FLAG#`, if they are defined in the database. For example: To export all values of all attributes from the `site` table. The tag is

```
#SITE_TABLE_ALL_ATTRIBS_EXPORT#
```

For the `#SITE_TABLE_ALL_ATTRIBS_EXPORT#` flag, the related subroutine will get the attributes' values and deal with the special case. such as : the `site.master` should be exported as `"SITEMASTER"`. And if the `noderes.xcatmaster` exists, the `noderes.xcatmaster` should be exported as `"MASTER"`, otherwise, we also should export `site.master` as the `"MASTER"`.

Other examples are:

```
#VLAN_VARS_EXPORT# - gets all vlan related items
#MONITORING_VARS_EXPORT# - gets all monitoring configuration and setup data
#OSIMAGE_VARS_EXPORT# - get osimage related variables, such as ospkgdir, ospkgs ...
#NETWORK_FOR_DISKLESS_EXPORT# - gets diskless network information
#INCLUDE_POSTSCRIPTS_LIST# - includes the list of all postscripts for the node
#INCLUDE_POSTBOOTSCRIPTS_LIST# - includes the list of all postbootscripts for the node
```

---

**Note:** Type4 is not an open interface to add extensions to the template.

---

**Type 5:** Get all the data from the specified table. The <TABLENAME> should not be a node table, like `nodelist`. This should be handles with TYPE 2 syntax to get specific attributes for the \$NODE. `tabdump` would result in too much data for a `nodetype` table. Also the `auditlog`, `eventlog` should not be in `tabdump` for the same reason. `site` table should not be specified, it is already provided with the `#SITE_TABLE_ALL_ATTRIBS_EXPORT#` flag. It can be used to get the data from the two key tables (like `switch`). The syntax is:

```
tabdump(<TABLENAME>)
```

## Edit mypostscript.tmpl

### Add new attributes into mypostscript.tmpl

When you add new attributes into the template, you should edit the `/install/postscripts/mypostscript.tmpl` which you created by copying `/opt/xcat/share/xcat/mypostscript/mypostscript.tmpl`. Make all additions before the `# postscripts-start-here` section. xCAT will first look in `/install/postscripts/mypostscript.tmpl` for a file and then, if not found, will use the one in `/opt/xcat/share/xcat/mypostscript/mypostscript.tmpl`.

For example:

```
UPDATESTATUS=#TABLE:nodelist:$NODE:updatestatus#
export UPDATESTATUS

...
# postscripts-start-here
#INCLUDE_POSTSCRIPTS_LIST#
## The following flag postscripts-end-here must not be deleted.
# postscripts-end-here
```

---

**Note:** If you have a hierarchical cluster, you must copy your new `mypostscript.tmpl` to `/install/postscripts/mypostscript.tmpl` on the service nodes, unless `/install/postscripts` directory is mounted from the MN to the service node.

---

### Remove attribute from mypostscript.tmpl

If you want to remove an attribute that you have added, you should remove all the related lines or comment them out with `##`. For example, comment out the added lines.

```
##UPDATESTATUS=#TABLE:nodelist:$NODE:updatestatus#
##export UPDATESTATUS
```

## Test the new template

There are two quick ways to test the template.

1. If the node is up

```
updatenode <nodename> -P syslog
```

Check your generated mypostscript on the compute node:

```
vi /xcatpost/mypostscript
```

2. Set the precreatemypostscripts option

```
chdef -t site -o clustersite precreatemypostscripts=1
```

Then run

```
nodeset <nodename> ....
```

Check your generated mypostscript

```
vi /tftpboot/mypostscripts/mypostscript.<nodename>
```

## Sample /xcatpost/mypostscript

This is an example of the generated postscript for a servicenode install. It is found in /xcatpost/mypostscript on the node.

```
# global value to store the running status of the postbootscripts,the value
#is non-zero if one postbootscript failed
return_value=0
# subroutine used to run postscripts
run_ps () {
    local ret_local=0
    logdir="/var/log/xcat"
    mkdir -p $logdir
    logfile="/var/log/xcat/xcat.log"
    if [ -f $1 ]; then
        echo "`date` Running postscript: $@" | tee -a $logfile
        #./$@ 2>&1 1> /tmp/tmp4xcatlog
        #cat /tmp/tmp4xcatlog | tee -a $logfile
        ./$@ 2>&1 | tee -a $logfile
        ret_local=${PIPESTATUS[0]}
        if [ "$ret_local" -ne "0" ]; then
            return_value=$ret_local
        fi
        echo "Postscript: $@ exited with code $ret_local"
    else
        echo "`date` Postscript $1 does NOT exist." | tee -a $logfile
        return_value=-1
    fi
    return 0
}
```

(continues on next page)

(continued from previous page)

```

}
# subroutine end
SHAREDFTFTP='1'
export SHAREDFTFTP
TFTPDIR='/tftpboot'
export TFTPDIR
CONSOLEONDEMAND='yes'
export CONSOLEONDEMAND
PPCTIMEOUT='300'
export PPCTIMEOUT
VSFTP='y'
export VSFTP
DOMAIN='cluster.com'
export DOMAIN
XCATIPORT='3002'
export XCATIPORT
DHCPINTERFACES='"xcatmn2|eth1;service|eth1"'
export DHCPINTERFACES
MAXSSH='10'
export MAXSSH
SITEMASTER=10.2.0.100
export SITEMASTER
TIMEZONE='America/New_York'
export TIMEZONE
INSTALLDIR='/install'
export INSTALLDIR
NTPSERVERS='xcatmn2'
export NTPSERVERS
EA_PRIMARY_HMC='c76v2hmc01'
export EA_PRIMARY_HMC
NAMESERVERS='10.2.0.100'
export NAMESERVERS
SNSYNCFILEDIR='/var/xcat/syncfiles'
export SNSYNCFILEDIR
DISJOINTDHCP='0'
export DISJOINTDHCP
FORWARDERS='8.112.8.1,8.112.8.2'
export FORWARDERS
VLANNETS='|(\d+)|10.10.($1+0).0|'
export VLANNETS
XCATDPORT='3001'
export XCATDPORT
USENMAPFROMMN='no'
export USENMAPFROMMN
DNSHANDLER='ddns'
export DNSHANDLER
ROUTENAMES='r1,r2'
export ROUTENAMES
INSTALLLOC='/install'
export INSTALLLOC
ENABLESSHBEETWEENNODES=YES
export ENABLESSHBEETWEENNODES

```

(continues on next page)

(continued from previous page)

```

NETWORKS_LINES=4
export NETWORKS_LINES
NETWORKS_LINE1='netname=public_net|net=8.112.154.64|mask=255.255.255.
↪192|mgtifname=eth0|gateway=8.112.154.126|dhcpserver=|tftpserver=8.112.154.
↪69|nameservers=8.112.8.
↪1|ntpserver=|logserver=|dynamicrange=|staticrange=|staticrangeincrement=|nodehostname=|ddnsd
↪'
export NETWORKS_LINE2
NETWORKS_LINE3='netname=sn21_net|net=10.2.1.0|mask=255.255.255.
↪0|mgtifname=eth1|gateway=<xcatmaster>|dhcpserver=|tftpserver=|nameservers=10.2.1.
↪100,10.2.1.
↪101|ntpserver=|logserver=|dynamicrange=|staticrange=|staticrangeincrement=|nodehostname=|ddnsd
↪'
export NETWORKS_LINE3
NETWORKS_LINE4='netname=sn22_net|net=10.2.2.0|mask=255.255.255.
↪0|mgtifname=eth1|gateway=10.2.2.100|dhcpserver=10.2.2.100|tftpserver=10.2.2.
↪100|nameservers=10.2.2.100|ntpserver=|logserver=|dynamicrange=10.2.2.120-10.2.2.
↪250|staticrange=|staticrangeincrement=|nodehostname=|ddnsdomain=|vlanid=|domain=|mtu=|disablen
↪'
export NETWORKS_LINE4
NODE=xcatsn23
export NODE
NFSSERVER=10.2.0.100
export NFSSERVER
INSTALLNIC=eth0
export INSTALLNIC
PRIMARYNIC=eth0
export PRIMARYNIC
MASTER=10.2.0.100
export MASTER
OSVER=sles11
export OSVER
ARCH=ppc64
export ARCH
PROFILE=service-xcattest
export PROFILE
PROVMETHOD=netboot
export PROVMETHOD
PATH=`dirname $0`: $PATH
export PATH
NODESETSTATE=netboot
export NODESETSTATE
UPDATENODE=1
export UPDATENODE
NTYPE=service
export NTYPE
MACADDRESS=16:3d:05:fa:4a:02
export MACADDRESS
NODEID=EA163d05fa4a02EA
export NODEID
MONSERVER=8.112.154.69
export MONSERVER

```

(continues on next page)

(continued from previous page)

```

MONMASTER=10.2.0.100
export MONMASTER
MS_NODEID=0360238fe61815e6
export MS_NODEID
OSPKGS='kernel-ppc64,udev,sysconfig,aaa_base,klogd,device-mapper,bash,openssl,nfs- utils,
↪ksh,syslog-ng,openssh,openssh-askpass,busybox,vim,rpm,bind,bind-utils,dhcp,dhcpd,dhcp-
↪server,dhcp-client,dhcp-relay,bzip2,cron,wget,vsftpd,util-linux,module-init-tools,
↪mkinitrd,apache2,apache2-prefork,perl-Bootloader,psmisc,procps,dbus-1,hal,timezone,
↪rsync,powerpc-utils,bc,iputils,uuid-runtime,unixODBC,gcc,zypper,tar'
export OSPKGS
OTHERPKGS1='xcat/xcat-core/xCAT-rmc,xcat/xcat-core/xCATsn,xcat/xcat-dep/sles11/ppc64/
↪conserver,perl-DBD-mysql,nagios/nagios-nsca-client,nagios/nagios,nagios/nagios-plugins-
↪nrpe,nagios/nagios-nrpe'
export OTHERPKGS1
OTHERPKGS_INDEX=1
export OTHERPKGS_INDEX
## get the diskless networks information. There may be no information.
NETMASK=255.255.255.0
export NETMASK
GATEWAY=10.2.0.100
export GATEWAY
# NIC related attributes for the node for confignetwork postscript
NICIPS=""
export NICIPS
NICHOSTNAMESUFFIXES=""
export NICHOSTNAMESUFFIXES
NICTYPES=""
export NICTYPES
NICCUSTOMSCRIPTS=""
export NICCUSTOMSCRIPTS
NICNETWORKS=""
export NICNETWORKS
NICCOMMENTS=
export NICCOMMENTS
# postscripts-start-here
# defaults-postscripts-start-here
run_ps test1
run_ps syslog
run_ps remoteshell
run_ps syncfiles
run_ps confNagios
run_ps configrmcnode
# defaults-postscripts-end-here
# node-postscripts-start-here
run_ps servicenode
run_ps configeth_new
# node-postscripts-end-here
run_ps setbootfromnet
# postscripts-end-here
# postbootscripts-start-here
# defaults-postbootscripts-start-here
run_ps otherpkgs

```

(continues on next page)

(continued from previous page)

```
# defaults-postbootscripts-end-here
# node-postbootscripts-start-here
run_ps test
# The following line node-postbootscripts-end-here must not be deleted.
# node-postbootscripts-end-here
# postbootscripts-end-here
exit $return_value
```

## Using postinstall scripts

While running `genimage` to generate diskless or statelite osimage, you may want to customize the root image after the package installation step. The `postinstall` attribute of the osimage definition provides a hook to run user specified script(s), in non-chroot mode, against the directory specified by `rootimgdir` attribute.

xCAT ships a default `postinstall` script for the diskless/statelite osimages that must be executed to ensure a successful provisioning of the OS:

```
lsdef -t osimage -o rhels7.3-ppc64le-netboot-compute -i postinstall
Object name: rhels7.3-ppc64le-netboot-compute
postinstall=/opt/xcat/share/xcat/netboot/rh/compute.rhels7.ppc64le.postinstall
```

Customizing the `postinstall` script, can be done by either one of the methods below:

- Append your own `postinstall` scripts

```
chdef -t osimage -o <osimage> -p postinstall=/install/custom/postinstall/rh7/
↪ mypostscript
```

- Create your own `postinstall` script based on the default `postinstall` script

```
cp /opt/xcat/share/xcat/netboot/rh/compute.rhels7.ppc64le.postinstall /install/
↪ custom/postinstall/rh7/mypostscript
# edit /install/custom/postinstall/rh7/mypostscript
chdef -t osimage -o <osimage> postinstall=/install/custom/postinstall/rh7/
↪ mypostscript
```

## Common questions about the usage of postinstall scripts:

### When do postinstall scripts run?

High level flow of `genimage` process:

- install the packages specified by `pkglist` into `rootimgdir` directory
- customize the `rootimgdir` directory
- generate the `initrd` based on the `rootimgdir` directory

The `postinstall` scripts are executed in step b).

### **Do postinstall scripts execute in chroot mode under rootimgdir directory?**

No. Unlike postscripts and postbootscripts, the `postinstall` scripts are run in non-chroot environment, directly on the management node. In the `postinstall` scripts, all the paths of the directories and files are based on `/` of the management node. To reference inside the `rootimgdir`, use the `$IMG_ROOTIMGDIR` environment variable, exported by `genimage`.

### **What are some of the environment variables available to my customized postinstall scripts?**

Environment variables, available to be used in the `postinstall` scripts are listed in `postinstall` attribute section of *linuximage*

## **Synchronizing Files**

### **Add Additional Software Packages**

### **Customize network adapter**

This section describes how to configure network adapters with persistent configuration using xCAT. The `confignetwork` postscript can be used to configure the network interfaces on the compute nodes to support Ethernet adapters, VLAN, BONDS, and BRIDGES.

### **Configure Additional Network Interfaces - confignetwork**

The `confignetwork` postscript can be used to configure the network interfaces on the compute nodes to support Ethernet adapters, VLAN, BONDS, and BRIDGES. `confignetwork` can be used in postscripts during OS provisioning, it can also be executed in `updatenode`. The way the `confignetwork` postscript decides what IP address to give the secondary adapter is by checking the `nics` table, in which the nic configuration information is stored. In order for the `confignetwork` postscript to run successfully, the following attributes must be configured for the node in the `nics` table:

- `nicips`
- `nictypes`
- `nicnetworks`

If configuring VLAN, BOND, or BRIDGES, `nicdevices` in `nics` table must be configured. VLAN, BOND or BRIDGES is only supported on RHEL.

- `nicdevices` - resolves the relationship among the physical network interface devices

The following scenarios are examples to configure Ethernet adapters/BOND/VLAN/Bridge.

1. Configure static install or application Ethernet adapters:
  - Scenario 1: Configure Ethernet Network Interface
2. Configure BOND [RHEL]:
  - Scenario 2: Configure Bond using two Ethernet Adapters
3. Configure VLAN [RHEL]:
  - Scenario 3: Configure VLAN Based on Ethernet Adapter
  - Scenario 4: Configure VLAN Based on Bond Adapters



#### 4. Configure Bridge [RHEL]:

- Scenario 5: Configure Bridge Based On Ethernet NIC
- Scenario 6: Configure Bridge Based on Bond Adapters
- Scenario 7: Configure Bridge Based on VLAN
- Scenario 8: Configure Bridge Based on VLAN,VLAN use BOND adapter

#### 5. Advanced topics:

- Use Customized Scripts To Configure NIC
- Use Extra Parameters In NIC Configuration File
- Configure Aliases

### Configure routes

There are 2 ways to configure OS route in xCAT:

- **makeroutes**: command to add or delete routes on the management node or any given nodes.
- **setroute**: script to replace/add the routes to the node, it can be used in postscripts/postbootscripts.

**makeroutes** or **setroute** will modify OS temporary route, it also modifies persistent route in `/etc/sysconfig/static-routes` file.

Before using **makeroutes** or **setroute** to configure OS route, details of the routes data such as routename, subnet, net mask and gateway should be stored in **routes** table.

**Notes:** the gateway in the **networks** table assigns gateway from DHCP to compute node, so if use **makeroutes** or **setroute** to configure OS static route for compute node, make sure there is no gateway for the specific network in **networks** table.

### Configure routes table

#### 1. Store default route data in routes table:

```
chdef -t route defaultroute net=default mask=255.0.0.0 gateway=10.0.0.101
```

#### 2. Store additional route data in routes table:

```
chdef -t route 20net net=20.0.0.0 mask=255.0.0.0 gateway=0.0.0.0 ifname=eth1
```

#### 3. Check data in routes table:

```
tabdump routes
#routename,net,mask,gateway,ifname,comments,disable
"30net","30.0.0.0","255.0.0.0","0.0.0.0","eth2",,
"20net","20.0.0.0","255.0.0.0","0.0.0.0","eth1",,
"defaultroute","default","255.0.0.0","10.0.0.101",,,
```

### Use `makeroutes` to configure OS route on xCAT management node

1. define the names of the routes to be setup on the management node in site table:

```
chdef -t site mnrouutenames="defaultroute,20net"
lsdef -t site clustersite -i mnrouutenames
    Object name: clustersite
        mnrouutenames=defaultroute,20net
```

2. add all routes from the `mnrouutenames` to the OS route table for the management node:

```
makeroutes
```

3. add route `20net` and `30net` to the OS route table for the management node:

```
makeroutes -r 20net,30net
```

4. delete route `20net` from the OS route table for the management node:

```
makeroutes -d -r 20net
```

### Use `makeroutes` to configure OS route for compute node

1. define the names of the routes to be setup on the compute node:

```
chdef -t cn1 routenames="defaultroute,20net"
```

2. add all routes from the `routenames` to the OS route table for the compute node:

```
makeroutes cn1
```

3. add route `20net` and `30net` to the OS route table for the compute node:

```
makeroutes cn1 -r 20net,30net
```

4. delete route `20net` from the OS route table for the compute node:

```
makeroutes cn1,cn2 -d -r 20net
```

### Use `setroute` to configure OS route for compute node

1. define the names of the routes to be setup on the compute node:

```
chdef -t cn1 routenames="defaultroute,20net"
```

2. If adding `setroute [replace | add]` into the node's postscripts list, `setroute` will be executed during OS deployment on compute node to replace/add routes from `routenames`:

```
chdef cn1 -p postscripts="setroute replace"
```

3. Or if the compute node is already running, use `updatenode` command to run `setroute [replace | add]` postscript:

```
updatenode cn1 -P "setroute replace"
```

## Check result

1. Use `route` command in xCAT management node to check OS route table.
2. Use `xdsh cn1 route` to check compute node OS route table.

## Enable kdump Over Ethernet

### Overview

`kdump` is an feature of the Linux kernel that allows the system to be booted from the context of another kernel. This second kernel reserves a small amount of memory and its only purpose is to capture the core dump in the event of a kernel crash. The ability to analyze the core dump helps to determine causes of system failures.

### xCAT Interface

The following attributes of an osimage should be modified to enable `kdump`:

- `pkglist`
- `exlist`
- `postinstall`
- `dump`
- `crashkernelsize`
- `postscripts`

### Configure the `pkglist` file

The `pkglist` for the osimage needs to include the appropriate RPMs. The following list of RPMs are provided as a sample, always refer to the Operating System specific documentation to ensure the required packages are there for `kdump` support.

- **[RHEL5]**

```
kexec-tools
crash
```

- **[SLES]**

```
kdump
kexec-tools
makedumpfile
```

- **[Ubuntu]**

```
<TODO>
```

## Modify the `exlist` file

The default diskless image created by `copycds` excludes the `/boot` directory in the exclude list file, but this is required for `kdump`.

Update the `exlist` for the target osimage and remove the line `/boot`:

```
./boot*  # <-- remove this line
```

Run `packimage` to update the diskless image with the changes.

## The `postinstall` file

The `kdump` will create a new `initrd` which is used in the dumping stage. The `/tmp` or `/var/tmp` directory will be used as the temporary directory. These two directories are only allocated 10M space by default. You need to enlarge it to 200M. Modify the `postinstall` file to increase `/tmp` space.

- [RHEL5]

```
tmpfs    /var/tmp    tmpfs    defaults,size=500m    0 2
```

- [SLES11]

```
tmpfs    /tmp        tmpfs    defaults,size=500m    0 2
```

- [Ubuntu]

```
<TODO>
```

## The `dump` attribute

To support kernel dumps, the `dump` attribute **must** be set in the osimage definition. If not set, `kdump` service will not be enabled. The `dump` attribute defines the NFS remote path where the crash information is to be stored.

Use the `chdef` command to set a value of the `dump` attribute:

```
chdef -t osimage <image name> dump=nfs://<nfs_server_ip>/<kdump_path>
```

If the NFS server is the Service Node or Management Node, the server can be left out:

```
chdef -t osimage <image name> dump=nfs:///<kdump_path>
```

---

**Note:** Only NFS is currently supported as a storage location. Make sure the NFS remote path (`nfs://<nfs_server_ip>/<kdump_path>`) is exported and it is read-writeable on the node where `kdump` service is enabled.

---

## The crashkernelsize attribute

To allow the Operating System to automatically reserve the appropriate amount of memory for the `kdump` kernel, set `crashkernelsize=auto`. For RHEL 8.5 on System P machines, do not use `auto`, instead set specific size (see below).

For setting specific sizes, use the following example:

- For System X machines, set the `crashkernelsize` using this format:

```
chdef -t osimage <image name> crashkernelsize=<size>M
```

- For Power System AC922, set the `crashkernelsize` using this format:

```
chdef -t osimage <image name> crashkernelsize=<size>M
```

- For System P machines, set the `crashkernelsize` using this format:

```
chdef -t osimage <image name> crashkernelsize=<size>@32M
```

- For System P machines running RHEL 8.5, set the `crashkernelsize` using this format:

```
chdef -t osimage <image name> crashkernelsize=<size>@64M
```

**Note:** The value of the `crashkernelsize` depends on the total physical memory size on the machine. For more about size, refer to [Appedix](#)

If `kdump` start displays error like this:

```
Your running kernel is using more than 70% of the amount of space you reserved for kdump,
↪ you should consider increasing your crashkernel
```

The `crashkernelsize` is not large enough, you should increase the `crashkernelsize` until the error message disappears.

## The enablekdump postscript

xCAT provides a postscript `enablekdump` that can be added to the node definition to automatically start the `kdump` service when the node boots.

```
chdef -t node <node range> -p postscripts=enablekdump
```

## Manually trigger a kernel panic on Linux

Normally, kernel `panic()` will trigger booting into capture kernel. Once the kernel panic is triggered, the node will reboot into the capture kernel, and a kernel dump (`vmcore`) will be automatically saved to the directory on the specified NFS server (`<nfs_server_ip>`).

Check your Operating System specific documentation for the path where the kernel dump is saved. For example:

- [RHEL6]

```
<kdump_path>/var/crash/<node_ip>-<time>/
```

- [SLES11]

```
<kdump_path>/<node_hostname>/<date>
```

To trigger a dump, use the following commands:

```
echo 1 > /proc/sys/kernel/sysrq
echo c > /proc/sysrq-trigger
```

This will force the Linux kernel to crash, and the `address-YYYY-MM-DD-HH:MM:SS/vmcore` file should be copied to the location you set on the NFS server.

## Dump Analysis

Once the system has returned from recovering the crash, you can analyze the kernel dump using the `crash` tool.

1. Locate the recent vmcore dump file.
2. Locate the kernel file for the crash server. The kernel is under `/tftpboot/xcat/netboot/<OS_name="">/<ARCH>/<profile>/kernel` on the management node.
3. Once you have located a vmcore dump file and kernel file, call `crash`:

```
crash <vmcore_dump_file> <kernel_file>
```

---

**Note:** If `crash` cannot find any files, make sure you have the `kernel-debuginfo` package installed.

---

## Appendix

1. OS Documentations on kdump configuration:
  - <http://www.novell.com/support/kb/doc.php?id=3374462>.
  - [https://access.redhat.com/knowledge/docs/en-US/Red\\_Hat\\_Enterprise\\_Linux/6/html/Deployment\\_Guide/s2-kdump-configuration-cli.html](https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/s2-kdump-configuration-cli.html).
  - [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/html/kernel\\_crash\\_dump\\_guide/sect-kdump-config-cli](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/kernel_crash_dump_guide/sect-kdump-config-cli).
  - [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/8/html/system\\_design\\_guide/installing-and-configuring-kdump\\_system-design-guide](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/system_design_guide/installing-and-configuring-kdump_system-design-guide)
2. OS Documentation on dump analysis:
  - [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/6/html/deployment\\_guide/s1-kdump-crashdd](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/deployment_guide/s1-kdump-crashdd)

## Installing a New Kernel in the Diskless Image

[TODO : Verify on ppc64le]

Note: This procedure assumes you are using xCAT 2.6.1 or later.

To add a new kernel, create a directory named <kernelver> under /install/kernels directory, and genimage will pick them up from there.

The following examples assume you have the kernel RPM in /tmp and is using a new kernel in the directory /install/kernels/<kernelver>.

The RPM names below are only examples, substitute your specific level and architecture.

- [RHEL]

The RPM kernel package is usually named: kernel-<kernelver>.rpm. For example, kernel-3.10.0-229.ael7b.ppc64le.rpm means kernelver=3.10.0-229.ael7b.ppc64le.

```
mkdir -p /install/kernels/3.10.0-229.ael7b.ppc64le
cp /tmp/kernel-3.10.0-229.ael7b.ppc64le.rpm /install/kernels/3.10.0-229.ael7b.ppc64le
createrepo /install/kernels/3.10.0-229.ael7b.ppc64le/
```

Append kernel directory /install/kernels/<kernelver> in pkgdir of specific osimage.

```
chdef -t osimage <imagename> -p pkgdir=/install/kernels/3.10.0-229.ael7b.ppc64le/
```

Run genimage/packimage to update the image with the new kernel. Note: If downgrading the kernel, you may need to first remove the rootimg directory.

```
genimage <imagename> -k 3.10.0-229.ael7b.ppc64le
packimage <imagename>
```

- [SLES]

The RPM kernel package is usually separated into two parts: kernel-<arch>-base and kernel<arch>. For example, /tmp contains the following two RPMs:

```
kernel-default-3.12.28-4.6.ppc64le.rpm
kernel-default-base-3.12.28-4.6.ppc64le.rpm
kernel-default-devel-3.12.28-4.6.ppc64le.rpm
```

3.12.28-4.6.ppc64le is NOT the kernel version, 3.12.28-4.ppc64le is the kernel version. The “4.6.ppc64le” is replaced with “4.ppc64le”:

```
mkdir -p /install/kernels/3.12.28-4.ppc64le/
cp /tmp/kernel-default-3.12.28-4.6.ppc64le.rpm /install/kernels/3.12.28-4.ppc64le/
cp /tmp/kernel-default-base-3.12.28-4.6.ppc64le.rpm /install/kernels/3.12.28-4.ppc64le/
cp /tmp/kernel-default-devel-3.12.28-4.6.ppc64le.rpm /install/kernels/3.12.28-4.ppc64le/
```

Append kernel directory /install/kernels/<kernelver> in pkgdir of specific osimage.

```
chdef -t osimage <imagename> -p pkgdir=/install/kernels/3.12.28-4.ppc64le/
```

Run genimage/packimage to update the image with the new kernel. Note: If downgrading the kernel, you may need to first remove the rootimg directory.

Since the kernel version name is different from the kernel rpm package name, the -k flag MUST to be specified on the genimage command.

```
genimage <imagename> -k 3.12.28-4-ppc64le 3.12.28-4.6
packimage <imagename>
```

## Installing New Kernel Drivers to Diskless Initrd

The kernel drivers in the diskless initrd are used for the devices during the netboot. If you are missing one or more kernel drivers for specific devices (especially for the network device), the netboot process will fail. xCAT offers two approaches to add additional drivers to the diskless initrd during the running of genimage.

Use the ‘-n’ flag to add new drivers to the diskless initrd:

```
genimage <imagename> -n <new driver list>
```

Generally, the genimage command has a default driver list which will be added to the initrd. But if you specify the ‘-n’ flag, the default driver list will be replaced with your <new driver list>. That means you need to include any drivers that you need from the default driver list into your <new driver list>.

The default driver list:

```
rh-x86:   tg3 bnx2 bnx2x e1000 e1000e igb mlx_en virtio_net be2net
rh-ppc:   e1000 e1000e igb ibmveth ehea
rh-ppcle: ext3 ext4
sles-x86: tg3 bnx2 bnx2x e1000 e1000e igb mlx_en be2net
sels-ppc: tg3 e1000 e1000e igb ibmveth ehea be2net
sles-ppcle: scsi_mod libata scsi_tgt jbd2 mbcache crc16 virtio virtio_ring libahci crc-
↳ t10dif scsi_transport_srp af_packet ext3 ext4 virtio_pci virtio_blk scsi_dh ahci
↳ megaraid_sas sd_mod ibmvscsi
```

Note: With this approach, xCAT will search for the drivers in the rootimage. You need to make sure the drivers have been included in the rootimage before generating the initrd. You can install the drivers manually in an existing rootimage (using chroot) and run genimage again, or you can use a postinstall script to install drivers to the rootimage during your initial genimage run.

Use the driver rpm package to add new drivers from rpm packages to the diskless initrd. Refer to the [Configure Additional Network Interfaces - confignetwork](#) for details.

## Accelerating the diskless initrd and rootimg generating

Generating diskless initrd with genimage and compressed rootimg with packimage and liteimg is a time-consuming process, it can be accelerated by enabling parallel compression tool pigz on the management node with multiple processors and cores. See [Appendix](#) for an example on packimage performance optimized with pigz enabled.

## Enabling the pigz for diskless initrd and rootimg generating

The parallel compression tool pigz can be enabled by installing pigz package on the management server or diskless rootimg. Depending on the method of generating the initrd and compressed rootimg, the steps differ in different Linux distributions.

- [RHEL]

The package pigz is shipped in Extra Packages for Enterprise Linux (or EPEL) instead of RedHat iso, this involves some complexity.



Extra Packages for Enterprise Linux (or EPEL) is a Fedora Special Interest Group that creates, maintains, and manages a high quality set of additional packages for Enterprise Linux, including, but not limited to, Red Hat Enterprise Linux (RHEL), CentOS and Scientific Linux (SL), Oracle Linux (OL).

EPEL has an `epel-release` package that includes `gpg` keys for package signing and repository information. Installing this package for your Enterprise Linux version should allow you to use normal tools such as `yum` to install packages and their dependencies.

Refer to the <http://fedoraproject.org/wiki/EPEL> for more details on EPEL

- 1) Enabling the `pigz` in `genimage` (only supported in RHEL 7 or above)

`pigz` should be installed in the diskless rootimg. Download `pigz` package from <https://dl.fedoraproject.org/pub/epel/>, then customize the diskless osimage to install `pigz` as the additional packages, see *Install Additional Other Packages* for more details.

- 2) Enabling the `pigz` in `packimage`

`pigz` should be installed on the management server. Download `pigz` package from <https://dl.fedoraproject.org/pub/epel/>, then install the `pigz` with `yum` or `rpm`.

#### • [UBUNTU]

Make sure the `pigz` is installed on the management node with the following command:

```
dpkg -l | grep pigz
```

If not, `pigz` can be installed with the following command:

```
apt-get install pigz
```

#### • [SLES]

- 1) Enabling the `pigz` in `genimage` (only supported in SLES12 or above)

`pigz` should be installed in the diskless rootimg, since `pigz` is shipped in the SLES iso, this can be done by adding `pigz` into the `pkglist` of diskless osimage.

- 2) Enabling the `pigz` in `packimage`

Make sure the `pigz` is installed on the management node with the following command:

```
rpm -qa | grep pigz
```

If not, `pigz` can be installed with the following command:

```
zypper install pigz
```

## Appendix: An example on packimage performance optimization with “pigz” enabled

This is an example on performance optimization with `pigz` enabled.

In this example, a `xcAT` command `packimage rhels7-ppc64-netboot-compute` is run on a Power 7 machine with 4 cores.

The system info:

```
# uname -a
Linux c910f03c01p03 3.10.0-123.el7.ppc64 #1 SMP Mon May 5 11:18:37 EDT 2014 ppc64 ppc64_
↳ ppc64 GNU/Linux
```

(continues on next page)

(continued from previous page)

```
# cat /etc/os-release
NAME="Red Hat Enterprise Linux Server"
VERSION="7.0 (Maipo)"
ID="rhel"
ID_LIKE="fedora"
VERSION_ID="7.0"
PRETTY_NAME="Red Hat Enterprise Linux Server 7.0 (Maipo)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:redhat:enterprise_linux:7.0:GA:server"
HOME_URL="https://www.redhat.com/"
BUG_REPORT_URL="https://bugzilla.redhat.com/"

REDHAT_BUGZILLA_PRODUCT="Red Hat Enterprise Linux 7"
REDHAT_BUGZILLA_PRODUCT_VERSION=7.0
REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux"
REDHAT_SUPPORT_PRODUCT_VERSION=7.0
```

The CPU info:

```
# cat /proc/cpuinfo
processor       : 0
cpu            : POWER7 (architected), altivec supported
clock          : 3550.000000MHz
revision       : 2.0 (pvr 003f 0200)

processor       : 1
cpu            : POWER7 (architected), altivec supported
clock          : 3550.000000MHz
revision       : 2.0 (pvr 003f 0200)

processor       : 2
cpu            : POWER7 (architected), altivec supported
clock          : 3550.000000MHz
revision       : 2.0 (pvr 003f 0200)

processor       : 3
cpu            : POWER7 (architected), altivec supported
clock          : 3550.000000MHz
revision       : 2.0 (pvr 003f 0200)

timebase       : 512000000
platform       : pSeries
model          : IBM,8233-E8B
machine        : CHRP IBM,8233-E8B
```

The time spent on packimage with gzip:

```
# time packimage rhels7-ppc64-netboot-compute
Packing contents of /install/netboot/rhels7/ppc64/compute/rootimg
compress method:gzip
```

(continues on next page)

(continued from previous page)

```
real    1m14.896s
user    0m0.159s
sys     0m0.019s
```

The time spent on packimage with pigz:

```
# time packimage rhels7-ppc64-netboot-compute
Packing contents of /install/netboot/rhels7/ppc64/compute/rootimg
compress method:pigz

real    0m23.177s
user    0m0.176s
sys     0m0.016s
```

## Trim diskless rootimg

To reduce the memory and boot-up time for the diskless node, the initrd and rootimg.gz should be kept as compact as possible under the premise of meeting the user's requirements.

## Exclude list

xCAT provides an attribute **exlist** in the **osimage** object definition, that allows the user to select files to exclude when building the rootimg.gz file for the diskless node.

Take the osimage **sles12.1-ppc64le-netboot-compute** for example:

```
# lsdef -t osimage -o sles12.1-ppc64le-netboot-compute -i exlist
Object name: sles12.1-ppc64le-netboot-compute
    exlist=/opt/xcat/share/xcat/netboot/sles/compute.sles12.ppc64le.exlist
```

## Content of the Exclude List file

The file specified in linuximage.exlist includes **relative path** of the directories and files that will be excluded from the **rootimg.gz** generated by packimage. The **relative path** assumes the rootimg directory, /install/netboot/sles12.1/ppc64le/compute/rootimg here, to be the base directory.<sup>1</sup>

The following is a sample of exlist file

```
...
./usr/share/X11/locale/*
./usr/lib/perl[0-9]/[0-9.]* /ppc64le-linux-thread-multi/Encode/JP*
+ ./usr/share/X11/locale/C*
...
```

The content above presents some syntax supported in exlist file:

- Exclude files:

<sup>1</sup> The exlist file entry should not end with a slash /, For example, this entry will never match anything: ./usr/lib/perl[0-9]/[0-9.]\* /ppc64le-linux-thread-multi/Encode/.

```
./usr/share/X11/locale/*
```

All the files and subdirectories under `rootimg/usr/share/X11/locale/` will be excluded.

- Exclude Files using Patterns<sup>2</sup>:

```
./usr/lib/perl[0-9]/[0-9.]*ppc64le-linux-thread-multi/Encode/JP*
```

Use regular expression to easily exclude files. The above example will exclude any Perl library installed under `/usr/lib/` matching `ppc64le-linux-thread-multi/Encode/JP*`

- Include files:

```
+./usr/share/locale/C*
```

It is useful to include files following an exclude entry to quickly remove a larger set of files using a wildcard and then adding back the few necessary files using the `+` sign. In the above example, all the files and sub-directories matching the pattern `/usr/share/locale/C*` will be included in the `rootimg.gz` file.

## Customize the `exlist` file and the `osimage` definition

Check the default `exlist` file and make sure:

- all files and directories you do not want in the image will be excluded from the `rootimg`.
- no file or directory you need will be excluded from the `rootimg`.

If you want to customize the `osimage` `sles12.1-ppc64le-netboot-compute` with your own `exlist` file, follow the following steps:

```
#create a customized exlist file based on the default one
cp /opt/xcat/share/xcat/netboot/sles/compute.sles12.ppc64le.exlist /install/custom/
↪netboot/sles/compute.sles12.ppc64le.exlist

#edit the newly created exlist file according to your need
vi /install/custom/netboot/sles/compute.sles12.ppc64le.exlist

#specify the newly created exlist file in the osimage definition
chdef -t osimage -o sles12.1-ppc64le-netboot-compute exlist=/install/custom/netboot/sles/
↪compute.sles12.ppc64le.exlist
```

## Enabling the `localdisk` option

---

**Note:** You can skip this section if not using the `localdisk` option in your `litefile` table.

---

---

<sup>2</sup> Pattern match test applies to the whole file name, starting from one of the start points specified in the `exlist` file entry. The regex syntax should comply with the regex syntax of system command `find -path`, refer to its doc for details.

## Define how to partition the local disk

When a node is deployed, the local hard disk needs to be partitioned and formatted before it can be used. This section explains how provide a configuration file that tells xCAT to partition a local disk and make it ready to use for the directories listed in the litefile table.

The configuration file needs to be specified in the `partitionfile` attribute of the `osimage` definition. The configuration file includes several sections:

- Global parameters to control enabling or disabling the function
- `[disk]` section to control the partitioning of the disk
- `[localspace]` section to control which partition will be used to store the `localdisk` directories listed in the `litefile` table
- `[swap space]` section to control the enablement of the swap space for the node.

An example `localdisk` configuration file:

```
enable=yes
enablepart=no

[disk]
dev=/dev/sda
clear=yes
parts=10,20,30

[disk]
dev=/dev/sdb
clear=yes
parts=100M-200M,1G-2G

[disk]
dev=/dev/sdc
ptype=gpt
clear=yes
parts=10,20,30

[localspace]
dev=/dev/sda1
fstype=ext4

[swap space]
dev=/dev/sda2
```

The two global parameters `enable` and `enablepart` can be used to control the enabling/disabling of the functions:

- `enable`: The `localdisk` feature only works when `enable` is set to `yes`. If it is set to `no`, the `localdisk` configuration will not be run.
- `enablepart`: The partition action (refer to the `[disk]` section) will be run only when `enablepart=yes`.

The `[disk]` section is used to configure how to partition a hard disk:

- `dev`: The path of the device file.
- `clear`: If set to `yes` it will clear all the existing partitions on this disk.
- `ptype`: The partition table type of the disk. For example, `msdos` or `gpt`, and `msdos` is the default.

- `fstype`: The file system type for the new created partitions. `ext4` is the default.
- `parts`: A comma separated list of space ranges, one for each partition that will be created on the device. The valid format for each space range is `<startpoint>-<endpoint>` or `<percentage of the disk>`. For example, you could set it to `100M-10G` or `50`. If set to `50`, 50% of the disk space will be assigned to that partition.

The `[localspace]` section is used to specify which partition will be used as local storage for the node.

- `dev`: The path of the partition.
- `fstype`: The file system type on the partition.

the `[swapspace]` section is used to configure the swap space for the statelite node.

- `dev`: The path of the partition file which will be used as the swap space.

To enable the local disk capability, create the configuration file (for example in `/install/custom`) and set the path in the `partitionfile` attribute for the `osimage`:

```
chdef -t osimage <osimage> partitionfile=/install/custom/cfglocaldisk
```

Now all nodes that use this `osimage` (i.e. have their `provmethod` attribute set to this `osimage` definition name), will have its local disk configured.

### Configure the files in the litefile table

For the files/directories to store on the local disk, add an entry in the `litefile` table:

```
"ALL", "/tmp/", "localdisk", ,
```

---

**Note:** you do not need to specify the swap space in the `litefile` table. Just putting it in the `partitionfile` config file is enough.

---

Add an entry in policy table to permit the running of the `getpartition` command from the node

```
chtab priority=7.1 policy.commands=getpartition policy.rule=allow
```

Run `genimage` and `packimage` for the `osimage`

---

**Note:** `enablepart=yes` in partition file will partition the local disk at every boot. If you want to preserve the contents on local disk at next boot, change to `enablepart=no` after the initial provision. A log file `/.sllocal/log/localdisk.log` on the target node can be used for debugging.

---

### Generate Diskless Image

The `copycds` command copies the contents of the Linux media to `/install/<os>/<arch>` so that it will be available for installing nodes or creating diskless images. After executing `copycds`, there are several `osimage` definitions created by default. Run `lsdef -t osimage` to view these images:

```
lsdef -t osimage
```

The output should be similar to the following:

```
rhels8.5.0-ppc64le-install-compute (osimage)
rhels8.5.0-ppc64le-install-service (osimage)
rhels8.5.0-ppc64le-netboot-compute (osimage)
```

The netboot-compute is the default **diskless** osimage created for rhels8.5 ppc64le. Run genimage to generate a diskless image based on the *rhels8.5.0-ppc64le-netboot-compute* definition:

```
genimage rhels8.5.0-ppc64le-netboot-compute
```

Before packing the diskless image, you have the opportunity to change any files in the image by changing to the rootimgdir and making modifications. (e.g. /install/netboot/rhels7.1/ppc64le/compute/rootimg).

However it's recommended that all changes to the image are made via post install scripts so that it's easily repeatable. Refer to *Prescripts and Postscripts* for more details.

## Pack Diskless Image

After running genimage to create the image, run packimage to create the ramdisk:

```
packimage rhels8.5.0-ppc64le-netboot-compute
```

## Export and Import Image

### Overview

**Note:** There is a current restriction that exported 2.7 xCAT images cannot be imported on 2.8 xCAT <https://sourceforge.net/p/xcat/bugs/3813/>.

We want to create a system of making xCAT images more portable so that they can be shared and prevent people from reinventing the wheel. While every install is unique there are some things that can be shared among different sites to make images more portable. In addition, creating a method like this allows us to create snap shots of images we may find useful to revert to in different situations.

Image exporting and importing is supported for stateful (diskful) and stateless (diskless) clusters. The following documentation will show how to use *imgexport* to export images and *imgimport* to import images.

## Exporting an image

Working image:

```
lsdef -t osimage myimage
Object name: myimage
exlist=/install/custom/netboot/sles/compute1.exlist
imagetype=linux
netdrivers=e1000
osarch=ppc64le
osname=Linux
osvers=sles12
otherpkgdir=/install/post/otherpkgs/sles12/ppc64
```

(continues on next page)

(continued from previous page)

```
otherpkglist=/install/custom/netboot/sles/compute1.otherpkgs.pkglist
pkgdir=/install/sles11/ppc64le
pkglist=/install/custom/netboot/sles/compute1.pkglist
postinstall=/install/custom/netboot/sles/compute1.postinstall
profile=compute1
provmethod=netboot
rootimgdir=/install/netboot/sles12/ppc64le/compute1
synclists=/install/custom/netboot/sles/compute1.list
```

Run the `imgexport` command:

```
imgexport myimage -p node1 -e /install/postscripts/myscript1 -e /install/postscripts/
↪myscript2
```

A bundle file called *myimage.tgz* will be created under the current directory. The bundle file contains the ramdisk, boot kernel, the root image and all the configuration files for generating the image for a diskless node. For diskful, it contains the kickstart/autoyast configuration file. (see appendix). The optional **-p** flag puts the names of the postscripts for node1 into the image bundle. The optional **-e** flags put additional files into the bundle. In this case two postscripts *myscript1* and *myscript2* are included. This image can now be used on other systems.

## Importing an image

1. Download the image bundle file generated by the `imgexport`.
2. Run the `imgimport` command.:

```
imgimport myimage.tgz -p group1
```

This command fills out the `osimage` and `linuximage` tables, and populates file directories with appropriate files from the image bundle file such as ramdisk, boot kernel, root image, configuration files for diskless. Any additional files that come with the bundle file will also be put into the appropriate directories. If optional **-p** flag is specified, the postscript names that come with the image will be put into the `postscripts` table for the given node or group.

## Copy an image to a new image name on the MN

Very often, the user wants to make a copy of an existing image on the same xCAT MN as a start point to make modifications. In this case, you can run `imgexport` first as described above, then run `imgimport` with **-f** flag to change the profile name of the image. That way the image will be copied into a different directory on the same xCAT MN.:

```
imgimport myimage.tgz -p group1 -f compute2
```

## Modify an image (optional)

Skip this section if you want to use the image as is.

1. You can modify the image to fit your needs. The following can be modified.
  - `.pkglist` file to add or remove packages that are from the os distro
  - `.otherpkgs.pkglist` to add or remove packages from other sources. Refer to [Using updatenode](#) for details
  - For diskful, `.tmpl` file to change the kickstart/autoyast configuration



- .synclist file to change the files that are going to be synchronized to the nodes
- postscripts table for the nodes to be deployed
- the osimage and/or linuximage tables for the location of the source rpms and the rootimage location

2. Run genimage:

```
genimage image_name
```

3. Run packimage:

```
packimage image_name
```

## Deploying nodes

You can now deploy the node with the new `<image_name>`

```
rinstall <noderange> osimage=<image_name>
```

## Appendix

You can only export/import one image at a time. Each tarball will have the following simple structure:

```
manifest.xml
<files>
extra/ (optional)
```

### manifest.xml

The `manifest.xml` will be analogous to an autoyast or windows `unattend.xml` file where it tells xCAT how to store the items. The following is an example for a diskless cluster:

```
manifest.xml:

<?xml version="1.0"?>
<xcatimage>
  <exlist>/install/custom/netboot/sles/compute1.exlist</exlist>
  <extra>
    <dest>/install/postscripts</dest>
    <src>/install/postscripts/myscript1</src>
  </extra>
  <imagename>myimage</imagename>
  <imagetype>linux</imagetype>
  <kernel>/install/netboot/sles12/ppc64le/compute1/kernel</kernel>
  <netdrivers>e1000</netdrivers>
  <osarch>ppc64le</osarch>
  <osname>Linux</osname>
  <osvers>sles12</osvers>
  <otherpkgdir>/install/post/otherpkgs/sles12/ppc64</otherpkgdir>
  <otherpkglist>/install/custom/netboot/sles/compute1.otherpkgs.pkglist</otherpkglist>
```

(continues on next page)

(continued from previous page)

```
<pkgdir>/install/sles12/ppc64le</pkgdir>
<pkglist>/install/custom/netboot/sles/compute1.pkglist</pkglist>
<postbootscripts>my4,otherpkgs,my3,my4</postbootscripts>
<postinstall>/install/custom/netboot/sles/compute1.postinstall</postinstall>
<postscripts>syslog,remoteshell,my1,configrmcnode,syncfiles,my1,my2</postscripts>
<profile>compute1</profile>
<provmethod>netboot</provmethod>
<ramdisk>/install/netboot/sles12/ppc64le/compute1/initrd-diskless.gz</ramdisk>
<rootimg>/install/netboot/sles12/ppc64le/compute1/rootimg.gz</rootimg>
<rootimgdir>/install/netboot/sles12/ppc64le/compute1</rootimgdir>
<synclists>/install/custom/netboot/sles/compute1.list</synclists>
</xcatimage>
```

In the above example, we have a directive of where the files came from and what needs to be processed.

Note that even though source destination information is included, all files that are standard will be copied to the appropriate place that xCAT thinks they should go.

## Exported files

The following files will be exported, assuming x is the profile name:

For diskful:

```
x.pkglist
x.otherpkgs.pkglist
x.tmpl
x.synclist
```

For diskless:

```
kernel
initrd.gz
rootimg.gz
x.pkglist
x.otherpkgs.pkglist
x.synclist
x.postinstall
x.exlist
```

---

**Note:** Although the postscripts names can be exported by using the **-p** flag, the postscripts themselves are not included in the bundle file by default. Use **-e** flag to get them included one by one if needed.

---

## Initialize the Compute for Deployment

XCAT use **nodeset** command to associate a specific image to a node which will be installed with this image.

```
nodeset <nodename> osimage=<osimage>
```

There are more attributes of nodeset used for some specific purpose or specific machines, for example:

- **runimage**: If you would like to run a task after deployment, you can define that task with this attribute.
- **runcmd**: This instructs the node to boot to the xCAT nbfs environment and proceed to configure BMC for basic remote access. This causes the IP, netmask, gateway, username, and password to be programmed according to the configuration table.
- **shell**: This instructs the node to boot to the xCAT genesis environment, and present a shell prompt on console. The node will also be able to be sshed into and have utilities such as wget, tftp, scp, nfs, and cifs. It will have storage drivers available for many common systems.

Choose such additional attribute of nodeset according to your requirement, if want to get more information about nodeset, refer to nodeset's man page.

## Start the OS Deployment

Start the deployment involves two key operations. First specify the boot device of the next boot to be network, then reboot the node:

For **Power servers**, those two operations can be completed by one command **rnetboot**:

```
rnetboot <node>
```

For **x86\_64 servers**, those two operations need two independent commands.

1. set the next boot device to be from the "network"

```
rsetboot <node> net
```

2. Reboot the xSeries server: ::

```
rpower <node> reset
```

## Statelite Installation

### Overview

This document details the design and setup for the statelite solution of xCAT. **Statelite** is an intermediate mode between **diskful** and **diskless**.

Statelite provides two kinds of efficient and flexible solutions, most of the OS image can be NFS mounted read-only, or the OS image can be in the ramdisk with tmpfs type. Different from the stateless solution, statelite provides a configurable list of directories and files that can be read-write. These read-write directories and files can be configured to either persist or not persist across reboots.

### Solutions

There are two solutions: NFSROOT-based and RAMdisk-based.

1. **NFSROOT-based(default):**

1. rootfstype in the osimage xCAT data objects is left as blank, or set to `nfs`, the NFSROOT-base statelite solution will be enabled.
  2. the ROOTFS is NFS mounted read-only.
2. **RAMdisk-based:**
1. rootfstype in the osimage xCAT data objects is set to `ramdisk`.
  2. one image file will be downloaded when the node is booting up, and the file will be extracted to the ramdisk, and used as the ROOTFS.

### Advantages

Statelite offers the following advantages over xCAT's stateless (RAMdisk) implementation:

1. Some files can be made persistent over reboot. This is useful for license files or database servers where some state is needed. However, you still get the advantage of only having to manage a single image.
2. Changes to hundreds of machines can take place instantly, and automatically, by updating one main image. In most cases, machines do not need to reboot for these changes to take affect. This is only for the NFSROOT-based solution.
3. Ease of administration by being able to lock down an image. Many parts of the image can be read-only, so no modifications can transpire without updating the central image.
4. Files can be managed in a hierarchical manner. For example: Suppose you have a machine that is in one lab in Tokyo and another in London. You could set table values for those machines in the xCAT database to allow machines to sync from different places based on their attributes. This allows you to have one base image with multiple sources of file overlay.
5. Ideal for virtualization. In a virtual environment, you may not want a disk image (neither stateless nor stateful) on every virtual node as it consumes memory and disk. Virtualizing with the statelite approach allows for images to be smaller, easier to manage, use less disk, less memory, and more flexible.

### Disadvantages

However, there are still several disadvantages, especially for the NFSROOT-based solution.

1. NFS Root requires more network traffic to run as the majority of the disk image runs over NFS. This may depend on your workload, but can be minimized. Since the bulk of the image is read-only, NFS caching on the server helps minimize the disk access on the server, and NFS caching on the client helps reduce the network traffic.
2. NFS Root can be complex to set up. As more files are created in different places, there are greater chances for failures. This flexibility is also one of the great virtues of Statelite. The image can work in nearly any environment.

### Configuration

Statelite configuration is done using the following tables in xCAT:

- `litefile`
- `litetree`
- `statelite`
- `policy`
- `nodes`

## litefile table

The litefile table specifies the directories and files on the statelite nodes that should be read/write, persistent, or read-only overlay. All other files in the statelite nodes come from the read-only statelite image.

1. The first column in the litefile table is the image name this row applies to. It can be an exact osimage definition name, an osimage group (set in the groups attribute of osimages), or the keyword ALL.
2. The second column in the litefile table is the full path of the directory or file on the node that you are setting options for.
3. The third column in the litefile table specifies options for the directory or file:
  1. tmpfs - It provides a file or directory for the node to use when booting, its permission will be the same as the original version on the server. In most cases, it is read-write; however, on the next statelite boot, the original version of the file or directory on the server will be used, it means it is non-persistent. This option can be performed on files and directories.
  2. rw - Same as above. Its name "rw" does NOT mean it always be read-write, even in most cases it is read-write. Do not confuse it with the "rw" permission in the file system.
  3. persistent - It provides a mounted file or directory that is copied to the xCAT persistent location and then over-mounted on the local file or directory. Anything written to that file or directory is preserved. It means, if the file/directory does not exist at first, it will be copied to the persistent location. Next time the file/directory in the persistent location will be used. The file/directory will be persistent across reboots. Its permission will be the same as the original one in the statelite location. It requires the statelite table to be filled out with a spot for persistent statelite. This option can be performed on files and directories.
  4. con - The contents of the pathname are concatenated to the contents of the existing file. For this directive the searching in the litetree hierarchy does not stop when the first match is found. All files found in the hierarchy will be concatenated to the file when found. The permission of the file will be "-rw-r--r--", which means it is read-write for the root user, but readonly for the others. It is non-persistent, when the node reboots, all changes to the file will be lost. It can only be performed on files. Do not use it for one directory.
  5. ro - The file/directory will be overmounted read-only on the local file/directory. It will be located in the directory hierarchy specified in the litetree table. Changes made to this file or directory on the server will be immediately seen in this file/directory on the node. This option requires that the file/directory to be mounted must be available in one of the entries in the litetree table. This option can be performed on files and directories.
  6. tmpfs,rw - Only for compatibility it is used as the default option if you leave the options column blank. It has the same semantics with the link option, so when adding new items into the \_litefile table, the link option is recommended.
  7. link - It provides one file/directory for the node to use when booting, it is copied from the server, and will be placed in tmpfs on the booted node. In the local file system of the booted node, it is one symbolic link to one file/directory in tmpfs. And the permission of the symbolic link is "lrwxrwxrwx", which is not the real permission of the file/directory on the node. So for some application sensitive to file permissions, it will be one issue to use "link" as its option, for example, "/root/.ssh/", which is used for SSH, should NOT use "link" as its option. It is non-persistent, when the node is rebooted, all changes to the file/directory will be lost. This option can be performed on files and directories.
  8. link,ro - The file is readonly, and will be placed in tmpfs on the booted node. In the local file system of the booted node, it is one symbolic link to the tmpfs. It is non-persistent, when the node is rebooted, all changes to the file/directory will be lost. This option requires that the file/directory to be mounted must be available in one of the entries in the litetree table. The option can be performed on files and directories.
  9. link,con - Similar to the "con" option. All the files found in the litetree hierarchy will be concatenated to the file when found. The final file will be put to the tmpfs on the booted node. In the local file system of

the booted node, it is one symbolic link to the file/directory in tmpfs. It is non-persistent, when the node is rebooted, all changes to the file will be lost. The option can only be performed on files.

10. link,persistent - It provides a mounted file or directory that is copied to the xCAT persistent location and then over-mounted to the tmpfs on the booted node, and finally the symbolic link in the local file system will be linked to the over-mounted tmpfs file/directory on the booted node. The file/directory will be persistent across reboots. The permission of the file/directory where the symbolic link points to will be the same as the original one in the statelite location. It requires the statelite table to be filled out with a spot for persistent statelite. The option can be performed on files and directories.
11. localdisk - The file or directory will be stored in the local disk of the statelite node. Refer to the section To enable the localdisk option to enable the 'localdisk' support.

Currently, xCAT does not handle the relative links very well. The relative links are commonly used by the system libraries, for example, under /lib/ directory, there will be one relative link matching one .so file. So, when you add one relative link to the litefile table (Not recommend), make sure the real file also be included, or put its directory name into the litefile table.

---

**Note:** It is recommended that you specify at least the entries listed below in the litefile table, because most of these files need to be writeable for the node to boot up successfully. When any changes are made to their options, make sure they won't affect the whole system. If you want to run a command like /bin/ping using non-root users, add this command into litefile, then root user have privilege to authorize the command for non-root users.

---

### Sample Data for Redhat statelite setup

This is the minimal list of files needed, you can add additional files to the litefile table.

```
#image,file,options,comments,disable
"ALL","/etc/adjtime","tmpfs",,
"ALL","/etc/securetty","tmpfs",,
"ALL","/etc/lvm/", "tmpfs",,
"ALL","/etc/ntp.conf","tmpfs",,
"ALL","/etc/rsyslog.conf","tmpfs",,
"ALL","/etc/rsyslog.conf.XCATORIG","tmpfs",,
"ALL","/etc/rsyslog.d/", "tmpfs",,
"ALL","/etc/udev/", "tmpfs",,
"ALL","/etc/ntp.conf.predhclient","tmpfs",,
"ALL","/etc/resolv.conf","tmpfs",,
"ALL","/etc/yp.conf","tmpfs",,
"ALL","/etc/resolv.conf.predhclient","tmpfs",,
"ALL","/etc/sysconfig/", "tmpfs",,
"ALL","/etc/ssh/", "tmpfs",,
"ALL","/etc/inittab","tmpfs",,
"ALL","/tmp/", "tmpfs",,
"ALL","/var/", "tmpfs",,
"ALL","/opt/xcat/", "tmpfs",,
"ALL","/xcatpost/", "tmpfs",,
"ALL","/etc/systemd/system/multi-user.target.wants/", "tmpfs",,
"ALL","/root/.ssh/", "tmpfs",,
"ALL","/etc/rc3.d/", "tmpfs",,
"ALL","/etc/rc2.d/", "tmpfs",,
"ALL","/etc/rc4.d/", "tmpfs",,
"ALL","/etc/rc5.d/", "tmpfs",,
```

## Sample Data for SLES statelite setup

This is the minimal list of files needed, you can add additional files to the litefile table.

```
#image,file,options,comments,disable
"ALL","/etc/lvm/","tmpfs",,
"ALL","/etc/ntp.conf","tmpfs",,
"ALL","/etc/ntp.conf.org","tmpfs",,
"ALL","/etc/resolv.conf","tmpfs",,
"ALL","/etc/hostname","tmpfs",,
"ALL","/etc/ssh/","tmpfs",,
"ALL","/etc/sysconfig/","tmpfs",,
"ALL","/etc/syslog-ng/","tmpfs",,
"ALL","/etc/inittab","tmpfs",,
"ALL","/tmp/","tmpfs",,
"ALL","/etc/init.d/rc3.d/","tmpfs",,
"ALL","/etc/init.d/rc5.d/","tmpfs",,
"ALL","/var/","tmpfs",,
"ALL","/etc/yp.conf","tmpfs",,
"ALL","/etc/fstab","tmpfs",,
"ALL","/opt/xcat/","tmpfs",,
"ALL","/xcatpost/","tmpfs",,
"ALL","/root/.ssh/","tmpfs",,
"ALL","/etc/systemd/system/","tmpfs",,
"ALL","/etc/adjtime","tmpfs",,
```

## litetree table

The litetree table controls where the initial content of the files in the litefile table come from, and the long term content of the ro files. When a node boots up in statelite mode, it will by default copy all of its tmpfs files from the `.default` directory of the root image, for example `/install/netboot/rhels7.3/x86_64/compute/rootimg/.default`, so there is not required to set up a litetree table. If you decide that you want some of the files pulled from different locations that are different per node, you can use this table.

You can choose to use the defaults and not set up a litetree table.

## statelite table

The statelite table specifies location on an NFS server where a nodes persistent files are stored. This is done by entering the information into the statelite table.

In the statelite table, the node or nodegroups in the table must be unique; that is a node or group should appear only once in the first column table. This makes sure that only one statelite image can be assigned to a node. An example would be:

```
"compute",,"<nfssvr_ip>:/gpfs/state",,
```

Any nodes in the compute node group will have their state stored in the `/gpfs/state` directory on the machine with `<nfssvr_ip>` as its IP address.

When the node boots up, then the value of the `statemnt` attribute will be mounted to `/.statelite/persistent`. The code will then create the following subdirectory `/.statelite/persistent/<nodename>`, if there are persistent files that have been added in the litefile table. This directory will be the root of the image for this node's persistent files.

By default, xCAT will do a hard NFS mount of the directory. You can change the mount options by setting the `mntopts` attribute in the `statelite` table.

Also, to set the `statemnt` attribute, you can use variables from xCAT database. It follows the same grammar as the `litetree` table. For example:

```
#node,image,statemnt,mntopts,comments,disable
"cn1",, "$noderes.nfsserver:/lite/state/$nodetype.profile", "soft,timeo=30",,
```

**Note:** Do not name your persistent storage directory with the node name, as the node name will be added in the directory automatically. If you do, then a directory named `/state/cn1` will have its state tree inside `/state/cn1/cn1`.

## Policy

Ensure policies are set up correctly in the Policy Table. When a node boots up, it queries the xCAT database to get the `litefile` and `litetree` table information. In order for this to work, the commands (of the same name) must be set in the policy table to allow nodes to request it. This should happen automatically when xCAT is installed, but you may want to verify that the following lines are in the policy table:

```
chdef -t policy -o 4.7 commands=litefile rule=allow
chdef -t policy -o 4.8 commands=litetree rule=allow
```

## noderes

`noderes.nfsserver` attribute can be set for the NFSroot server. If this is not set, then the default is the Management Node.

`noderes.nfsdir` can be set. If this is not set, the default is `/install`

## Provision statelite

### Show current provisioning method

To determine the current provisioning method of your node, execute:

```
lsdef <noderange> -i provmethod
```

**Note:** `synfiles` is not currently supported for `statelite` nodes.

## Generate default statelite image from distro media

In this example, we are going to create a new compute node `osimage` for `rhels7.3` on `ppc64le`. We will set up a test directory structure that we can use to create our image. Later we can just move that into production.

Use the `copycds` command to copy the appropriate iso image into the `/install` directory for xCAT. The `copycds` commands will copy the contents to `/install/rhels7.3/<arch>`. For example:

```
copycds RHEL-7.3-20161019.0-Server-ppc64le-dvd1.iso
```



The contents are copied into `/install/rhels7.3/ppc64le/`

The configuration files pointed to by the attributes are the defaults shipped with xCAT. We will want to copy them to the `/install` directory, in our example the `/install/test` directory and modify them as needed.

## Statelike Directory Structure

Each statelike image will have the following directories:

```
/.statelike/tmpfs/
/.default/
/etc/init.d/statelike
```

All files with link options, which are symbolic links, will link to `/.statelike/tmpfs`.

`tmpfs` files that are persistent link to `/.statelike/persistent/<nodename>/`, `/.statelike/persistent/<nodename>` is the directory where the node's individual storage will be mounted to.

`/.default` is where default files will be copied to from the image to `tmpfs` if the files are not found in the litetree hierarchy.

## Customize your statelike osimage

### Create the osimage definition

Setup your `osimage/linuximage` tables with new test image name, `osvers`, `osarch`, and paths to all the files for building and installing the node. So using the above generated `rhels7.3-ppc64le-statelike-compute` as an example, I am going to create my own image. The value for the provisioning method attribute is `osimage` in my example.:

```
mkdef rhels7.3-custom-statelike -u profile=compute provmethod=statelike
```

Check your setup:

```
lsdef -t osimage rhels7.3-custom-statelike
```

Customize the paths to your `pkglist`, `syncfile`, etc to the `osimage` definition, that you require. Note, if you modify the files on the `/opt/xcat/share/...` path then copy to the appropriate `/install/custom/...` path. Remember all files must be under `/install` if using hierarchy (service nodes).

Copy the sample `*list` files and modify as needed:

```
mkdir -p /install/test/netboot/rh
cp -p /opt/xcat/share/xcat/netboot/rh/compute.rhels7.ppc64le.pkglist \
/install/test/netboot/rh/compute.rhels7.ppc64le.pkglist
cp -p /opt/xcat/share/xcat/netboot/rh/compute.exlist \
/install/test/netboot/rh/compute.exlist

chdef -t osimage -o rhels7.3-custom-statelike \
  pkgdir=/install/rhels7.3/ppc64le \
  pkglist=/install/test/netboot/rh/compute.rhels7.ppc64le.pkglist \
  exlist=/install/test/netboot/rh/compute.exlist \
  rootimgdir=/install/test/netboot/rh/ppc64le/compute
```

## Setup pkglists

In the above example, you have defined your pkglist to be in `/install/test/netboot/rh/compute.rhels7.ppc64le.pkglist`.

Edit `compute.rhels7.ppc64le.pkglist` and `compute.exlist` as needed.

```
vi /install/test/netboot/rh/compute.rhels7.ppc64le.pkglist
vi /install/test/netboot/rh/compute.exlist
```

Make sure nothing is excluded in `compute.exlist` that you need.

## Install other specific packages

Make the directory to hold additional rpms to install on the compute node.

```
mkdir -p /install/test/post/otherpkgs/rh/ppc64le
```

Now copy all the additional OS rpms you want to install into `/install/test/post/otherpkgs/rh/ppc64le`.

At first you need to create one text file which contains the complete list of files to include in the repository. The name of the text file is `rpms.list` and must be in `/install/test/post/otherpkgs/rh/ppc64le` directory. Create `rpms.list`:

```
cd /install/test/post/otherpkgs/rh/ppc64le
ls *.rpm > rpms.list
```

Then, run the following command to create the repodata for the newly-added packages:

```
createrepo -i rpms.list /install/test/post/otherpkgs/rh/ppc64le
```

The `createrepo` command with `-i rpms.list` option will create the repository for the rpm packages listed in the `rpms.list` file. It won't destroy or affect the rpm packages that are in the same directory, but have been included into another repository.

Or, if you create a sub-directory to contain the rpm packages, for example, named `other` in `/install/test/post/otherpkgs/rh/ppc64le`. Run the following command to create repodata for the directory `/install/test/post/otherpkgs/rh/ppc64le`.

```
createrepo /install/post/otherpkgs/<os>/<arch>/**other**
```

**Note:** Replace `other` with your real directory name.

Define the location of of your otherpkgs in your osimage:

```
chdef -t osimage -o rhels7.3-custom-statelite \
otherpkgdir=/install/test/post/otherpkgs/rh/ppc64le \
otherpkglist=/install/test/netboot/rh/compute.otherpkgs.pkglist
```

There are examples under `/opt/xcat/share/xcat/netboot/<platform>` of typical `*otherpkgs.pkglist` files that can be used as an example of the format.

## Set up Post scripts for statelite

The rules to create post install scripts for statelite image is the same as the rules for stateless/diskless install images.

There are two kinds of postscripts for statelite (also for stateless/diskless).

The first kind of postscript is executed at genimage time, it is executed again the image itself on the MN . It was setup in The postinstall file section before the image was generated.

The second kind of postscript is the script that runs on the node during node deployment time. During init.d timeframe, /etc/init.d/gettyset calls /opt/xcat/xcatdsklspost that is in the image. This script uses wget to get all the postscripts under mn:/install/postscripts and copy them to the /xcatpost directory on the node. It uses openssl or stunnel to connect to the xcatd on the mn to get all the postscript names for the node from the postscripts table. It then runs the postscripts for the node.

## Setting up postinstall files (optional)

Using postinstall files is optional. There are some examples shipped in /opt/xcat/share/xcat/netboot/<platform>.

If you define a postinstall file to be used by genimage, then

```
chdef -t osimage -o rhels7.3-custom-statelite postinstall=<your postinstall file path>.
```

## Generate the image

Run the following command to generate the image based on your osimage named rhels7.3-custom-statelite. Adjust your genimage parameters to your architecture and network settings. See man genimage.

```
genimage rhels7.3-custom-statelite
```

The genimage will create a default /etc/fstab in the image, if you want to change the defaults, on the management node, edit fstab in the image:

```
cd /install/netboot/rhels7/ppc64le/compute/rootimg/etc
cp fstab fstab.ORIG
vi fstab
```

Note: adding /tmp and /var/tmp to /etc/fstab is optional, most installations can simply use /. It was documented her to show that you can restrict the size of filesystems, if you need to. The indicated values are just an example, and you may need much bigger filesystems, if running applications like OpenMPI.

## Pack the image

Execute liteimg

```
liteimg rhels7.3-custom-statelite
```

## Boot the statelite node

Execute `rinstall`

```
rinstall node1 osimage=rhels7.3-custom-statelite
```

## Switch to the RAMdisk based solution

It is optional, if you want to use RAMdisk-based solution, follow this section.

## Set rootfstype

If you want the node to boot with a RAMdisk-based image instead of the NFS-base image, set the `rootfstype` attribute for the `osimage` to `ramdisk`. For example:

```
chdef -t osimage -o rhels7.3-custom-statelite rootfstype=ramdisk
```

## Run liteimg command

The `liteimg` command will modify your statelite image (the image that `genimage` just created) by creating a series of links. Once you are satisfied with your image contains what you want it to, run `liteimg <osimagename>`:

```
liteimg rhels7.3-custom-statelite
```

For files with link options, the `liteimg` command creates two levels of indirection, so that files can be modified while in their image state as well as during runtime. For example, a file like `$imageroot/etc/ntp.conf` with link option in the `litefile` table, will have the following operations done to it:

In our case `$imageroot` is `/install/netboot/rhels5.3/x86_64/compute/rootimg`

The `liteimg` script, for example, does the following to create the two levels of indirection.

```
mkdir -p $imageroot/.default/etc
mkdir -p $imageroot/.statelite/tmpfs/etc
mv $imgroot/etc/ntp.conf $imgroot/.default/etc
cd $imgroot/.statelite/tmpfs/etc
ln -sf ../../../../default/etc/ntp.conf .
cd $imgroot/etc
ln -sf ../.statelite/tmpfs/etc/ntp.conf .
```

When finished, the original file will reside in `$imgroot/.default/etc/ntp.conf`. `$imgroot/etc/ntp.conf` will link to `$imgroot/.statelite/tmpfs/etc/ntp.conf` which will in turn link to `$imgroot/.default/etc/ntp.conf`.

But for files without link options, the `liteimg` command only creates clones in `$imageroot/.default/` directory, when the node is booting up, the `mount` command with `--bind` option will get the corresponding files from the `litetree` places or `.default` directory to the `sysroot` directory.

**Note:** If you make any changes to your `litefile` table after running `liteimg` then you will need to rerun `liteimg` again. This is because files and directories need to have the two levels of redirects created.

## Boot the statelite node

Make sure you have set up all the attributes in your node definitions correctly following the node installation instructions corresponding to your hardware:

You can now deploy the node by running the following commands:

```
rinstall <noderange>
```

You can then use `rcons` or `wcons` to watch the node boot up.

## Adding/updating software and files for the running nodes

### Make changes to the files which configured in the litefile table

During the preparation or booting of node against statelite mode, there are specific processes to handle the files which configured in the litefile table. The following operations need to be done after made changes to the statelite files.

1. Run `liteimg` against the osimage and reboot the node : Added, removed or changed the entries in the litefile table.
2. Reboot the node :
  - Changed the location directory in the litetree table.
  - Changed the location directory in the statelite table.
  - Changed, removed the original files in the location of litetree or statelite table.

Note: Thing should not do:

- When there are node running on the nfs-based statelite osimage, do not run the packimage against this osimage.

### Make changes to the common files

Because most of system files for the nodes are NFS mounted on the Management Node with read-only option, installing or updating software and files should be done to the image. The image is located under `/install/netboot/<os>/<arch>/<profile>/rootimg` directory.

To install or update an rpm, do the following:

- Install the rpm package into rootimg

```
rpm --root /install/netboot/<os>/<arch>/<profile>/rootimg -ivh rpm_name
```

- Restart the software application on the nodes

```
xdsh <noderange> <restart_this_software_command>
```

It is recommended to follow the section (Adding third party software) to add the new rpm to the `otherpkgs.pkglist` file, so that the rpm will get installed into the new image next time the image is rebuilt.

Note: The newly added rpms are not shown when running `rpm -qa` on the nodes although the rpm is installed. It will shown next time the node is rebooted.

To create or update a file for the nodes, just modify the file in the image and restart any application that uses the file.

For the ramdisk-based node, you need to reboot the node to take the changes.

## Hierarchy Support

In the `statelite` environment, the service node needs to provide NFS service for the compute node with `statelite`, the service nodes must to be setup with diskfull installation.

### Setup the diskfull service node

1. Setup one diskfull service node at first.
2. Since `statelite` is a kind of NFS-hybrid method, you should remove the `installloc` attribute in the site table. This makes sure that the service node does not mount the `/install` directory from the management node on the service node.

### Generate the statelite image

To generate the `statelite` image for your own profile follow instructions in *Customize your statelite osimage*.

NOTE: if the NFS directories defined in the `litetree` table are on the service node, it is better to setup the NFS directories in the service node following the chapter.

### Sync the /install directory

The command `prsync` is used to sync the `/install` directory to the service nodes.

Run the following:

```
cd /  
prsync install <sn>:/
```

<sn> is the hostname of the service node you defined.

Since the `prsync` command will sync all the contents in the `/install` directory to the service nodes, the first time will take a long time. But after the first time, it will take very short time to sync.

NOTE: if you make any changes in the `/install` directory on the management node, and the changes can affect the `statelite` image, you need to sync the `/install` directory to the service node again.

### Set the boot state to statelite

You can now deploy the node:

```
rinstall <noderange> osimage=rhel5.3-x86_64-statelite-compute
```

This will create the necessary files in `/tftpboot` for the node to boot correctly.

## Advanced features

### Both directory and its child items coexist in litefile table

As described in the above chapters, we can add the files/directories to litefile table. Sometimes, it is necessary to put one directory and also its child item(s) into the litefile table. Due to the implementation of the statelite on Linux, some scenarios works, but some doesn't work.

Here are some examples of both directory and its child items coexisting:

Both the parent directory and the child file coexist:

```
"ALL", "/root/testblank/", , ,
"ALL", "/root/testblank/tempfschild", "tempfs", ,
```

One more complex example:

```
"ALL", "/root/", , ,
"ALL", "/root/testblank/tempfschild", "tempfs", ,
```

Another more complex example, but we don't intend to support such one scenario:

```
"ALL", "/root/", , ,
"ALL", "/root/testblank/", , ,
"ALL", "/root/testblank/tempfschild", "tempfs", ,
```

For example, in scenario 1, the parent is /root/testblank/, and the child is /root/testblank/tempfschild. In scenario 2, the parent is /root/, and the child is /root/testblank/tempfschild.

In order to describe the hierarchy scenarios we can use , P to denote parent, and C to denote child.

Option	Example	Remarks
P:tmpfs	“ALL”,,”/root/testblank/”,,, “ALL”,,”/root/testblank/tmpfschild”,,”l	Both the parent and the child are mounted to tmpfs on the booted node following their respective options. Only the parent are mounted to the local file system.
P:tmpfs C:persistent	“ALL”,,”/root/testblank/”,,, “ALL”,,”/root/testblank/testpersfile”,,”l	Both parent and child are mounted to tmpfs on the booted node following their respective options. Only the parent is mounted to the local file system.
P:persistent C:tmpfs	“ALL”,,”/root/testblank/”,,”persistent”, “ALL”,,”/root/testblank/tmpfschild”,,”l	Not permitted now. But plan to support it.
P:persistent C:persistent	“ALL”,,”/root/testblank/”,,”persistent”, “ALL”,,”/root/testblank/testpersfile”,,”l	Both parent and child are mounted to tmpfs on the booted node following their respective options. Only the parent is mounted to local file system.
P:ro C:any		Not permitted
P:tmpfs C:ro		Both parent and child are mounted to tmpfs on the booted node following their respective options. Only the parent is mounted to local file system.
P:tmpfs C:con		Both parent and child are mounted to tmpfs on the booted node following their respective options. Only the parent is mounted to local file system.
P:link C:link	“ALL”,,”/root/testlink/”,,”link”,, “ALL”,,”/root/testlink/testlinkchild”,,”l	Both parent and child are created in tmpfs on the booted node following their respective options; there’s only one symbolic link of the parent is created in the local file system.
P: link C: link, persistent	“ALL”,,”/root/testlinkpers/”,,”link”,, “ALL”,,”/root/testlink/testlinkchild”,, “link,persistent”	Both parent and child are created in tmpfs on the booted node following their respective options; there’s only one symbolic link of the parent is created in the local file system.
<b>P:link,</b> persistent C: link	“ALL”,,”/root/testlinkpers/”,,”link,pers: “ALL”,,”/root/testlink/testlinkchild”,,”l	NOT permitted
<b>P:link,</b> persistent <b>C:link,</b> persistent	“ALL”,,”/root/testlinkpers/”,,”link,pers: “ALL”,,”/root/testlink	Both parent and child are created in tmpfs on the booted node following “link,persistent” way; there’s only one symbolic link of the parent is created in the local file system.
P:link C:link,ro	“ALL”,,”/root/testlink/”,,”link”,, “ALL”,,”/root/testlink/testlinkro”,,”link	Both parent and child are created in tmpfs on the booted node, there’s only one symbolic link of the parent is created in the local file system.
P:link C:link,con	“ALL”,,”/root/testlink/”,,”link”,, “ALL”,,”/root/testlink/testlinkconchild	Both parent and child are created in tmpfs on the booted node, there’s only one symbolic link of the parent



## litetree table

The litetree table controls where the initial content of the files in the litetree table come from, and the long term content of the ro files. When a node boots up in statelite mode, it will by default copy all of its tmpfs files from the `/.default` directory of the root image, so there is not requirement to setup a litetree table. If you decide that you want some of the files pulled from different locations that are different per node, you can use this table.

See litetree man page for description of attributes.

For example, a user may have two directories with a different `/etc/motd` that should be used for nodes in two locations:

```
10.0.0.1:/syncdirs/newyork-590Madison/rhels5.4/x86_64/compute/etc/motd
10.0.0.1:/syncdirs/shanghai-11foo/rhels5.4/x86_64/compute/etc/motd
```

You can specify this in one row in the litetree table:

```
1,,10.0.0.1:/syncdirs/$nodepos.room/$nodetype.os/$nodetype.arch/$nodetype.profile
```

When each statelite node boots, the variables in the litetree table will be substituted with the values for that node to locate the correct directory to use. Assuming that `/etc/motd` was specified in the litetree table, it will be searched for in all of the directories specified in the litetree table and found in this one.

You may also want to look by default into directories containing the node name first:

```
$noderes.nfsserver:/syncdirs/$node
```

The litetree prioritizes where node files are found. The first field is the priority. The second field is the image name (ALL for all images) and the final field is the mount point.

Our example is as follows:

```
1,$noderes.nfsserver:/statelite/$node
2,,cnfs:/gpfs/dallas/
```

The two directories `/statelite/$node` on the node's `$noderes.nfsserver` and the `/gpfs/dallas` on the node `cnfs` contain root tree structures that are sparsely populated with files that we want to place in those nodes. If files are not found in the first directory, it goes to the next directory. If none of the files can be found in the litetree hierarchy, then they are searched for in `/.default` on the local image.

## Installing a new Kernel in the statelite image

Obtain you new kernel and kernel modules on the MN, for example here we have a new SLES kernel.

1. Copy the kernel into `/boot` :

```
cp **vmlinux-2.6.32.10-0.5-ppc64**/boot
```

2. Copy the kernel modules into `/lib/modules/<new kernel directory>`

```
/lib/modules # ls -al
total 16
drwxr-xr-x 4 root root 4096 Apr 19 10:39 .
drwxr-xr-x 17 root root 4096 Apr 13 08:39 ..
drwxr-xr-x 3 root root 4096 Apr 13 08:51 2.6.32.10-0.4-ppc64
**drwxr-xr-x 4 root root 4096 Apr 19 10:12 2.6.32.10-0.5-ppc64**
```

3. Run `genimage` to update the statelite image with the new kernel

```
genimage -k 2.6.32.10-0.5-ppc64 <osimage_name>
```

4. Then after a `nodeset` command and `netbooti`, shows the new kernel:

```
uname -a
```

## Enabling the localdisk option

---

**Note:** You can skip this section if not using the `localdisk` option in your `litefile` table.

---

### Define how to partition the local disk

When a node is deployed, the local hard disk needs to be partitioned and formatted before it can be used. This section explains how provide a configuration file that tells xCAT to partition a local disk and make it ready to use for the directories listed in the `litefile` table.

The configuration file needs to be specified in the `partitionfile` attribute of the `osimage` definition. The configuration file includes several sections:

- Global parameters to control enabling or disabling the function
- `[disk]` section to control the partitioning of the disk
- `[localspace]` section to control which partition will be used to store the `localdisk` directories listed in the `litefile` table
- `[swap space]` section to control the enablement of the swap space for the node.

An example `localdisk` configuration file:

```
enable=yes
enablepart=no

[disk]
dev=/dev/sda
clear=yes
parts=10,20,30

[disk]
dev=/dev/sdb
clear=yes
parts=100M-200M,1G-2G

[disk]
dev=/dev/sdc
ptype=gpt
clear=yes
parts=10,20,30

[localspace]
```

(continues on next page)

(continued from previous page)

```
dev=/dev/sda1
fstype=ext4

[swapspace]
dev=/dev/sda2
```

The two global parameters `enable` and `enablepart` can be used to control the enabling/disabling of the functions:

- `enable`: The `localdisk` feature only works when `enable` is set to `yes`. If it is set to `no`, the `localdisk` configuration will not be run.
- `enablepart`: The partition action (refer to the `[disk]` section) will be run only when `enablepart=yes`.

The `[disk]` section is used to configure how to partition a hard disk:

- `dev`: The path of the device file.
- `clear`: If set to `yes` it will clear all the existing partitions on this disk.
- `ptype`: The partition table type of the disk. For example, `msdos` or `gpt`, and `msdos` is the default.
- `fstype`: The file system type for the new created partitions. `ext4` is the default.
- `parts`: A comma separated list of space ranges, one for each partition that will be created on the device. The valid format for each space range is `<startpoint>-<endpoint>` or `<percentage of the disk>`. For example, you could set it to `100M-10G` or `50`. If set to `50`, 50% of the disk space will be assigned to that partition.

The `[localspace]` section is used to specify which partition will be used as local storage for the node.

- `dev`: The path of the partition.
- `fstype`: The file system type on the partition.

the `[swapspace]` section is used to configure the swap space for the statelite node.

- `dev`: The path of the partition file which will be used as the swap space.

To enable the local disk capability, create the configuration file (for example in `/install/custom`) and set the path in the `partitionfile` attribute for the `osimage`:

```
chdef -t osimage <osimage> partitionfile=/install/custom/cfglocaldisk
```

Now all nodes that use this `osimage` (i.e. have their `provmethod` attribute set to this `osimage` definition name), will have its local disk configured.

## Configure the files in the litefile table

For the files/directories to store on the local disk, add an entry in the `litefile` table:

```
"ALL", "/tmp/", "localdisk", ,
```

**Note:** you do not need to specify the swap space in the `litefile` table. Just putting it in the `partitionfile` config file is enough.

Add an entry in policy table to permit the running of the `getpartition` command from the node

```
chtab priority=7.1 policy.commands=getpartition policy.rule=allow
```

Run `genimage` and `packimage` for the `osimage`

### If Using the RAMdisk-based Image

If you want to use the local disk option with a RAMdisk-based image, remember to follow the instructions in [Switch to the RAMdisk based solution](#).

If your reason for using a RAMdisk image is to avoid compute node runtime dependencies on the service node or management node, then the only entries you should have in the `litefile` table should be files/dirs that use the `localdisk` option.

### Debugging techniques

When a node boots up in `statelite` mode, there is a script that runs called `statelite` that is in the root directory of `$imageroot/etc/init.d/statelite`. This script is not run as part of the `rc` scripts, but as part of the pre-switch root environment. Thus, all the linking is done in this script. There is a `set -x` near the top of the file. You can uncomment it and see what the script runs. You will then see lots of `mkdirs` and links on the console.

You can also set the machine to shell. Just add the word `shell` on the end of the `pxeboot` file of the node in the `append` line. This will make the init script in the `initramfs` pause 3 times before doing a `switch_root`.

When all the files are linked they are logged in `/.statelite/statelite.log` on the node. You can get into the node after it has booted and look in the `/.statelite` directory.

### Using Updatenode

#### Introduction

After initial node deployment, you may need to make changes/updates to your nodes. The `updatenode` command is for this purpose. It allows you to add or modify the followings on your nodes:

1. Add additional software
2. Re-run postscripts or run additional postscripts
3. Synchronize new/updated configuration files
4. Update `ssh` keys and `xCAT` certificates

Each of these will be explained in the document. The basic way to use `updatenode` is to set the definition of nodes on the management node the way you want it and then run `updatenode` to push those changes out to the actual nodes. Using options to the command, you can control which of the above categories `updatenode` pushes out to the nodes.

Most of what is described in this document applies to **stateful** and **stateless** nodes. In addition to the information in this document, check out the `updatenode` man page.

## Add Additional Software

The packages that will be installed on the node are stored in the packages list files. There are **two kinds of package list files**:

1. The **package list file** contains the names of the packages that come from the os distro. They are stored in **.pkglist** file.
2. The **other package list file** contains the names of the packages that do **NOT** come from the os distro. They are stored in **.otherpkgs.pkglist** file.

## Installing Additional OS Distro Packages

For packages from the OS distro, add the new package names (without the version number) in the .pkglist file. If you have newer updates to some of your operating system packages that you would like to apply to your OS image, you can place them in another directory, and add that directory to your osimage pkgdir attribute. How to add additional OS distro packages, go to [Install Additional OS Packages for RHEL and SLES](#)

Note: If the objective node is not installed by xCAT, make sure the correct osimage pkgdir attribute so that you could get the correct repository data.

## Install Additional non-OS Packages

If you have additional packages (packages not in the distro) that you also want installed, make a directory to hold them, create a list of the packages you want installed, and add that information to the osimage definition. How to add Additional Other Packages, go to [Install Additional Other Packages for RHEL and SLES](#)

## Update Nodes

Run the updatenode command to push the new software to the nodes:

```
updatenode <noderange> -S
```

The -S flag updates the nodes with all the new or updated packages specified in both .pkglist and .otherpkgs.pkglist.

If you have a configuration script that is necessary to configure the new software, then instead run:

```
cp myconfigscript /install/postscripts/  
chdef -p -t compute postbootscripts=myconfigscript  
updatenode <noderange> ospkgs,otherpkgs,myconfigscript
```

The next time you re-install these nodes, the additional software will be automatically installed.

**If you update stateless nodes, you must also do this next step**, otherwise the next time you reboot the stateless nodes, the new software won't be on the nodes. Run genimage and packimage to install the extra rpms into the image:

```
genimage <osimage>  
packimage <osimage>
```

## Update the delta changes in Sysclone environment

Updatenode can also be used in Sysclone environment to push delta changes to target node. After capturing the delta changes from the golden client to management node, just run below command to push delta changes to target nodes. See Sysclone environment related section: *Update Nodes Later On* for more information.

```
updatenode <targetnoderange> -S
```

## Rerun Postscripts or Run Additional Postscripts

You can use the updatenode command to perform the following functions after the nodes are up and running:

- Rerun postscripts defined in the postscripts table.
- Run any additional postscript one time.

Go to *Using Postscript* to see how to configure postscript.

Go to *Using Prescript* to see how to configure prepostscript.

To rerun all the postscripts for the nodes. (In general, xCAT postscripts are structured such that it is not harmful to run them multiple times.)

```
updatenode <noderange> -P
```

To rerun just the syslog postscript for the nodes:

```
updatenode <noderange> -P syslog
```

To run a list of your own postscripts, make sure the scripts are copied to /install/postscripts directory, then:

```
updatenode <noderange> -P "script1,script2"
```

If you need to, you can also pass arguments to your scripts:

```
updatenode <noderange> -P "script1 p1 p2,script2"
```

mypostscript template for updatenode

You can customize what attributes you want made available to the postscript, using the shipped mypostscript.tmpl file *Using the mypostscript template*.

## Synchronize new/updated configuration files

### Setting up syncfile

Use instructions in doc: *The synclist file*.

## Syncfiles to the nodes

After compute node is installed, you would like to sync files to the nodes:

```
updatenode <noderange> -F
```

With the `updatenode` command the `syncfiles` postscript cannot be used to sync files to the nodes. Therefore, if you run `updatenode <noderange> -P syncfiles`, nothing will be done. A message will be logged that you must use `updatenode <noderange> -F` to sync files.

## Update the ssh Keys and Credentials on the Nodes

If after node deployment, the ssh keys or xCAT ssl credentials become corrupted, xCAT provides a way to quickly fix the keys and credentials on your service and compute nodes:

```
updatenode <noderange> -K
```

Note: this option can't be used with any of the other `updatenode` options.

## Appendix : Debugging Tips

Internally `updatenode` command uses the `xdsh` in the following ways:

Linux: `xdsh <noderange> -e /install/postscripts/xcatdsklspost -m <server> <scripts>`

where `<scripts>` is a comma separated postscript like `ospkgs,otherpkgs` etc.

- `wget` is used in `xcatdsklspost/xcataixpost` to get all the postscripts from the `<server>` to the node. You can check `/tmp/wget.log` file on the node to see if `wget` was successful or not. You need to make sure the `/xcatpost` directory has enough space to hold the postscripts.
- A file called `/xcatpost/mypostscript (Linux)` is created on the node which contains the environmental variables and scripts to be run. Make sure this file exists and it contains correct info. You can also run this file on the node manually to debug.
- For `ospkgs/otherpkgs`, if `/install` is not mounted on the `<server>`, it will download all the rpms from the `<server>` to the node using `wget`. Make sure `/tmp` and `/xcatpost` have enough space to hold the rpms and check `/tmp/wget.log` for errors.
- For `ospkgs/otherpkgs`, If `zypper` or `yum` is installed on the node, it will be used the command to install the rpms. Make sure to run `createrepo` on the source directory on the `<server>` every time a rpm is added or removed. Otherwise, the `rpm` command will be used, in this case, make sure all the necessary depended rpms are copied in the same source directory.
- You can append `-x` on the first line of `ospkgs/otherpkgs` to get more debug info.

## Parallel Commands

xCAT provides a set of commands that can run common remote commands (`ssh`, `scp`, `rsh`, `rcp`, `rsync`, `ping`, `cons`) in parallel on xCAT managed nodes. The xCAT commands will format the output making the results easier to parse and help administrators manage large clusters.

The following commands are provided:

- `pcons` - runs a command on the noderange using the out-of-band console
- `pping` - parallel ping
- `ppping` - parallel ping between nodes in a cluster
- `prsync` - parallel rsync
- `pscp` - parallel remote copy ( supports `scp` and not hierarchy)
- `psh` - parallel remote shell ( supports `ssh` and not hierarchy)
- `pasu` - parallel ASU utility
- `xdcp` - concurrently copies files to and from multiple nodes. ( `scp/rcp` and hierarchy)
- `xdsh` - concurrently runs commands on multiple nodes. ( supports `ssh/rsh` and hierarchy)
- `xdshbak` - formats the output of the `xdsh` command
- `xcoll` - Formats command output of the `psh`, `xdsh`, `rinv` command

## Examples for `xdsh`

- To set up the SSH keys for root on node1, run as root:

```
xdsh node1 -K
```

- To run the `ps -ef` command on node targets node1 and node2, enter:

```
xdsh node1,node2 "ps -ef"
```

- To run the `ps` command on node targets node1 and run the remote command with the `-v` and `-t` flag, enter:

```
xdsh node1,node2 -o "-v -t" ps
```

- To execute the commands contained in myfile in the XCAT context on several node targets, with a fanout of 1, enter:

```
xdsh node1,node2 -f 1 -e myfile
```

- To run the `ps` command on node1 and ignore all the dsh environment variable except the `DSH_NODE_OPTS`, enter:

```
xdsh node1 -X `DSH_NODE_OPTS` ps
```

- To run on Linux, the `xdsh` command `dpkg -l| grep vim` on the node ubuntu diskless image, enter:

```
xdsh -i /install/netboot/ubuntu14.04.2/ppc64el/compute/rooting "dpkg -l|grep vim"
```

- To run `xdsh` with the non-root userid “user1” that has been setup as an xCAT userid and with `sudo` on node1 and node2 to run as root, do the following, see [Granting users xCAT privileges](#):



```
xdsh node1,node2 --sudo -l user1 "cat /etc/passwd"
```

## Examples for xdcp

- To copy the /etc/hosts file from all nodes in the cluster to the /tmp/hosts.dir directory on the local host, enter:

```
xdcp all -P /etc/hosts /tmp/hosts.dir
```

A suffix specifying the name of the target is appended to each file name. The contents of the /tmp/hosts.dir directory are similar to:

```
hosts._node1  hosts._node4  hosts._node7
hosts._node2  hosts._node5  hosts._node8
hosts._node3  hosts._node6
```

- To copy /localnode/smallfile and /tmp/bigfile to /tmp on node1 using rsync and input -t flag to rsync, enter:

```
xdcp node1 -r /usr/bin/rsync -o "-t" /localnode/smallfile /tmp/bigfile /tmp
```

- To copy the /etc/hosts file from the local host to all the nodes in the cluster, enter:

```
xdcp all /etc/hosts /etc/hosts
```

- To rsync the /etc/hosts file to your compute nodes:

Create a rsync file /tmp/myrsync, with this line:

```
/etc/hosts -> /etc/hosts
or
/etc/hosts -> /etc/ (last / is required)
```

Run:

```
xdcp compute -F /tmp/myrsync
```

- To rsync the /etc/file1 and file2 to your compute nodes and rename to filex and filey:

Create a rsync file /tmp/myrsync, with these line:

```
/etc/file1 -> /etc/filex
/etc/file2 -> /etc/filey
```

Run:

```
xdcp compute -F /tmp/myrsync to update the Compute Nodes
```

- To rsync files in the Linux image at /install/netboot/ubuntu14.04.2/ppc64el/compute/rooting on the MN:

Create a rsync file /tmp/myrsync, with this line:

```
/etc/hosts /etc/passwd -> /etc
```

Run:

```
xdcp -i /install/netboot/ubuntu14.04.2/ppc64el/compute/rootimg -F /tmp/myrsync
```

## Virtual Machines

xCAT supports the following virtualization infrastructures:

### Kernel-based Virtual Machine (KVM):

A full virtualization solution for Enterprise Linux distributions, known as the *de facto* open source virtualization mechanism and currently used by many software companies.

### IBM PowerKVM:

A product that leverages the Power resilience and performance with the openness of KVM, which provides several advantages:

- Higher workload consolidation with processors overcommitment and memory sharing
- Dynamic addition and removal of virtual devices
- Microthreading scheduling granularity
- Integration with **IBM PowerVC** and **OpenStack**
- Simplified management using open source software
- Avoids vendor lock-in
- Uses POWER8 hardware features, such as SMT8 and microthreading

The xCAT based KVM solution offers users the ability to:

- provision the hypervisor on bare metal nodes
- provision virtual machines with the any OS supported in xCAT
- migrate virtual machines to different hosts
- install copy on write instances of virtual machines
- clone virtual machines

This section introduces the steps of management node preparation, hypervisor setup and virtual machine management, and presents some typical problems and solutions on xCAT kvm support.

## Set Up the Management Node for KVM

### Install the kvm related packages

Additional packages need to be installed on the management node for kvm support.

Please make sure the following packages have been installed on the management node, if not, install them manually.

perl-Sys-Virt

## Set Up the kvm storage directory on the management node(optional)

It is a recommended configuration to create a shared file system for virtual machines hosting. The shared file system, usually on a SAN, NAS or GPFS, is shared among KVM hypervisors, which simplifies VM migration from one hypervisor to another with xCAT.

The easiest shared file system is /install directory on the management node, it can be shared among hypervisors via NFS. Please refer to the following steps :

- Create a directory to store the virtual disk files

```
mkdir -p /install/vms
```

- export the storage directory

```
echo "/install/vms *(rw,no_root_squash,sync,fsid=0)" >> /etc/exports
exportfs -r
```

**Note:** make sure the root permission is turned on for nfs clients (i.e. use the no\_root\_squash option). Otherwise, the virtual disk file can not work.

## Install and Configure Hypervisor

### Provision Hypervisor

#### [PowerKVM]

Obtain a PowerKVM ISO and create PowerKVM osimages with it:

```
copycds ibm-powerkvm-3.1.0.0-39.0-ppc64le-gold-201511041419.iso
```

The following PowerKVM osimage will be created

```
# lsdef -t osimage -o pkvm3.1-ppc64le-install-compute
Object name: pkvm3.1-ppc64le-install-compute
  imagetype=linux
  osarch=ppc64le
  osdistrname=pkvm3.1-ppc64le
  osname=Linux
  osvers=pkvm3.1
  otherpkgdir=/install/post/otherpkgs/pkvm3.1/ppc64le
  pkgdir=/install/pkvm3.1/ppc64le
  profile=compute
  provmethod=install
  template=/opt/xcat/share/xcat/install/pkvm/compute.pkvm3.ppc64le.
→ tmp1
```

## [RHEV]

Red Hat Virtualization (formally known as RHEV or Red Hat Enterprise Virtualization) is a virtualization solution provided by Red Hat.

At the time of this writing there is no RHEV-H prebuilt hypervisor image on Power LE. The method for creating a Red Hat Hypervisor on Power LE is to first install RHEL and apply the KVM support on top with the provided RPMs.

Obtain and download the RHEV RPM packages from the Red Hat download site.

- Management-Agent-Power-7
- Power\_Tools-7

In the following example, the RPMs are downloaded to `/install/post/otherpkgs/rhels7.3/ppc64le/RHEV4/4.0-GA`

- Create a yum repository for the downloaded RPMs

```
createrepo /install/post/otherpkgs/rhels7.3/ppc64le/RHEV4/4.0-GA
```

- Create new osimage definition based on an existing RHEL7 osimage definition

```
mkdef -t osimage -o rhels7.3-ppc64le-RHEV4-install-compute \
--template rhels7.3-ppc64le-install-compute
```

- Modify otherpkgdir attribute to point to the package directory with downloaded RPMs

```
chdef -t osimage rhels7.3-ppc64le-RHEV4-install-compute \
otherpkgdir=/install/post/otherpkgs/rhels7.3/ppc64le/RHEV4/4.0-GA
```

- Create a new package list file `/install/custom/rhels7.3/ppc64le/rhelv4.pkglist` to include necessary packages provided from the OS.

```
#INCLUDE:/opt/xcat/share/xcat/install/rh/compute.rhels7.pkglist#
yum
libvirt
screen
bridge-utils
```

- Modify pkglist attribute to point to the package list file from the step above

```
chdef -t osimage rhels7.3-snap3-ppc64le-RHEV4-install-compute \
pkglist=/install/custom/rhels7.3/ppc64le/rhelv4.pkglist
```

- Create a new package list file `/install/custom/rhels7.3/ppc64le/rhev4.otherpkgs.pkglist` to list required packages

```
qemu-kvm-rhev
qemu-kvm-tools-rhev
virt-manager-common
virt-install
```

- Modify otherpkglist attribute to point to the package list file from the step above

```
chdef -t osimage rhels7.3-snap3-ppc64le-RHEV4-install-compute \
  otherpkglist=/install/custom/rhels7.3/ppc64le/rhev4.otherpkgs.
↪pkglist
```

- The RHEV osimage should look similar to:

```
Object name: rhels7.3-ppc64le-RHEV4-install-compute
  imagetype=linux
  osarch=ppc64le
  osdistrname=rhels7.3-ppc64le
  osname=Linux
  osvers=rhels7.3
  otherpkgdir=/install/post/otherpkgs/rhels7.3/ppc64le/RHEV4/4.0-
↪GA
  otherpkglist=/install/custom/rhels7.3/ppc64le/rhev4.otherpkgs.
↪pkglist
  pkgdir=/install/rhels7.3/ppc64le
  pkglist=/install/custom/rhels7.3/ppc64le/rhelv4.pkglist
  profile=compute
  provmethod=install
  template=/opt/xcat/share/xcat/install/rh/compute.rhels7.tmpl
```

**Note:** If diskless RHEV osimage is needed, `localdisk` option can be used to preserve VMs over RHEV hypervisor reprovision. For instructions see [Enabling the localdisk option](#). Set `/var/lib/libvirt/` directory where by default VM images are stored to `localdisk`.

#### 1. Customize the hypervisor node definition to create network bridge

xCAT ships a postscript **xHRM** to create a network bridge on kvm host during installation/netbooting. Specify the **xHRM** with appropriate parameters in **postscripts** attribute. For example:

- To create a bridge named ‘br0’ against the installation network device specified by **installnic**:

```
chdef kvmhost1 -p postscripts="xHRM bridgeprereq br0"
```

- To create a bridge with default name ‘default’ against the installation network device specified by **installnic**:

```
chdef kvmhost1 -p postscripts="xHRM bridgeprereq"
```

- To create a bridge named ‘br0’ against the network device ‘eth0’:

```
chdef kvmhost1 -p postscripts="xHRM bridgeprereq eth0:br0"
```

**Note:** The network bridge name you use should not be the virtual bridges (vbrX) created by libvirt installation<sup>1</sup>.

#### 1. Customize the hypervisor node definition to mount the shared kvm storage directory on management node (**optional**)

If the shared kvm storage directory on the management node has been exported, it can be mounted on PowerKVM hypervisor for virtual machines hosting.

<sup>1</sup> Every standard libvirt installation provides NAT based connectivity to virtual machines out of the box using the “virtual bridge” interfaces (virbr0, virbr1, etc) Those will be created by default.

An easy way to do this is to create another postscript named “mountvms” which creates a directory `/install/vms` on hypervisor and then mounts `/install/vms` from the management node, the content of “mountvms” can be:

```
logger -t xcat "Install: setting vms mount in fstab"
mkdir -p /install/vms
echo "$MASTER:/install/vms /install/vms nfs \
    rsize=8192,wsz=8192,timeo=14,intr,nfsvers=2 1 2" >> /etc/fstab
```

Then set the file permission and specify the script in **postscripts** attribute of hypervisor node definition:

```
chmod 755 /install/postscripts/mountvms
chdef kvmhost1 -p postscripts=mountvms
```

## 2. Provision the hypervisor node with the osimage

```
rinstall kvmhost1 osimage=<osimage_name>
```

## Create network bridge on hypervisor

To launch VMs, a network bridge must be created on the KVM hypervisor.

If the hypervisor is provisioned successfully according to the steps described above, a network bridge will be created and attached to a physical interface. This can be checked by running `brctl show` on the hypervisor to show the network bridge information, please make sure a network bridge has been created and configured according to the parameters passed to postscript “xHRM”

```
# brctl show
bridge name      bridge id        STP enabled      interfaces
br0              8000.000000000000 no                eth0
```

If the network bridge is not created or configured successfully, run “xHRM” with **updatenode** on management node to create it manually::

```
updatenode kvmhost1 -P "xHRM bridgeprereq eth0:br0"
```

## Start libvirtd service

Verify **libvirtd** service is running:

```
systemctl status libvirtd
```

If service is not running, it can be started with:

```
systemctl start libvirtd
```

## Manage Virtual Machine (VM)

Now the PowerKVM hypervisor “kvmhost1” is ready, this section introduces the VM management in xCAT, including examples on how to create, remove and clone VMs.

### Create Virtual Machine

#### Create VM Node Definition

Create a virtual machine node object “vm1”, assign it to be a member of group “vm”, its ip is “192.168.0.1”, run makehost to add an entry in /etc/hosts file:

```
mkdef vm1 groups=vm,all
chdef vm1 ip=192.168.0.1
makehosts vm1
```

Update DNS configuration and database:

```
makedns -n
makedns -a
```

### Specify VM attributes

After the VM object is created, several key attributes need to be specified with chdef :

1. the hardware management module, “kvm” for PowerKVM:

```
chdef vm1 mgt=kvm
```

2. the number of virtual cpus in the VM:

```
chdef vm1 vmcpus=2
```

3. the kvm hypervisor of the VM:

```
chdef vm1 vmhost=kvmhost1
```

4. the virtual memory size (in Megabytes):

```
chdef vm1 vmmemory=2048
```

5. Define the virtual network card, it should be set to the bridge “br0” which has been created in the hypervisor. If no bridge is specified, no network device will be created for the VM node “vm1”:

```
chdef vm1 vmnics=br0
```

6. The **vmnicnicmodel** attribute is used to set the type and corresponding driver for the nic. If not set, the default value is ‘virtio’.

```
chdef vm1 vmnicnicmodel=virtio
```

7. Define the storage for the vm1, three types of storage source format are supported.

## A. Create storage on a NFS server

The format is `nfs://<IP_of_NFS_server>/dir`, that means the kvm disk files will be created at `nfs://<IP_of_NFS_server>/dir`:

```
chdef vm1 vmstorage=nfs://<IP_of_NFS_server>/install/vms/
```

## B. Create storage on a device of hypervisor

The format is `phy:/dev/sdb1`:

```
chdef vm1 vmstorage=phy:/dev/sdb1
```

## C. Create storage on a directory of hypervisor

The format is `dir:///var/lib/libvirt/images`:

```
chdef vm1 vmstorage=dir:///var/lib/libvirt/images
```

---

**Note:** The attribute **vmstorage** is only valid for diskful VM node.

---

8. Define the **console** attributes for VM:

```
chdef vm1 serialport=0 serialspeed=115200
```

9. (Optional) For monitoring and access the VM with vnc client, set **vidpassword** value:

```
chtab node=vm1 vm.vidpassword=abc123
```

10. (Optional) For assigning PCI devices to the VM, set **othersettings** value:

```
chtab node=vm1 vm.othersettings="devpassthrough:0000:01:00.2"
```

Or:

```
chtab node=vm1 vm.othersettings="devpassthrough:pci_0000_01_00_2"
```

Take assigning SR-IOV VFs to the VM as an example:

- Use `lspci` to get VFs PCI from hypervisor:

```
lspci | grep -i "Virtual Function"
0000:01:00.1 Infiniband controller: Mellanox Technologies MT27700 Family_
↪ [ConnectX-4 Virtual Function]
0000:01:00.2 Infiniband controller: Mellanox Technologies MT27700 Family_
↪ [ConnectX-4 Virtual Function]
```

- Set the VFs PCI into vm table on MN:

```
chtab node=vm1 vm.othersettings="devpassthrough:0000:01:00.1,0000:01:00.2"
```

11. Set **netboot** attribute

- **[x86\_64]**

```
chdef vm1 netboot=xnba
```



- [PPC64LE]

```
chdef vm1 netboot=grub2
```

Make sure “grub2” had been installed on the management node:

```
#rpm -aq | grep grub2
grub2-xcat-1.0-1.noarch
```

## Make virtual machine

If **vmstorage** is a NFS mounted directory or a device on hypervisor, run

```
mkvm vm1
```

To create the virtual machine “vm1” with 20G hard disk on a hypervisor directory, run

```
mkvm vm1 -s 20G
```

When “vm1” is created successfully, a VM hard disk file with a name like “vm1.sda.qcow2” will be found in the location specified by **vmstorage**. What’s more, the **mac** attribute of “vm1” is set automatically, check it with:

```
lsdef vm1 -i mac
```

Once a VM “vm1” is created, it can be provisioned like any other node in xCAT. The VM node can be powered on by:

```
rpower vm1 on
```

If “vm1” is powered on successfully, the VM status can be obtained by running the following command on management node

```
rpower vm1 status
```

or running the following command on the kvm hypervisor “kvmhost1”

```
#virsh list
Id Name                               State
-----
6 vm1                                running
```

## Monitoring the Virtual Machine

When the VM has been created and powered on, choose one of the following methods to monitor and access it.

- Open the console on kvm hypervisor:

```
virsh console vm1
```

- Use **rcons/wcons** on xCAT management node to open text console:

```
chdef vm1 cons=kvm
makegocons vm1
rcons vm1
```

- Connect to virtual machine through vnc console

In order to connect the virtual machine's vnc server, a new set of credentials need to be generated by running:

```
xcatclient getrvidparms vm1
vm1: method: kvm
vm1: textconsole: /dev/pts/0
vm1: password: JOQTUtn0dU0Bv9o3
vm1: vidproto: vnc
vm1: server: kvmhost1
vm1: vidport: 5900
```

---

**Note:** Now just pick a favorite vnc client to connect the hypervisor, with the password generated by `getrvidparms`. If the vnc client complains “the password is not valid”, the reason might be that the hypervisor and headnode clocks are out of sync! Please try to sync them by running `ntpdate <ntp server>` on both the hypervisor and the headnode.

---

- Use `wvid` on management node

Make sure **firewalld** service is stopped, disable it if not:

```
chkconfig firewalld off
```

or

```
systemctl disable firewalld
```

Then, run `wvid` on MN:

```
wvid vm1
```

- For PowerKVM, **kimchi** on the kvm hypervisor can be used to monitor and access the VM.

## Remove the virtual machine

Remove the VM “vm1” even when it is in “power-on” status:

```
rmvm vm1 -f
```

Remove the definition of “vm1” and related storage:

```
rmvm vm1 -p
```

## Clone the virtual machine

**Clone** is an operation that creating a VM from an existed one by inheriting most of its attributes and data.

Steps to **clone** a VM: first create a **VM master**, then create a VM with the newly created **VM master** in **attaching** or **detaching** mode.

### In attaching mode

In this mode, all the newly created VMs are attached to the VM master. Since the image of the newly created VM only includes the differences from the VM master, which requires less disk space. The newly created VMs can NOT run without the VM master.

An example is shown below:

Create the VM master “vm5” from a VM node “vm1”:

```
#clonevm vm1 -t vm5
vm1: Cloning vm1.sda.qcow2 (currently is 1050.6640625 MB and has a capacity of 4096MB)
vm1: Cloning of vm1.sda.qcow2 complete (clone uses 1006.74609375 for a disk size of ↪
↪4096MB)
vm1: Rebasing vm1.sda.qcow2 from master
vm1: Rebased vm1.sda.qcow2 from master
```

The newly created VM master “vm5” can be found in the **vmmaster** table.

```
#tabdump vmmaster
name,os,arch,profile,storage,storagemodel,nics,vintage,originator,comments,disable
"vm5","<os>","<arch>","compute","nfs://<storage_server_ip>/vms/kvm",,"br0","<date>","root
↪",,
```

Clone a new node vm2 from VM master vm5:

```
clonevm vm2 -b vm5
```

### In detaching mode

Create a VM master “vm6” .

```
#clonevm vm2 -t vm6 -d
vm2: Cloning vm2.sda.qcow2 (currently is 1049.4765625 MB and has a capacity of 4096MB)
vm2: Cloning of vm2.sda.qcow2 complete (clone uses 1042.21875 for a disk size of 4096MB)
```

Clone a VM “vm3” from the VM master “vm6” in detaching mode:

```
#clonevm vm3 -b vm6 -d
vm3: Cloning vm6.sda.qcow2 (currently is 1042.21875 MB and has a capacity of 4096MB)
```

## Migrate Virtual Machines

Virtual machine migration is a process that moves the virtual machines (guests) between different hypervisors (hosts).

Note: The VM storage directory should be accessible from both hypervisors (hosts).

Migrate the VM “kvm1” from hypervisor “hyp01” to hypervisor “hyp02”:

```
#rmigrate kvm1 hyp02
kvm1: migrated to hyp02
```

## Trouble Shooting

### VNC client complains the credentials are not valid

**Issue:**

While connecting to the hypervisor with VNC, the vnc client complains “Authentication failed”.

**Solution:**

Check whether the clocks on the hypervisor and headnode are synced

### rpower fails with “Error: internal error Process exited while reading console log qemu: Permission denied”

**Issue:**

```
#rpower vm1 on
vm1: Error: internal error Process exited while reading console log output:
↳ char device redirected to /dev/pts/1
qemu: could not open disk image /var/lib/xcat/pools/2e66895a-e09a-53d5-74d3-
↳ eccdd9746eb5/vm1.sda.qcow2: Permission denied: internal error Process exited
↳ while reading console log output: char device redirected to /dev/pts/1
qemu: could not open disk image /var/lib/xcat/pools/2e66895a-e09a-53d5-74d3-
↳ eccdd9746eb5/vm1.sda.qcow2: Permission denied
```

**Solution:**

Usually caused by incorrect permission in NFS server/client configuration. NFSv4 is enabled in some Linux distributions such as CentOS6 by default. The solution is simply to disable NFSv4 support on the NFS server by uncommenting the following line in “/etc/sysconfig/nfs”:

```
RPCNFSDARGS="-N 4"
```

Then restart the NFS services and try to power on the VM again...

**Note:** For stateless hypervisor, purge the VM by `rmvm -p vm1`, reboot the hypervisor and then create the VM.

### rpower fails with “Error: internal error: process exited while connecting to monitor qemu: Permission denied”

**Issue:**

```
#rpower vm1 on
vm1: Error: internal error: process exited while connecting to monitor: 2016-
↳ 02-03T08:28:54.104601Z qemu-system-ppc64: -drive file=/var/lib/xcat/pools/
↳ c7953a80-89ca-53c7-64fb-2dcfc549bd45/vm1.sda.qcow2,if=none,id=drive-scsi0-0-
↳ 0-0,format=qcow2,cache=none: Could not open '/var/lib/xcat/pools/c7953a80-
↳ 89ca-53c7-64fb-2dcfc549bd45/vm1.sda.qcow2': Permission denied
```

**Solution:**

Usually caused by SELinux policies. The solution is simply to disable SELinux on the vmhost/hypervisor by editing “/etc/selinux/config” and change the SELINUX line to SELINUX=disabled:

```
SELINUX=disabled
```

Then reboot the hypervisor. . .

**rmigrate fails with “Error: libvirt error code: 38, message: unable to connect to server at ‘c910f05c35:49152’: No route to host.”**

#### Issue:

```
#rmigrate vm1 kvmhost2
vm1: Error: libvirt error code: 38, message: unable to connect to server at
↳ 'kvmhost2:49152': No route to host: Failed migration of vm1 from kvmhost1 to
↳ kvmhost2
```

#### Solution:

Usually caused by active firewall. To disable the firewall issue:

```
systemctl disable firewalld
```

**rmigrate fails with “Error: 38, message: failed to create directory ‘<dir-name>’: File exists: Unknown issue libvirt error code.”**

#### Issue:

```
#rmigrate vm1 kvmhost2
vm1: Error: 38, message: failed to create directory '<dir-name>': File exists:
↳ Unknown issue libvirt error code.
```

#### Solution:

Usually happens when *nfs*: is specified for vmstorage attribute but that NFS directory is no longer mounted. Make sure the directory /var/lib/xcats/pools is empty on the destination kvmhost.

### Error: Cannot communicate via libvirt to kvmhost1

#### Issue:

The kvm related commands complain “Error: Cannot communicate via libvirt to kvmhost1”

#### Solution:

Usually caused by incorrect ssh configuration between xCAT management node and hypervisor. Make sure it is possible to access the hypervisor from management node via ssh without password.

## Fail to ping the installed VM

### Issue:

The newly installed stateful VM node is not pingable, the following message can be observed in the console during VM booting:

```
ADDRCONF(NETDEV_UP): eth0 link is not ready.
```

### Solution:

Usually caused by the incorrect VM NIC model. Try the following steps to specify “virtio”:

```
rmvm vm1
chdef vm1 vmnicnicmodel=virtio
mkvm vm1
```

## x86\_64

This section is not available at this time.

Refer to [xCAT Documentation](#) on SourceForge for information on System X servers.

## 1.4.3 References

### xCAT Man Pages

These man pages are auto generated from .pod files to .rst files using the `create_man_pages.py` script under `xcat-core`

#### man1

#### addkit.1

#### NAME

**addkit** - Adds product software Kits to an xCAT cluster environment.

#### SYNOPSIS

**addkit** [-? | -h | --help] [-v | --version]

**addkit** [-i | --inspection] *kitlist*

**addkit** [-V | --verbose] [-p | --path *path*] *kitlist*

## DESCRIPTION

The **addkit** command installs a kit on the xCAT management node from a kit tarfile or directory. It creates xCAT database definitions for the kit, kitrepo, and kitcomponent.

**Note:** xCAT Kit support is ONLY available for Linux operating systems.

## OPTIONS

### **-h|--help**

Display usage message.

### **-V|--verbose**

Verbose mode.

### **-v|--version**

Command version.

### **-i|--inspection**

Show the summary of the given kits

### **-p|--path *path***

The destination directory to which the contents of the kit tarfiles and/or kit deploy directories will be copied. When this option is not specified, the default destination directory will be formed from the installdir site attribute with *./kits* subdirectory.

### *kitlist*

A comma delimited list of *kit\_tarball\_files* or *kit\_deploy\_directories* to be added to the xCAT environment. Each entry can be an absolute or relative path. See xCAT documentation for more information on building kits.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To add kits from tarball files:

```
addkit kit-test1.tar.bz2,kit-test2.tar.bz2
```

2. To add kits from directories:

```
addkit kit-test1,kit-test2
```

3. To add kits from tarball *kit-test1.tar.bz2* to target path */install/test*:

```
addkit -p /install/test kit-test1.tar.bz2
```

4. To see general information about kit *kit-test1.tar.bz2* without adding the kit to xCAT:

```
addkit -i kit-test1.tar.bz2
```

## SEE ALSO

lskit(1)|lskit.1, rmkit(1)|rmkit.1, addkitcomp(1)|addkitcomp.1, rmkitcomp(1)|rmkitcomp.1, chkitcomp(1)|chkitcomp.1

## addkitcomp.1

### NAME

**addkitcomp** - Assign Kit components to an xCAT osimage.

### SYNOPSIS

**addkitcomp** [-? | -h | --help] [-v | --version]

**addkitcomp** [-V | --verbose] [-a | --adddeps] [-f | --force] [-n | --noupgrade] [--noscripts] -i *osimage kitcomp-name\_list*

### DESCRIPTION

The **addkitcomp** command will assign kit components to an xCAT osimage. The kit component meta rpm, package rpm and deploy parameters will be added to osimage's otherpkg.pkglist and postbootscripts will be added to osimages's postbootscripts attribute.

**Note:** xCAT Kit support is ONLY available for Linux operating systems.

### OPTIONS

**-a|--adddeps**

Assign kitcomponent dependencies to the osimage.

**-h|--help**

Display usage message.

**-V|--verbose**

Verbose mode.

**-v|--version**

Command version.

**-f|--force**

Add kit component to osimage even if there is a mismatch in OS, version, arch, serverrole, or kitcompdeps

**-i *osimage***

The osimage name that the kit component is assigning to.

**-n|--noupgrade**



1. Allow multiple versions of kitcomponent to be installed into the osimage, instead of kitcomponent upgrade.
2. Kit components added by addkitcomp -n will be installed separately behind all other ones which have been added.

### **--noscripts**

Do not add kitcomponent's postbootscripts to osimage

### *kitcompname\_list*

A comma-delimited list of valid full kit component names or kit component basenames that are to be added to the osimage.

## **RETURN VALUE**

- 0 The command completed successfully.
- 1 An error has occurred.

## **EXAMPLES**

1. To add a single kit component to osimage "rhels6.2-ppc64-netboot-compute":

```
addkitcomp -i rhels6.2-ppc64-netboot-compute comp-test1-1.0-1-rhels-6.2-ppc64
```

2. To add a kit component to osimage with dependencies, use the -a (addeps) option:

```
addkitcomp -a -i rhels6.2-ppc64-netboot-compute comp-test2-1.0-1-rhels-6.2-ppc64
```

3. To add a kit component to osimage with incompatible osarch, osversion or ostype, use the -f (force) option:

```
addkitcomp -f -i rhels6.2-ppc64-netboot-compute comp-test1-1.0-1-rhels-6.2-ppc64
```

4. To add a new version of kit component to osimage without upgrade, use the -n (noupgrade) option:

```
addkitcomp -n -i rhels6.2-ppc64-netboot-compute comp-test2-1.0-1-rhels-6.2-ppc64
```

## **SEE ALSO**

lskit(1)|lskit.1, addkit(1)|addkit.1, rmkit(1)|rmkit.1, rmkitcomp(1)|rmkitcomp.1, chkkitcomp(1)|chkkitcomp.1

## **bmcdiscover.1**

## **NAME**

**bmcdiscover** - Discover Baseboard Management Controllers (BMCs) using a scan method

## SYNOPSIS

**bmcdiscover** [-? | -h | --help]

**bmcdiscover** [-v | --version]

**bmcdiscover** --range *ip\_ranges* [--sn *SN\_nodename*] [-s *scan\_method*] [-u *bmc\_user*] [-p *bmc\_passwd*] [-n *new\_bmc\_passwd*] [-z] [-w]

## DESCRIPTION

The **bmcdiscover** command will discover Baseboard Management Controllers (BMCs) using a scan method.

The command uses **nmap** to scan active nodes over a specified IP range. The IP range format should be a format that is acceptable by **nmap**.

**Note:** The scan method currently supported is **nmap**.

**Note:** Starting on January 1, 2020, some newly shipped systems will require the default BMC password to be changed before they can be managed by xCAT. Use **bmcdiscover** with **-n** option to specify new BMC password.

## OPTIONS

### --range

Specify one or more IP ranges acceptable to **nmap**. IP range can be hostnames, IP addresses, networks, etc. A single IP address (10.1.2.3), several IPs with commas (10.1.2.3,10.1.2.10), IP range with “-” (10.1.2.0-100) or an IP range (10.1.2.0/24) can be specified. If the range is very large, the **bmcdiscover** command may take a long time to return.

### --sn

Specify one or more service nodes on which **bmcdiscover** will run. In hierarchical cluster, the MN may not be able to access the BMC of CN directly, but SN can. In that case, **bmcdiscover** will be dispatched to the specified SNs. Then, the nodename of the service node that **bmcdiscover** is running on will be set to the ‘servicenode’ attribute of the discovered BMC node.

### -s

Scan method (The only supported scan method at this time is **nmap**)

### -z

List the data returned in xCAT stanza format

### -w

Write to the xCAT database.

### -u|--bmcuser

BMC user name.

### -p|--bmcpasswd

BMC user password.

### -n|--newbmcpw

New BMC user password.

### -h|--help

Display usage message

**-v|--version**

Display version information

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To get all responding BMCs from IP range “10.4.23.100-254” and “50.3.15.1-2”:

```
bmcdiscover -s nmap --range "10.4.23.100-254 50.3.15.1-2"
```

Note: Input for IP range can be in the form: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254.

2. To get all BMCs in IP range “10.4.22-23.100-254”, displayed in xCAT stanza format:

```
bmcdiscover -s nmap --range "10.4.22-23.100-254" -z
```

3. To discover BMCs through sn01:

```
bmcdiscover --sn sn01 -s nmap --range "10.4.22-23.100-254" -z
```

Output is similar to:

```
node-70e28414291b:
  objtype=node
  groups=all
  bmc=10.4.22.101
  cons=openbmc
  mgt=openbmc
  servicenode=sn01
  conserver=sn01
```

4. Discover the BMCs and write the discovered node definitions into the xCAT database and write out the stanza format to the console:

```
bmcdiscover -s nmap --range "10.4.22-23.100-254" -w -z
```

5. Discover the BMC with the specified IP address, change its default BMC password and display in xCAT stanza format:

```
bmcdiscover --range "10.4.22-23.100" -u root -p OpenBmc -n OpenBmc123 -z
```

## SEE ALSO

lsslp(1)|lsslp.1

## buildkit.1

### NAME

**buildkit** - Used to build a software product Kit which may be used to install software in an xCAT cluster.

### SYNOPSIS

**buildkit** [-? | -h | --help] [-v | --version]

To build a new Kit

**buildkit** [-V | --verbose] *subcommand* [*kit\_name*] [*repo\_name* | **all**] [-l | --kitloc *kit\_location*]

To add packages to an existing Kit.

**buildkit** [-V | --verbose] *addpkgs kit\_tarfile* [-p | --pkgdir *package\_directory\_list*] [-k | --kitversion *version*] [-r | --kitrelease *release*] [-l | --kitloc *kit\_location*]

### DESCRIPTION

The **buildkit** command provides a collection of utilities that may be used to package a software product as a Kit tarfile that can be used to install software on the nodes of an xCAT cluster. A Kit contains the product software packages, configuration and control information, and install and customization scripts.

Note: The xCAT support for Kits is only available for Linux operating systems.

You will need to run the **buildkit** command several times with different subcommands to step through the process of building a kit:

By default the **buildkit** subcommands will operate in the current working directory, (ie. look for files, create directories etc.). You could specify a different location by using the “-l | --kitloc *kit\_location*” option.

The *kit\_location* is the full path name of the directory that contains the kit files. You would use the same location value for all the buildkit subcommands.

For example, to create a new kit named “prodkit” in the directory /home/mykits/ *either* run:

1. If no location is provided then the command will create a subdirectory called “prodkit” in the current directory “/home/mykits” and the new kit files will be created there.

**cd /home/mykits**

**buildkit create prodkit**

or

2. If a location is provided then the Kit files will be created there. Note that the Kit name does not necessarily have to be the directory name where the kit files are located.

**buidkit create prodkit -l /home/mykits/prodkit**

In both cases the `/home/mykits/prodkit` directory is created and the initial files for the kit are created in that directory. The following example illustrates the basic process for building a new Kit. In this example we are building a Kit named “mytstkit”.

1. Change to the directory where you wish to create the Kit.

2. Create a template directory for your kit:

```
buildkit create mytstkit
```

3. Change directory to the new “mytstkit” subdirectory that was just created.

```
cd mytstkit
```

4. Edit the buildkit configuration file for your kit:

```
vi buildkit.conf
```

(See xCAT Kit documentation for details.)

5. Create all required files, scripts, plugins, and packages for your kit.

6. Validate your kit build configuration and fix any errors that are reported:

```
buildkit chkconfig
```

7. List the repos defined in your buildkit configuration file:

```
buildkit listrepo
```

8. For each repo name listed, build the repository. Note that if you need to build repositories for OS distributions, versions, or architectures that do not match the current system, you may need to copy your kit template directory to an appropriate server to build that repository, and then copy the results back to your main build server. For example, to build a repo named “rhels6.3” you would run the following command.

```
buildkit buildrepo rhels6.3
```

or, you can build all of the repos at one time if there are no OS or architecture dependencies for kitcomponent package builds or kitpackage builds:

```
buildkit buildrepo all
```

9. Build the kit tar file:

```
buildkit buildtar
```

## OPTIONS

**-h|--help**

Display usage message.

**-k|--kitversion *version***

Product version.

**-l|--kitloc *kit\_location***

The directory location of the Kit files.

**-p|--pkgdir *package\_directory\_list***

A comma-separated list of directory locations for product RPMs.

**-r|--kitrelease *release***

Product release.

**-V |--verbose**

Verbose mode.

**-v|--version**

Command version.

## **SUB-COMMANDS**

**create** *kit\_basename*

Creates a new kit build directory structure for kit *kit\_basename* using the location specified on the command line or the current directory. The sample kit files from `/opt/xcat/share/xcat/kits/kit_template` are copied over, and the `buildkit.conf` file is modified for the specified *kit\_basename*.

**chkconfig**

Reads the `buildkit.conf` file, verifies that the file syntax is correct and that all specified files exist.

**listrepo**

Reads the `buildkit.conf` file, lists all Kit package repositories listed in the file, and reports the build status for each repository.

**buildrepo** { *repo\_name* | **all** }

Reads the `buildkit.conf` file, and builds the specified Kit package repository. The built packages are placed in the directory `<kit_location>/build/kit_repodir/repo_name`. If **all** is specified, all kit repositories are built.

**cleanrepo** { *repo\_name* | **all** }

Reads the `buildkit.conf` file, and deletes all the package files and package meta data files from the `<kit_location>/build/kit_repodir/repo_name` directory. If **all** is specified, all kit repository files are deleted.

**buildtar**

Reads the `buildkit.conf` file, validates that all kit repositories have been built, and builds the Kit tar file `<kit_location>/kitname.tar.bz2`.

**cleantar**

Reads the `<kit_location>/buildkit.conf` file and *deletes* the following:

- Kit tar files matching `<kit_location>/kit_name\.tar.bz2*`.
- `<kit_location>/build/kit_name`
- `<kit_location>/rpmbuild`
- `<kit_location>/tmp`
- `<kit_location>/debbuild`

Caution: Make sure you back up any tar files you would like to keep before running this subcommand.

**cleanall**

Equivalent to running **buildkit cleanrepo all** and **buildkit cleantar**.

**addpkgs**

```
kit_tarfile {-p | --pkgdir package_directory_list} [-k | --kitversion version] [-r | --kitrelease release]
```

Add product package rpms to a previously built kit tar file. This is used for partial product kits that are built and shipped separately from the product packages, and are identified with a *kit\_tarfile* name of *kitname.NEED\_PRODUCT\_PKGS.tar.bz2*. Optionally, change the kit release and version values when building the new kit tarfile. If kitcomponent version and/or release values are defaulted to the kit values, those will also be changed and new kitcomponent rpms will be built. If kit or kitcomponent scripts, plugins, or other files specify name, release, or version substitution strings, these will all be replaced with the new values when built into the new complete kit tarfile *kit\_location/new\_kitname.tar.bz2*.

## RETURN VALUE

<B>0

The command completed successfully.

<B>1

An error has occurred.

## EXAMPLES

1. To create the sample kit shipped with the xCAT-buildkit rpm on a RHEL 6.3 server and naming it **mykit**, run the following commands:

```
cd /home/myuserid/kits
```

```
buildkit create mykit
```

```
cd mykit
```

```
vi buildkit.conf
```

```
buildkit chkconfig
```

```
buildkit listrepo
```

```
buildkit buildrepo all
```

```
buildkit buildtar
```

2. To clean up a kit repository directory after build failures on a RHEL 6.3 server to prepare for a new kit repository build, run:

```
buildkit cleanrepo rhel6.3
```

3. To clean up all kit build files, including a previously built kit tar file, run

```
buildkit cleanall
```

4. To create a kit named “tstkit” located in /home/foobar/tstkit instead of the current working directory.

```
buildkit create tstkit -l /home/foobar/tstkit
```

## FILES

/opt/xcat/bin/buildkit  
/opt/xcat/share/xcat/kits/kit\_template  
/opt/xcat/share/xcat/kits/kitcomponent.spec.template  
<kit location>/buildkit.conf  
<kit location>/build/kitname/kit.conf  
<kit location>/kitname.tar.bz2

## SEE ALSO

addkit(1), lskit(1), rmkit(1), addkitcomp(1), rmkitcomp(1), chkkitcomp(1)

## cfgve.1

## NAME

**cfgve** - Configure the elements for a virtual environment.

## SYNOPSIS

**cfgve -t dc -m manager -o object [-c -k nfs | localfs | -r]**  
**cfgve -t cl -m manager -o object [-c -p cpu type | -r -f]**  
**cfgve -t sd -m manager -o object [-c | -g | -s | -a | -b | -r -f]**  
**cfgve -t nw -m manager -o object [-c -d data center -n vlan ID | -a -l cluster | -b | -r]**  
**cfgve -t tpl -m manager -o object [-r]**

## DESCRIPTION

The **cfgve** command can be used to configure a virtual environment for ‘Storage Domain’, ‘Network’ and ‘Template’ objects.

The mandatory parameter **-m manager** is used to specify the address of the manager of virtual environment. xCAT needs it to access the RHEV manager.

The mandatory parameter **-t type** is used to specify the type of the target object.

Basically, **cfgve** command supports five types of object: **dc**, **cl**, **sd**, **nw** and **tpl**.

**dc** - The **create** and **remove** operations are supported.

**cl** - The **create** and **remove** operations are supported.

**sd** - The **create**, **attach**, **detach**, **activate**, **deactivate** and **remove** operations are supported.

**nw** - The **create**, **attach**, **detach** and **remove** operations are supported.

**tpl** - The **remove** operation is supported.



The mandatory parameter **-o** *object* is used to specify which object to configure.

## OPTIONS

**-a** To attach the target object.

**-b** To detach the target object.

**-c** To create the target object.

For creating of **Storage Domain**, the target storage domain will be created first, then attached to data center and activated.

The parameters that used to create the storage domain are gotten from 'virtsd' table. The detail parameters in the virtsd table:

**virtsd.node** - The name of the storage domain.

**virtsd.sdtype** - The type of storage domain. Valid value: data, iso, export. Default value is 'data'.

**virtsd.stype** - The storage type. "nfs" or "localfs".

**virtsd.location** - The location of the storage. **nfs**: Format: [nfsserver:nfspath]. The NFS export directory must be configured for read write access and must be owned by vds:m:kvm. **localfs**: "/data/images/rhev" is set by default.

**virtsd.host** - A host must be specified for a storage domain as SPM (Storage Pool Manager) when initialize the storage domain. The role of SPM may be migrated to other host by rhev-m during the running of the datacenter (For example, when the current SPM encountered issue or going to maintenance status.

**virtsd.datacenter** - The storage will be attached to. 'Default' data center is the default value.

**-d** *data center*

The name of data center.

Specify the 'Data Center' that will be used for the object to be attached to. It is used by <nw> type.

**-f** It can be used with **-r** to remove the target object by force.

For removing of **Storage Domain**, if **-f** is specified, the storage domain will be deactivated and detached from data center before the removing.

**-g** To activate the target object.

**-h** Display usage message.

**-k** *storage type*

To specify the type of the storage type when creating the data center.

Supported type: nfs; localfs.

**-l** *cluster*

Specify the cluster for the network to attach to.

**-m** *manager*

Specify the manager of the virtual environment.

For RHEV, the FQDN (Fully Qualified Domain Name) of the rhev manager have to be specified.

**-n** *vlan ID*

To specify the vlan number when creating a network.

**-o** *object*

The name of the target object.

**-p** *cpu type*

To specify the cpu type when creating the cluster. **Intel Penryn Family** is default type.

Supported type: **Intel Conroe Family, Intel Penryn Family, Intel Nehalem Family, Intel Westmere Family, AMD Opteron G1, AMD Opteron G2, AMD Opteron G3**

**-r** To remove the target object.

For removing of **Storage Domain**, the storage space will be formatted after removing.

**-s** To deactivate the target object.**-t** *type*

Specify the **type** of the target object.

**Supported types:**

**dc** - Data Center **cl** - Cluster **sd** - Storage Domain **nw** - Network **tpl** - Template

**RETURN VALUE**

0 The command completed successfully.

1 An error has occurred.

**EXAMPLES**

1. To create the Storage Domain 'sd1', enter:

```
cfgve -t sd -m <FQDN of rhev manager> -o sd1 -c
```

2. To deactivate the Storage Domain 'sd1' from data center, enter:

```
cfgve -t sd -m <FQDN of rhev manager> -o sd1 -s
```

3. To remove the Storage Domain 'sd1', enter:

```
cfgve -t sd -m <FQDN of rhev manager> -o sd1 -r
```

4. To create the network 'nw1', enter:

```
cfgve -t nw -m <FQDN of rhev manager> -o nw1 -c
```

5. To remove the template 'tpl01', enter:

```
cfgve -t tpl -m <FQDN of rhev manager> -o tpl01 -r
```

## FILES

/opt/xcat/bin/cfgve

## SEE ALSO

lsve(1)|lsve.1

## cfm2xcat.1

## NAME

**cfm2xcat** - Migrates the CFM setup in CSM to the xdcp rsync setup in xCAT.

## SYNOPSIS

**cfm2xcat** [-i *path of the CFM distribution files generated*] [-o *path of the xdcp rsync files generated from the CFM distribution files*]

**cfm2xcat** [-h]

## DESCRIPTION

Copy the cfm2xcat command to the CSM Management Server. Run the command, indicating where you want your files saved with the -i and -o flags. They can be in the same directory.

The cfm2xcat command will run cfmupdatenode -a, saving the generated CFM distribution files in the directory indicates with (-i). From those distribution files, it will generate xdcp rsync input files (-F option on xdcp) in the directory indicated by (-o).

Check the rsync files generated. There will be a file generated (rsyncfiles) from the input -o option on the command, and the same file with a (.nr) extension generated for each different noderange that will used to sync files based on your CFM setup in CSM. The rsyncfiles will contain the rsync file list. The rsyncfiles.nr will contain the noderange. If multiple noderanges then the file name (rsyncfiles) will be appended with a number.

## OPTIONS

**-h** Display usage message.

**-i** Path of the CFM distribution files generated from the cfmupdatenode -a command.

**-o** Path of the xdcp rsync input file generated from the CFM distribution files.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To build xCAT rsync files to use with `xdcp -F`, enter on the CSM Management Server, make sure the path exists:

```
cfm2xcat -i /tmp/cfm/cfmdistfiles -o /tmp/cfm/rsyncfiles
```

2. To use the file on the xCAT Management Node copy to `/tmp/cfm` on the xCAT MN:

```
xdcp ^/tmp/cfm/rsyncfiles.nr -F /tmp/cfm/rsyncfiles
xdcp ^/tmp/cfm/rsyncfiles.nr1 -F /tmp/cfm/rsyncfiles1
xdcp ^/tmp/cfm/rsyncfiles.nr2 -F /tmp/cfm/rsyncfiles2
```

## FILES

`/opt/xcat/share/xcat/tools/cfm2xcat`

## chdef.1

## NAME

**chdef** - Change xCAT data object definitions.

## SYNOPSIS

**chdef** [-h | --help] [-t *object-types*]

**chdef** [-t *object-types*] [-o *object-names*] [-n *new-name*] [*node*]

**chdef** [-V | --verbose] [-t *object-types*] [-o *object-names*] [-d | --dynamic] [-p | --plus] [-m | --minus] [-z | --stanza] [[-w *attr==val*] [-w *attr=~val*] ...] [*noderange*] [*attr=val* [*attr=val...*]] [-u [*provmethod=* {install | netboot | statelite}]] [*profile=xxx*] [*osvers=value*] [*osarch=value*]]

## DESCRIPTION

This command is used to change xCAT object definitions which are stored in the xCAT database. The default is to replace any existing attribute value with the one specified on the command line. The command will also create a new definition if one doesn't exist.

This command also can be used to change the xCAT object name to a new name. Note: the site,monitoring types can NOT be supported.

## OPTIONS

*attr=val [attr=val ...]*

Specifies one or more “attribute equals value” pairs, separated by spaces. Attr=val pairs must be specified last on the command line. Use the help option to get a list of valid attributes for each object type.

### **-d|--dynamic**

Use the dynamic option to change dynamic node groups definition. This option must be used with -w option.

### **-h|--help**

Display usage message.

### **-m|--minus**

If the value of the attribute is a list then this option may be used to remove one or more items from the list.

### **-n new-name**

Change the current object name to the new-name which is specified by the -n option. Objects of type site, group and monitoring cannot be renamed with the -n option. Note: For the -n option, only one node can be specified. For some special nodes such as fsp, bpa, frame, cec etc., their name is referenced in their own hcp attribute, or the hcp attribute of other nodes. If you use -n option, you must manually change all hcp attributes that refer to this name.

### *noderange*

A set of comma delimited node names and/or group names. (must be the first parameter) See the “noderange” man page for details on supported formats.

### **-o object-names**

A set of comma delimited object names.

### **-p|--plus**

This option will add the specified values to the existing value of the attribute. It will create a comma-separated list of values.

### **-t object-types**

A set of comma delimited object types. Use the help option to get a list of valid object types.

### **-V|--verbose**

Verbose mode.

### **-w attr==val -w attr=~val ...**

Use one or multiple -w flags to specify the selection string that can be used to select objects. The operators ==, !=, =~ and !~ are available. Use the help option to get a list of valid attributes for each object type.

#### **Operator descriptions:**

== Select nodes where the attribute value is exactly this value. != Select nodes where the attribute value is not this specific value. =~ Select nodes where the attribute value matches this regular expression. !~ Select nodes where the attribute value does not match this regular expression.

Note: the operator !~ will be parsed by shell, if you want to use !~ in the selection string, use single quote instead. For example:-w ‘mgt!~ipmi’.

### **-z|--stanza**

Indicates that the file being piped to the command is in stanza format. See the `xcatstanzafile` man page for details on using xCAT stanza files.

**-u**

Fill in the attributes such as `template` file, `pkglist` file and `otherpkglist` file of `osimage` object based on the specified parameters. It will search “/install/custom/” directory first, and then “/opt/xcat/share/”.

Note: this option only works for objtype **osimage**.

**RETURN VALUE**

0 The command completed successfully.

1 An error has occurred.

**EXAMPLES**

1. To change a site definition.

```
chdef -t site -o clustersite installdir=/xcatinstall
```

2. To change a basic node definition.

```
chdef -t node -o node01 groups="all,aix"
```

(The group definitions are also created if they don't already exist.)

3. To add another group to the “groups” attribute in the previous example.

```
chdef -p -t node -o node01 groups="compute"
```

4. To remove the “all” group from the “groups” attribute in the previous example.

```
chdef -m -t node -o node01 groups="all"
```

5. To replace the current “groups” attribute value of “node01”.

```
chdef -t node -o node01 groups="linux"
```

6. To add “node01” to the “members” attribute of a group definition called “LinuxNodes”.

```
chdef -p -t group -o LinuxNodes members="node01"
```

7. To update a set of definitions based on information contained in the stanza file `mystanzafile`.

```
cat mystanzafile | chdef -z
```

8. To update a dynamic node group definition to add the `cons=hmc` wherevals pair.

```
chdef -t group -o dynggrp -d -p -w cons==hmc
```

9. To change the node object name from `node1` to `node2`.

```
chdef -t node -o node1 -n node2
```

10. To change the node hwtype, this command will change the value of ppc.nodetype.

```
chdef -t node -o node1 hwtype=lpar
```

11. To change the policy table for policy number 7.0 for admin1

```
chdef -t policy -o 7.0 name=admin1 rule=allow
```

12. To change the node nic attributes

```
chdef -t node -o cn1 nicips.eth0="1.1.1.1|1.2.1.1" nicnetworks.eth0="net1|net2"
↔ nictypes.eth0="Ethernet"
```

13. To update an osimage definition.

```
chdef redhat6img -u provmethod=install
```

## FILES

\$XCATROOT/bin/chdef

(The XCATROOT environment variable is set when xCAT is installed. The default value is “/opt/xcat”.)

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

mkdef(1)|mkdef.1, lsdef(1)|lsdef.1, rmdef(1)|rmdef.1, xcatstanzafile(5)|xcatstanzafile.5

## chhypervisor.1

## NAME

**chhypervisor** - Configure the virtualization hosts.

## SYNOPSIS

**RHEV specific :**

**chhypervisor** *noderange* [-a]

**chhypervisor** *noderange* [-n]

**chhypervisor** *noderange* [-p]

**chhypervisor** *noderange* [-e]

**chhypervisor** *noderange* [-d]

**zVM specific :**

```

chhypervisor noderange [--adddisk2pool function region volume group]
chhypervisor noderange [--addscsi device_number device_path option persist]
chhypervisor noderange [--addvlan name owner type transport]
chhypervisor noderange [--addvswitch name osa_dev_addr osa_exp_adapter controller connect (0, 1, or 2) memory_queue router transport vlan_id port_type update gvrp native_vlan]
chhypervisor noderange [--addzfcp2pool pool status wwpn lun size owner]
chhypervisor noderange [--removediskfrompool function region group]
chhypervisor noderange [--removescsi device_number persist (YES or NO)]
chhypervisor noderange [--removevlan name owner]
chhypervisor noderange [--removevswitch name]
chhypervisor noderange [--removezfcpfrompool pool lun wwpn]
chhypervisor noderange [--smcli function arguments]

```

## DESCRIPTION

The **chhypervisor** command can be used to configure the RHEV-h.

The rhev-h host will register to the rhev-m automatically, but admin needs to approve the host can be added to the ‘cluster’ with **-a** flag .

After registering, the network interfaces of host need to be added to the ‘network’ of RHEV. And the power management for the host should be configured so that rhev-m could make proper decision when certain host encountered error.

The **chhypervisor** command can also be used to configure the zVM host.

For each host, an entry should be added to the hypervisor table:

The columns of hypervisor table:

**hypervisor.node** - rhev-h host name or zVM host name (lower-case).

**hypervisor.type** - Must be set to ‘rhev’h’ or ‘zvm’.

**hypervisor.mgr** - The rhev manager (The FQDN of rhev-m server) for the host.

**hypervisor.interface** - The configuration for the nics. Refer to **-n**.

**hypervisor.cluster** - The cluster that the host will be added to. The default is ‘Default’ cluster if not specified.

## OPTIONS

### RHEV specific :

**-a** Approve the host that to be added to cluster.

Before approve, the status of the host must be ‘pending\_approval’.

**-n** Configure the network interfaces for the host.

Note: This operation only can be run when host is in ‘maintenance mode’. Use **-d** to switch the host to ‘maintenance’ mode.

The interfaces which configured in hypervisor.interface will be added to the network of RHEV.



The format of hypervisor.interface is multiple [network:interfacename: protocol:IP:netmask:gateway] sections separated with '|'. For example: [rhevm2:eth0:static:10.1.0.236:255.255.255.0:0.0.0.0].

**network** - The logic network which has been created by 'cfgve -t nw' or the default management network 'rhevm'.

**interfacename** - Physical network name: 'eth0','eth1'...

**protocol** - To identify which boot protocol to use for the interface: dhcp or static.

**IP** - The IP address for the interface.

**netmask** - The network mask for the interface.

**gateway** - The gateway for the interface. This field only can be set when the interface is added to 'rhevm' network.

**-p** Configure the power management for the host.

The power management must be configured for the rhev-h host to make the rhev-m to monitor the power status of the host, so that when certain host failed to function, rhev-m will fail over certain role like SPM to other active host.

For rack mounted server, the bmc IP and user:password need to be set for the power management (These parameters are gotten from ipmi table). rhev-m uses the ipmi protocol to get the power status of the host.

**-e** To activate the host.

**-d** To deactivate the host to maintenance mode.

**-h** Display usage message.

## zVM specific :

**--adddisk2pool** *function region volume group*

Add a disk to a disk pool defined in the EXTENT CONTROL. Function type can be either: (4) Define region as full volume and add to group OR (5) Add existing region to group. If the volume already exists in the EXTENT CONTROL, use function 5. If the volume does not exist in the EXTENT CONTROL, but is attached to SYSTEM, use function 4.

**--addscsi** *device\_number device\_path option persist*

Dynamically add a SCSI disk to a running z/VM system.

**--addvlan** *name owner type transport*

Create a virtual network LAN.

**--addvswitch** *name osa\_dev\_addr osa\_exp\_adapter controller connect (0, 1, or 2) memory\_queue router transport vlan\_id port\_type update gvrp native\_vlan*

Create a virtual switch.

**--addzfcp2pool** *pool status wwpn lun size owner*

Add a zFCP device to a device pool defined in xCAT. The device must have been carved up in the storage controller and configured with a WWPN/LUN before it can be added to the xCAT storage pool. z/VM does not have the ability to communicate directly with the storage controller to carve up disks dynamically.

**--removediskfrompool** *function region group*

Remove a disk from a disk pool defined in the EXTENT CONTROL. Function type can be either: (1) Remove region, (2) Remove region from group, (3) Remove region from all groups, OR (7) Remove entire group .

**--removescsi** *device\_number persist (YES or NO)*

Delete a real SCSI disk.

**--removevlan** *name owner*

Delete a virtual network LAN.

**--removevswitch** *name*

Delete a virtual switch.

**--removezfcpfrompool** *pool lun*

Remove a zFCP device from a device pool defined in xCAT.

**--smcli** *function arguments*

Execute a SMAPI function. A list of APIs supported can be found by using the help flag, e.g. `chhypervisor pokdev61 --smcli -h`. Specific arguments associated with a SMAPI function can be found by using the help flag for the function, e.g. `chhypervisor pokdev61 --smcli Image_Query_DM -h`. Only z/VM 6.2 and older SMAPI functions are supported at this time. Additional SMAPI functions will be added in subsequent zHCP versions.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

### RHEV specific :

1. To approve the host 'host1', enter:

```
chhypervisor host1 -a
```

2. To configure the network interface for the host 'host1', enter:

```
chhypervisor host1 -n
```

3. To configure the power management for the host 'host1', enter:

```
chhypervisor host1 -p
```

4. To activate the host 'host1', enter:

```
chhypervisor host1 -e
```

5. To deactivate the host 'host1', enter:

```
chhypervisor host1 -d
```

**zVM specific :**

1. To add a disk to a disk pool defined in the EXTENT CONTROL, enter:

```
chhypervisor pokdev61 --adddisk2pool 4 DM1234 DM1234 POOL1
```

2. To add a zFCP device to a device pool defined in xCAT, enter:

```
chhypervisor pokdev61 --addzfc2pool zfc1 free 500501234567C890_
↪4012345600000000 8G
```

3. To remove a region from a group in the EXTENT CONTROL, enter:

```
chhypervisor pokdev61 --removediskfrompool 2 DM1234 POOL1
```

4. To remove a zFCP device from a device pool defined in xCAT, enter:

```
chhypervisor pokdev61 --removezfcfrompool zfc1 4012345600000000_
↪500501234567C890
```

5. To execute a SMAPI function (Image\_Query\_DM), enter:

```
chhypervisor pokdev61 --smcli Image_Query_DM -T LNX3
```

**FILES**

/opt/xcat/bin/chhypervisor

**chkkitcomp.1****NAME**

**chkkitcomp** - Check if Kit components are compatible with an xCAT osimage.

**SYNOPSIS**

```
chkkitcomp [-? | -h | --help] [-v | --version]
```

```
chkkitcomp [-V | --verbose] -i osimage kitcompname_list
```

**DESCRIPTION**

The **chkkitcomp** command will check if the kit components are compatible with the xCAT osimage.

This command will ignore the current osimage.kitcomponents setting and check if the kitcompname\_list is compatible with the osimage and kit component dependencies.

**Note:** xCAT Kit support is ONLY available for Linux operating systems.

## OPTIONS

### **-h|--help**

Display usage message.

### **-V|--verbose**

Verbose mode.

### **-v|--version**

Command version.

### **-i *osimage***

The name of the osimage to check against.

### ***kitcompname\_list***

A comma-delimited list of valid full kit component names or kit component basenames that are to be checked against the osimage.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To check if a kit component, *comp-test1-1.0-1-rhels-6.2-ppc64* can be added to osimage *rhels6.2-ppc64-netboot-compute*:

```
chkkitcomp -i rhels6.2-ppc64-netboot-compute comp-test1-1.0-1-rhels-6.2-ppc64
```

## SEE ALSO

lskit(1)|lskit.1, addkit(1)|addkit.1, rmkit(1)|rmkit.1, addkitcomp(1)|addkitcomp.1, rmkitcomp(1)|rmkitcomp.1

### **chkosimage.1**

## NAME

**chkosimage** - Use this xCAT command to check an xCAT osimage.

## SYNOPSIS

**chkosimage** [-h | --help ]

**chkosimage** [-V] [-c|--clean] *osimage\_name*

## DESCRIPTION

This command is currently supported for AIX osimages only.

Use this command to verify if the NIM lpp\_source directories contain the correct software. The lpp\_source directory must contain all the software that is specified in the “installp\_bundle” and “otherpkgs” attributes of the osimage definition.

The command gets the name of the lpp\_source resource from the xCAT osimage definition and the location of the lpp\_source directory from the NIM resource definition.

It will check for installp, rpm and emgr type packages.

Note: Remember to use the prefixes, “I:”, “R:”, and “E:”, respectively, when specifying package names in an installp\_bundle file or an otherpkgs list.

In addition to checking for missing software the chkosimage command will also check to see if there are multiple matches. This could happen when you use wildcards in the software file names. For example, if you have perl-xCAT\* in a bundle file it could match multiple versions of the xCAT rpm package saved in your lpp\_source directory.

If this happens you must remove the unwanted versions of the rpms. If the extra rpms are not removed you will get install errors.

To help with this process you can use the “-c|--clean” option. This option will keep the rpm package with the most recent timestamp and remove the others.

The chkosimage command should always be used to verify the lpp\_source content before using the osimage to install any AIX cluster nodes.

## OPTIONS

**-c |--clean**

Remove any older versions of the rpms. Keep the version with the latest timestamp.

**-h |--help**

Display usage message.

*osimage\_name*

The name of the xCAT for AIX osimage definition.

**-V |--verbose**

Verbose mode.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. Check the XCAT osimage called “61image” to verify that the lpp\_source directories contain all the software that is specified in the “installp\_bundle” and “otherpkgs” attributes.

```
chkosimage -V 61image
```

2. Clean up the lpp\_source directory for the osimage named “61img” by removing any older rpms with the same names but different versions.

```
chkosimage -c 61img
```

## FILES

/opt/xcat/bin/chkosimage

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

mknimimage(1)|mknimimage.1

## chvlan.1

## NAME

**chvlan** - It adds or removes nodes for the vlan.

## SYNOPSIS

**chvlan** *vlanid* **-n** | **--nodes** *noderange* [**-i** | **--interface** *nic*]

**chvlan** *vlanid* **-n** | **--nodes** *noderange* **-d** | **--delete**

**chvlan** [**-h** | **--help**]

**chvlan** [**-v** | **--version**]

## DESCRIPTION

The **chvlan** command adds nodes to the given vlan. If -d is specified, the nodes will be removed from the vlan.

For added security, the root guard and bpdu guard will be enabled for the ports added to this vlan. However, the guards will not be disabled if the ports are removed from the vlan using chvlan (-d) or rmvlan commands. To disable them, you need to use the switch command line interface. Please refer to the switch command line interface manual to see how to disable the root guard and bpdu guard for a port.

## Parameters

*vlanid* is a unique vlan number.

## OPTIONS

**-n|--nodes** The nodes or groups to be added or removed. It can be stand alone nodes or KVM guests. It takes the noderange format. Please check the man page for noderange for details.

**-i|--interface** (For adding only). The interface name where the vlan will be tagged on. If omitted, the xCAT management network will be assumed. For KVM, it is the interface name on the host.

**-h|--help** Display usage message.

**-v|--version** The Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To add node1, node2 and node3 to vlan 3.

```
chvlan 3 -n node1,node2,node3
```

2. To add node1, node2 and node3 to vlan 3 using eth1 interface.

```
chvlan 3 -n node1,node2,node3 -i eth1
```

3. TO remove node1, node2 and node3 from vlan 3.

```
chvlan -n node1,node2,node3 -d
```

4. To add KVM guests node1 and node2 to vlan 3

```
mkdef node1 arch=x86_64 groups=kvm,all installnic=mac primarynic=mac mgt=kvm
↪ netboot=pxe nfsserver=10.1.0.204 os=rhels6 profile=compute provmethod=install
↪ serialport=0 serialspeed=115200 vmcpus=1 vmhost=x3650n01 vmmemory=512 vmnics=br0
↪ vmstorage=nfs://10.1.0.203/vms
```

(continues on next page)

(continued from previous page)

```
mkdef node2 arch=x86_64 groups=kvm,all installnic=mac primarynic=mac mgt=kvm
↪netboot=pxe nfsserver=10.1.0.204 os=rhels6 profile=compute provmethod=install
↪serialport=0 serialspeed=115200 vmcpus=1 vmhost=x3650n01 vmmemory=512 vmnics=br0
↪vmstorage=nfs://10.1.0.203/vms

chvlan 3 -n node1,node2

mkvm node1,node2 -s 20G

rpower node1,node2 on

rinstall node1,node2
```

5. To remove KVM guests node1 and node2 from vlan 3

```
chvlan 3 -n node1,node2 -d

rpower node1,node2 off

rmvm node1,node2
```

## FILES

/opt/xcat/bin/chvlan

## SEE ALSO

mkvlan(1)|mkvlan.1, rmvlan(1)|rmvlan.1, lsvlan(1)|lsvlan.1

## chvlanports.1

### NAME

**chvlanports** - It adds or removes nodes' switch interfaces for the vlan.

### SYNOPSIS

**chvlanports** *vlanid* **-n** | **--nodes** *noderange* **-i** | **--interface** *nic*

**chvlanports** *vlanid* **-n** | **--nodes** *noderange* **-i** | **--interface** *nic* **-d** | **--delete**

**chvlanports** [**-h** | **--help**]

**chvlanports** [**-v** | **--version**]



## DESCRIPTION

The **chvlanports** command adds nodes switch interfaces to the given vlan. If -d is specified, the nodes switch interfaces will be removed from the vlan.

This command won't create/remove vlans on switches, it just add node's switch ports into existing vlan or remove them from existing vlan on switch. Before calling chvlanports, the nodes switch interfaces should be configured in table switch, and vlan must already existing in switches. =head1 Parameters

*vlanid* is a unique vlan number.

## OPTIONS

**-n|--nodes** The nodes or groups to be added or removed. It takes the noderange format. Check the man page for noderange for details.

**-i|--interface** The interface name where the vlan will be tagged on.

**-h|--help** Display usage message.

**-v|--version** The Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To add node1, node2 and node3 to vlan 3 using eth1 interface.

```
chvlanports 3 -n node1,node2,node3 -i eth1
```

2. TO remove eth1 interface of node1, node2 and node3 from vlan 3.

```
chvlanports 3 -n node1,node2,node3 -i eth1 -d
```

## FILES

/opt/xcat/bin/chvlanports

## SEE ALSO

mkvlan(1)|mkvlan.1, rmvlan(1)|rmvlan.1, lsvlan(1)|lsvlan.1, chvlan(1)|chvlan.1

## chvm.1

### NAME

**chvm** - Changes HMC-, DFM-, IVM-, KVM-, VMware-, and zVM-managed partition profiles or virtual machine attributes. For Power 775, chvm could be used to change the octant configuration values for generating LPARs; change the I/O slots assignment to LPARs within the same CEC.

### SYNOPSIS

**chvm** [-h] --help]

**chvm** [-v] --version]

#### PPC (with HMC) specific:

**chvm** [-V] --verbose] *noderange* [-p *profile*]

**chvm** [-V] --verbose] *noderange* *attr=val* [*attr=val...*]

#### PPC (using Direct FSP Management) specific:

**chvm** *noderange* --p775 [-p *profile*]

**chvm** *noderange* --p775 -i *id* [-m *memory\_interleaving*] -r *partition\_rule*

**chvm** *noderange* [lparname={ \* | *name* }]

**chvm** *noderange* [vmcpus= *min/req/max*] [vmmemory= *min/req/max*] [vmothersetting=hugepage:N,bsr:N] [add\_physlots= *drc\_index1,drc\_index2...*] [add\_vmnics= *vlan1[,vlan2..]*] [add\_vmstorage=<N|viosnode:slotid>] [--vios] [del\_physlots= *drc\_index1,drc\_index2...*] [del\_vadapter= *slotid*]

#### KVM specific:

**chvm** *noderange* [--cpupin *hostcpuset*]

**chvm** *noderange* [--membind *numanodeset*]

**chvm** *noderange* [--devpassthru *pcidevice...*]

**chvm** *noderange* [--devdetach *pcidevice...*]

#### VMware/KVM specific:

**chvm** *noderange* [-a *size*] [-d *disk*] [-p *disk*] [--resize *disk=size*] [--cpus *count*] [--mem *memory*]

## zVM specific:

```

chvm noderange [--add3390 disk_pool device_address size mode read_password write_password multi_password]
chvm noderange [--add3390active device_address mode]
chvm noderange [--add9336 disk_pool device_address size mode read_password write_password multi_password]
chvm noderange [--adddisk2pool function region volume group]
chvm noderange [--addnic device_address type device_count]
chvm noderange [--addpagespool volume_address volume_label volume_use system_config_name system_config_type parm_disk_owner parm_disk_number parm_disk_password]
chvm noderange [--addprocessor device_address]
chvm noderange [--addprocessoractive device_address type]
chvm noderange [--addvdisk device_address size]
chvm noderange [--addzfcp pool device_address loaddev size tag wwpn lun]
chvm noderange [--connectnic2guestlan device_address lan owner]
chvm noderange [--connectnic2vswitch device_address vswitch]
chvm noderange [--copydisk target_address source_node source_address]
chvm noderange [--dedicatedevice virtual_device real_device mode]
chvm noderange [--deleteipl]
chvm noderange [--disconnectnic device_address]
chvm noderange [--formatdisk device_address multi_password]
chvm noderange [--grantvswitch vswitch]
chvm noderange [--purgerdr]
chvm noderange [--removedisk device_address]
chvm noderange [--removenic device_address]
chvm noderange [--removeprocessor device_address]
chvm noderange [--removeloaddev wwpn lun]
chvm noderange [--removezfcp device_address wwpn lun]
chvm noderange [--replacevs directory_entry]
chvm noderange [--setipl ipl_target load_parms parms]
chvm noderange [--setpassword password]
chvm noderange [--setloaddev wwpn lun]
chvm noderange [--sharevolume volume_address share_enable]
chvm noderange [--undedicatdevice device_address]

```

## DESCRIPTION

### PPC (with HMC) specific:

The `chvm` command modifies the partition profile for the partitions specified in `noderange`. A partitions current profile can be read using `lsvm`, modified, and piped into the `chvm` command, or changed with the `-p` flag.

This command also supports to change specific partition attributes by specifying one or more “attribute equals value” pairs in command line directly, without whole partition profile.

### PPC (using Direct FSP Management) specific:

For Power 755(use option `-p775` to specify):

`chvm` could be used to change the octant configuration values for generating LPARs. `chvm` is designed to set the Octant configure value to split the CPU and memory for partitions, and set Octant Memory interleaving value. The `chvm` will only set the pending attributes value. After `chvm`, the CEC needs to be rebooted manually for the pending values to be enabled. Before reboot the cec, the administrator can use `chvm` to change the partition plan. If the partition needs I/O slots, the administrator should use `chvm` to assign the I/O slots.

`chvm` is also designed to assign the I/O slots to the new LPAR. Both the current IO owning lpar and the new IO owning lpar must be powered off before an IO assignment. Otherwise, if the I/O slot is belonged to an Lpar and the LPAR is power on, the command will return an error when trying to assign that slot to a different lpar.

The administrator should use `lsvm` to get the profile content, and then edit the content, and add the node name with “:” manually before the I/O which will be assigned to the node. And then the profile can be piped into the `chvm` command, or changed with the `-p` flag.

For normal power machine:

`chvm` could be used to modify the resources assigned to partitions. The admin shall specify the attributes with options `vmcpus`, `vmmemory`, `add_physlots`, `vmothersetting`, `add_vmnics` and/or `add_vmstorage`. If nothing specified, nothing will be returned.

### zVM specific:

The `chvm` command modifies the virtual machine’s configuration specified in `noderange`.

## OPTIONS

### Common:

**-h**

Display usage message.

**-v**

Command Version.

### PPC (with HMC) specific:

#### **-p** *profile*

Name of an existing partition profile.

#### *attr=val*

Specifies one or more “attribute equals value” pairs, separated by spaces.

#### **-V**

Verbose output.

### PPC (using Direct FSP Management) specific:

#### **--p775**

Specify the operation is for Power 775 machines.

#### **-i**

Starting numeric id of the newly created partitions. For Power 775 using Direct FSP Management, the id value only could be **1, 5, 9, 13, 17, 21, 25** and **29**. Shall work with option **--p775**.

#### **-m**

memory interleaving. The setting value only could be **1** or **2**. **2** means **non-interleaved** mode (also 2MC mode), the memory cannot be shared across the processors in an octant. **1** means **interleaved** mode (also 8MC mode), the memory can be shared. The default value is **1**. Shall work with option **--p775**.

#### **-r**

partition rule. Shall work with option **--p775**.

If all the octants configuration value are same in one CEC, it will be “**-r 0:7:value**”.

If the octants use the different configuration value in one cec, it will be “**-r 0:value1,1:value2,...7:value7**”, or “**-r 0:value1,1-7:value2**” and so on.

The octants configuration value for one Octant could be **1, 2, 3, 4, 5**. The meanings of the octants configuration value are as following:

```
1 -- 1 partition with all cpus and memory of the octant
2 -- 2 partitions with a 50/50 split of cpus and memory
3 -- 3 partitions with a 25/25/50 split of cpus and memory
4 -- 4 partitions with a 25/25/25/25 split of cpus and memory
5 -- 2 partitions with a 25/75 split of cpus and memory
```

#### **-p** *profile*

Name of I/O slots assignment profile. Shall work with option **--p775**.

#### **lparname={\\* | name}**

Set LPAR name for the specified lpars. If ‘\*’ specified, it means to get names from xCAT database and then set them for the specified lpars. If a string is specified, it only supports single node and the string will be set for the specified lpar. The user can use lsvm to check the lparnames for lpars.

#### **vmcpus=value vmmemory=value add\_physlots=value vmothersetting=value**

To specify the parameters that will be modified.

**add\_vmnics=value add\_vmstorage=value [--vios]**

To create new virtual adapter for the specified node.

**del\_physlots=drc\_index1,drc\_index2...**

To delete physical slots which are specified by the *drc\_index1,drc\_index2...*

**del\_vadapter=slotid**

To delete a virtual adapter specified by the *slotid*.

### **VMware/KVM specific:**

**-a size**

Add a new Hard disk with size defaulting to GB. Multiple can be added with comma separated values.

**--cpus count**

Set the number of CPUs.

**-d disk**

Deregister the Hard disk but leave the backing files. Multiple can be done with comma separated values. The disks are specified by SCSI id.

**--mem memory**

Set the memory size for kvm/vmware virtual machines, default unit is MB. Specify in MB or append K for KB, M for MB, or G for GB.

**-p disk**

Purge the Hard disk. Deregisters and deletes the files. Multiple can be done with comma separated values. The disks are specified by SCSI id.

**--resize disk=size**

Change the size of the Hard disk. The disk in *qcow2* format can not be set to less than its current size. The disk in *raw* format can be resized smaller, use caution. Multiple disks can be resized by using comma separated *disk=size* pairs. The disks are specified by SCSI id. Size defaults to GB.

### **KVM specific:**

**--cpupin hostcpuset**

To pin guest domain virtual CPUs to physical host CPUs specified with *hostcpuset*. *hostcpuset* is a list of physical CPU numbers. Its syntax is a comma separated list and a special markup using '-' and '^' (ex. '0-4', '0-3,^2') can also be allowed. The '-' denotes the range and the '^' denotes exclusive.

Note: The expression is sequentially evaluated, so "0-15,^8" is identical to "9-14,0-7,15" but not identical to "^8,0-15".

**--membind numanodeset**

It is possible to restrict a guest to allocate memory from the specified set of NUMA nodes *numanodeset*. If the guest vCPUs are also pinned to a set of cores located on that same set of NUMA nodes, memory access is local and improves memory access performance.

**--devpassthru pcidevice1,pcidevice2...**

The PCI passthrough gives a guest VM direct access to I/O devices *pcidevice1,pcidevice2...*. The PCI devices are assigned to a virtual machine, and the virtual machine can use this I/O exclusively. The devices list are a list of comma separated PCI device names delimited with comma, the PCI device names can be obtained by running **virsh nodedev-list** on the host.

#### **--devdetach pcidevice1,pcidevice2...**

To detaching the PCI devices which are attached to VM guest via PCI passthrough from the VM guest. The devices list are a list of comma separated PCI device names delimited with comma, the PCI device names can be obtained by running **virsh nodedev-list** on the host.

### **zVM specific:**

#### **--add3390** *disk\_pool device\_address size mode read\_password write\_password multi\_password*

Adds a 3390 (ECKD) disk to a virtual machine's directory entry. The device address can be automatically assigned by specifying 'auto'. The size of the disk can be specified in GB, MB, or the number of cylinders.

#### **--add3390active** *device\_address mode*

Adds a 3390 (ECKD) disk that is defined in a virtual machine's directory entry to that virtual server's active configuration.

#### **--add9336** *disk\_pool device\_address size mode read\_password write\_password multi\_password*

Adds a 9336 (FBA) disk to a virtual machine's directory entry. The device address can be automatically assigned by specifying 'auto'. The size of the disk can be specified in GB, MB, or the number of blocks.

#### **--adddisk2pool** *function region volume group*

Add a disk to a disk pool defined in the EXTENT CONTROL. Function type can be either: (4) Define region as full volume and add to group OR (5) Add existing region to group. The disk has to already be attached to SYSTEM.

#### **--addnic** *device\_address type device\_count*

Adds a network adapter to a virtual machine's directory entry (case sensitive).

#### **--addpagespool** *volume\_addr volume\_label volume\_use system\_config\_name system\_config\_type parm\_disk\_owner parm\_disk\_number parm\_disk\_password*

Add a full volume page or spool disk to the virtual machine.

#### **--addprocessor** *device\_address*

Adds a virtual processor to a virtual machine's directory entry.

#### **--addprocessoractive** *device\_address type*

Adds a virtual processor to a virtual machine's active configuration (case sensitive).

#### **--addvdisk** *device\_address size*

Adds a v-disk to a virtual machine's directory entry.

#### **--addzfcv** *pool device\_address loaddev size tag wwpn lun*

Add a zFCP device to a device pool defined in xCAT. The device must have been carved up in the storage controller and configured with a WWP/LUN before it can be added to the xCAT storage pool. z/VM does not have the ability to communicate directly with the storage controller to carve up disks dynamically. xCAT will find a zFCP device in the specified pool that meets the size required, if the WWP/LUN are not given. The device address can be automatically assigned by specifying 'auto'. The WWP/LUN can be set as the LOADDEV in the directory entry if (1) is specified as the 'loaddev'.

**--connectnic2guestlan** *device\_address lan owner*

Connects a given network adapter to a GuestLAN.

**--connectnic2vswitch** *device\_address vswitch*

Connects a given network adapter to a VSwitch.

**--copydisk** *target\_address source\_node source\_address*

Copy a disk attached to a given virtual server.

**--dedicateddevice** *virtual\_device real\_device mode*

Adds a dedicated device to a virtual machine's directory entry.

**--deleteipl**

Deletes the IPL statement from the virtual machine's directory entry.

**--disconnectnic** *device\_address*

Disconnects a given network adapter.

**--formatdisk** *disk\_address multi\_password*

Formats a disk attached to a given virtual server (only ECKD disks supported). The disk should not be linked to any other virtual server. This command is best used after add3390().

**--grantvswitch** *vswitch*

Grant vSwitch access for given virtual machine.

**--purgerdr**

Purge the reader belonging to the virtual machine

**--removedisk** *device\_address*

Removes a minidisk from a virtual machine's directory entry.

**--removenic** *device\_address*

Removes a network adapter from a virtual machine's directory entry.

**--removeprocessor** *device\_address*

Removes a processor from an active virtual machine's configuration.

**--removeloaddev** *wwpn lun*

Removes the LOADDEV statement from a virtual machine's directory entry.

**--removezfcp** *device\_address wwpn lun*

Removes a given SCSI/FCP device belonging to the virtual machine.

**--replacevs** *directory\_entry*

Replaces a virtual machine's directory entry. The directory entry can be echoed into stdin or a text file.

**--setipl** *ipl\_target load\_parms parms*

Sets the IPL statement for a given virtual machine.

**--setpassword** *password*

Sets the password for a given virtual machine.

**--setloaddev** *wwpn lun*



Sets the LOADDEV statement in the virtual machine's directory entry.

**--undedicatdevice** *device\_address*

Delete a dedicated device from a virtual machine's active configuration and directory entry.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

### PPC (with HMC) specific:

1. To change the partition profile for lpar4 using the configuration data in the file /tmp/lparfile, enter:

```
cat /tmp/lparfile | chvm lpar4
```

Output is similar to:

```
lpar4: Success
```

2. To change the partition profile for lpar4 to the existing profile 'prof1', enter:

```
chvm lpar4 -p prof1
```

Output is similar to:

```
lpar4: Success
```

3. To change partition attributes for lpar4 by specifying attribute value pairs in command line, enter:

```
chvm lpar4 max_mem=4096
```

Output is similar to:

```
lpar4: Success
```

### PPC (using Direct FSP Management) specific:

1. For Power 775, to create a new partition lpar1 on the first octant of the cec cec01, lpar1 will use all the cpu and memory of the octant 0, enter:

```
mkdef -t node -o lpar1 mgt=fsp groups=all parent=cec01 nodetype=lpar hcp=cec01
```

then:

```
chvm lpar1 --p775 -i 1 -m 1 -r 0:1
```

Output is similar to:

```
lpar1: Success
cec01: Please reboot the CEC cec1 firstly, and then use chvm to assign the I/O slots to
↳ the LPARs
```

2. For Power 775, to create new partitions lpar1-lpar8 on the whole cec cec01, each LPAR will use all the cpu and memory of each octant, enter:

```
mkdef -t node -o lpar1-lpar8 nodetype=lpar mgt=fsp groups=all parent=cec01 hcp=cec01
```

then:

```
chvm lpar1-lpar8 --p775 -i 1 -m 1 -r 0-7:1
```

Output is similar to:

```
lpar1: Success
lpar2: Success
lpar3: Success
lpar4: Success
lpar5: Success
lpar6: Success
lpar7: Success
lpar8: Success
cec01: Please reboot the CEC cec1 firstly, and then use chvm to assign the I/O slots to
↳ the LPARs
```

3. For Power 775 cec1, to create new partitions lpar1-lpar9, the lpar1 will use 25% CPU and 25% memory of the first octant, and lpar2 will use the left CPU and memory of the first octant. lpar3-lpar9 will use all the cpu and memory of each octant, enter:

```
mkdef -t node -o lpar1-lpar9 mgt=fsp groups=all parent=cec1 nodetype=lpar hcp=cec1
```

then:

```
chvm lpar1-lpar9 --p775 -i 1 -m 1 -r 0:5,1-7:1
```

Output is similar to:

```
lpar1: Success
lpar2: Success
lpar3: Success
lpar4: Success
lpar5: Success
lpar6: Success
lpar7: Success
lpar8: Success
lpar9: Success
cec1: Please reboot the CEC cec1 firstly, and then use chvm to assign the I/O slots to
↳ the LPARs
```

4. To change the I/O slot profile for lpar4 using the configuration data in the file /tmp/lparfile, the I/O slots information is similar to:

```
4: 514/U78A9.001.0123456-P1-C17/0x21010202/2/1
4: 513/U78A9.001.0123456-P1-C15/0x21010201/2/1
4: 512/U78A9.001.0123456-P1-C16/0x21010200/2/1
```

then run the command:

```
cat /tmp/lparfile | chvm lpar4 --p775
```

5. To change the I/O slot profile for lpar1-lpar8 using the configuration data in the file /tmp/lparfile. Users can use the output of lsvm, remove the cec information, modify the lpar id before each I/O, and run the command as following:

```
chvm lpar1-lpar8 --p775 -p /tmp/lparfile
```

6. To change the LPAR name, enter:

```
chvm lpar1 lparname=test_lpar01
```

Output is similar to:

```
lpar1: Success
```

7. For Normal Power machine, to modify the resource assigned to a partition:

Before modify, the resource assigned to node 'lpar1' can be shown with:

```
lsvm lpar1
```

The output is similar to:

```
lpar1: Lpar Processor Info:
Curr Processor Min: 1.
Curr Processor Req: 4.
Curr Processor Max: 16.
lpar1: Lpar Memory Info:
Curr Memory Min: 1.00 GB(4 regions).
Curr Memory Req: 4.00 GB(16 regions).
Curr Memory Max: 32.00 GB(128 regions).
lpar1: 1,513,U78AA.001.WZSGVU7-P1-T7,0x21010201,0xc03(USB Controller)
lpar1: 1,512,U78AA.001.WZSGVU7-P1-T9,0x21010200,0x104(RAID Controller)
lpar1: 1/2/2
lpar1: 128.
```

To modify the resource assignment:

```
chvm lpar1 vmcpus=1/2/16 vmmemory=1G/8G/32G add_physlots=0x21010202
```

The output is similar to:

```
lpar1: Success
```

The resource information after modification is similar to:

```
lpar1: Lpar Processor Info:
Curr Processor Min: 1.
```

(continues on next page)

(continued from previous page)

```
Curr Processor Req: 2.
Curr Processor Max: 16.
lpar1: Lpar Memory Info:
Curr Memory Min: 1.00 GB(4 regions).
Curr Memory Req: 8.00 GB(32 regions).
Curr Memory Max: 32.00 GB(128 regions).
lpar1: 1,514,U78AA.001.WZSGVU7-P1-C19,0x21010202,0xffff(Empty Slot)
lpar1: 1,513,U78AA.001.WZSGVU7-P1-T7,0x21010201,0xc03(USB Controller)
lpar1: 1,512,U78AA.001.WZSGVU7-P1-T9,0x21010200,0x104(RAID Controller)
lpar1: 1/2/2
lpar1: 128.
```

Note: The physical I/O resources specified with *add\_physlots* will be appended to the specified partition. The physical I/O resources which are not specified but belonged to the partition will not be removed. For more information about *add\_physlots*, refer to `lsvm(1)|lsvm.1`.

### VMware/KVM specific:

```
chvm vm1 -a 8,16 --mem 4096 --cpus 2
```

Output is similar to:

```
vm1: node successfully changed
```

### zVM specific:

1. To adds a 3390 (ECKD) disk to a virtual machine's directory entry:

```
chvm gpok3 --add3390 POOL1 0101 2G MR
```

Output is similar to:

```
gpok3: Adding disk 0101 to LNX3... Done
```

2. To add a network adapter to a virtual machine's directory entry:

```
chvm gpok3 --addnic 0600 QDIO 3
```

Output is similar to:

```
gpok3: Adding NIC 0900 to LNX3... Done
```

3. To connects a given network adapter to a GuestLAN:

```
chvm gpok3 --connectnic2guestlan 0600 GLAN1 LN10WNR
```

Output is similar to:

```
gpok3: Connecting NIC 0600 to GuestLan GLAN1 on LN10WNR... Done
```

4. To connects a given network adapter to a vSwitch:

```
chvm gpok3 --connectnic2vswitch 0600 VSW1
```

Output is similar to:

```
gpok3: Connecting NIC 0600 to vSwitch VSW1 on LNX3... Done
```

5. To removes a minidisk from a virtual machine's directory entry:

```
chvm gpok3 --removedisk 0101
```

Output is similar to:

```
gpok3: Removing disk 0101 on LNX3... Done
```

6. To Removes a network adapter from a virtual machine's directory entry:

```
chvm gpok3 --removenic 0700
```

Output is similar to:

```
gpok3: Removing NIC 0700 on LNX3... Done
```

7. To replaces a virtual machine's directory entry:

```
cat /tmp/dirEntry.txt | chvm gpok3 --replacevs
```

Output is similar to:

```
gpok3: Replacing user entry of LNX3... Done
```

8. To resize virtual machine's disk sdb to 10G and sdc to 15G:

```
chvm gpok3 --resize sdb=10G,sdc=15G
```

## FILES

/opt/xcat/bin/chvm

## SEE ALSO

mkvm(1)|mkvm.1, lsvm(1)|lsvm.1, rmvm(1)|rmvm.1

## chzone.1

## NAME

**chzone** - Changes a zone defined in the cluster.

## SYNOPSIS

**chzone** *zonename* [--defaultzone] [-K] [-k *full path to the ssh RSA private key*] [-a *noderange* | -r *noderange*] [-g] [-f]  
[-s {yes|no}] [-V]

**chzone** [-h | -v]

## DESCRIPTION

The **chzone** command is designed to change the definition of a zone previous defined in the cluster. The **chzone** command is only supported on Linux ( No AIX support). The nodes are not updated with the new root ssh keys by **chzone**. You must run **updatenode -k** or **xdsh -K** to the nodes to update the root ssh keys to the new generated zone keys. This will also sync any service nodes with the zone keys, if you have a hierarchical cluster. Note: if any zones in the zone table, there must be one and only one defaultzone. Otherwise, errors will occur.

## OPTIONS

**-h | --help**

Displays usage information.

**-v | --version**

Displays command version and build date.

**-k | --sshkeypath** *full path to the ssh RSA private key*

This is the path to the id\_rsa key that will be used to build new root's ssh keys for the zone. If **-k** is used, it will generate the ssh public key from the input ssh RSA private key, and store both in /etc/xcats/sshkeys/<zonename>/.ssh directory.

**-K | --genkeys**

Using this flag, will generate new ssh RSA private and public keys for the zone into the /etc/xcats/sshkeys/<zonename>/.ssh directory. The nodes are not automatically updated with the new root ssh keys by **chzone**. You must run **updatenode -k** or **xdsh -K** to the nodes to update the root ssh keys to the new generated zone keys. This will also sync any service nodes with the zone keys, if you have a hierarchical cluster.

**--defaultzone**

if **--defaultzone** is input, then it will set the zone defaultzone attribute to yes. if **--defaultzone** is input and another zone is currently the default, then the **-f** flag must be used to force a change to the new defaultzone. If **-f** flag is not use an error will be returned and no change made. Note: if any zones in the zone table, there must be one and only one defaultzone. Otherwise, errors will occur.

**-a | --addnoderange** *noderange*

For each node in the noderange, it will set the zonename attribute for that node to the input zonename. If the **-g** flag is also on the command, then it will add the group name "zonename" to each node in the noderange.

**-r | --rmnoderange** *noderange*

For each node in the noderange, if the node is a member of the input zone, it will remove the zonename attribute for that node. If any of the nodes in the noderange is not a member of the zone, you will get an error and nothing will be changed. If the **-g** flag is also on the command, then it will remove the group name "zonename" from each node in the noderange.

### **-s | --sshbetweennodes yes|no**

If **-s** entered, the zone `sshbetweennodes` attribute will be set to `yes` or `no` based on the input. When this is set to `yes`, then `ssh` will be setup to allow passwordless root access between nodes. If `no`, then root will be prompted for a password when running `ssh` between the nodes in the zone.

### **-f | --force**

Used with the **--defaultzone** flag to override the current default zone.

### **-g | --assigngroup**

Used with the **-a** or **-r** flag to add or remove the group `zonename` for all nodes in the input `noderange`.

### **-V | --verbose**

Verbose mode.

## EXAMPLES

1. To chzone `zone1` to the default zone, enter:

```
chzone> zone1 --default -f
```

2. To generate new root `ssh` keys for `zone2A` using the `ssh id_rsa` private key in `/root/.ssh`:

```
chzone zone2A -k /root/.ssh
```

Note: you must use `xdsh -K` or `updatenode -k` to update the nodes with the new keys

3. To generate new root `ssh` keys for `zone2A`, enter :

```
chzone zone2A -K
```

Note: you must use `xdsh -K` or `updatenode -k` to update the nodes with the new keys

4. To add a new group of nodes (`compute3`) to `zone3` and add `zone3` group to the nodes, enter:

```
chzone zone3 -a compute3 -g
```

5. To remove a group of nodes (`compute4`) from `zone4` and remove `zone4` group from the nodes, enter:

```
chzone> zone4 -r compute4 -g
```

6. To change the `sshbetweennodes` setting on the zone to not allow passwordless `ssh` between nodes, enter:

```
chzone zone5 -s no
```

Note: you must use **xdsh -K** or **updatenode -k** to update the nodes with this new setting.

## FILES

/opt/xcat/bin/chzone/

Location of the chzone command.

## SEE ALSO

mkzone(1)|mkzone.1, rmzone(1)|rmzone.1, xdsh(1)|xdsh.1, updatenode(1)|updatenode.1

## clonevm.1

## NAME

**clonevm** - Create masters from virtual machines and virtual machines from masters.

## SYNOPSIS

**clonevm** *noderange* [ **-t** *master\_to\_be\_made* | **-b** *master\_to\_base\_vms\_upon* ] [ **-d|--detached**] [**-f|--force**]

## DESCRIPTION

Command to promote a VM's current configuration and storage to a master as well as performing the converse operation of creating VMs based on a master.

By default, attempting to create a master from a running VM will produce an error. The **--force** argument will request that a master be made of the VM anyway.

Also, by default a VM that is used to create a master will be rebased as a thin clone of that master. If the **--force** argument is used to create a master of a powered on vm, this will not be done. Additionally, the **--detached** option can be used to explicitly request that a clone not be tethered to a master image, allowing the clones to not be tied to the health of a master, at the cost of additional storage.

When promoting a VM's current state to master, all related virtual disks will be copied and merged with any prerequisite images. A master will not be tethered to other masters.

## OPTIONS

### **-h|--help**

Display usage message.

### **-b** *master\_to\_base\_vms\_upon*

The master to base the clones upon

### **-t** *master\_to\_be\_made*

The target master to copy a single VM's state to

### **-d|--detached**

Explicitly request that the noderange be untethered from any masters.



### **-f|--force**

Force cloning of a powered on VM. Implies **-d** if the VM is on.

### **-v|--version**

Command Version.

### **-V|--verbose**

Verbose output.

## **RETURN VALUE**

0: The command completed successfully.

Any other value: An error has occurred.

## **EXAMPLES**

1. Creating a master named *appserver* from a node called *vm1*:

```
clonevm vm1 -t appserver
```

2. Cleating 30 VMs from a master named *appserver*:

```
clonevm vm1-vm30 -b appserver
```

## **FILES**

/opt/xcat/bin/clonevm

## **SEE ALSO**

chvm(1)|chvm.1, lsvm(1)|lsvm.1, rmvm(1)|rmvm.1, mkvm(1)|mkvm.1, vmmaster(5)|vmmaster.5

## **configfpc.1**

## **NAME**

**configfpc** - discover the Fan Power Controllers (FPCs) and configure the FPC interface

## SYNOPSIS

**configfpc** **-i** *interface*

**configfpc** **-i** *interface* **--ip** *default ip address*

**configfpc** [**-V** | **--verbose**]

**configfpc** [**-h** | **--help** | **-?**]

## DESCRIPTION

**configfpc** will discover and configure all FPCs that are set to the default IP address. If not supplied the default ip is 192.168.0.100.

The **-i interface** is required to direct **configfpc** to the xCAT MN interface which is on the same VLAN as the FPCs.

There are several bits of information that must be included in the xCAT database before running this command.

You must create the FPC node definitions for all FPCs being discovered including the IP address and switch port information.

The **configfpc** command discovers the FPCs and collects the MAC address. The MAC address is used to relate the FPC to a FPC node using the switch information for this MAC. Once the relationship is discovered the FPC is configured with the FPC node IP settings.

This process is repeated until no more FPCs are discovered.

For more information on xCAT support of NeXtScale and configfpc see the following doc: [XCAT\\_NeXtScale\\_Clusters](#)

## OPTIONS

**-i interface**

Use this flag to specify which xCAT MN interface (example: eth4) that is connected to the NeXtScale FPCs. This option is required.

**--ip default ip address**

Use this flag to override the default ip address of 192.168.0.100 with a new address.

**-V** | **--verbose**

Verbose mode

## EXAMPLES

1. To discover and configure all NeXtScale Fan Power Controllers (FPCs) connected on eth0 interface.

```
configfpc -i eth0
```

2. To override the default ip address and run in Verbose mode.

```
configfpc -i eth0 --ip 196.68.0.100 -V
```

## csm2xcat.1

### NAME

**csm2xcat** - Allows the migration of a CSM database to an xCAT database.

### SYNOPSIS

**csm2xcat** [--dir *path*]

**csm2xcat** [-h]

### DESCRIPTION

The **csm2xcat** command must be run on the Management Server of the CSM system that you want to migrate to xCAT. The command will build two xCAT stanza files that can update the xCAT database with the **chdef** command.

Copy the **csm2xcat** command to the CSM Management Server. Run the command, indicating where you want your stanza files saved with the **--dir** parameter. Check the stanza files to see if the information is what you want put in the xCAT database. Copy the two stanza files: **node.stanza**, **device.stanza** back to your xCAT Management node, and run the **chdef** command to input into the xCAT database.

### OPTIONS

**-h** Display usage message.

**--dir** Path to the directory containing the stanza files.

### RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

### EXAMPLES

1. To build xCAT stanza files, enter on the CSM Management Server:

```
csm2xcat --dir /tmp/mydir
```

2. To put the data in the xCAT database on the xCAT Management Node:

```
cat node.stanza | chdef -z
cat device.stanza | chdef -z
```

## FILES

/opt/xcat/share/xcat/tools/csm2xcat

\$dir/conversion.log

## SEE ALSO

chdef(1)|chdef.1

## db2sqlsetup.1

## NAME

**db2sqlsetup** - Sets up the IBM DB2 for xCAT to use.

## SYNOPSIS

**db2sqlsetup** [-h | --help]

**db2sqlsetup** [-v | --version]

**db2sqlsetup** [-i | --init] {-S | -C} [-o | --setupODBC] [-V | --verbose]

**db2sqlsetup** [-i | --init] {-S} [-N | --nostart] [-o | --setupODBC] [-V | --verbose]

**db2sqlsetup** [-o | --setupODBC] {-S | -C} [-V | --verbose]

**db2sqlsetup** [-p | --passwd] [-S | -C]

## DESCRIPTION

**db2sqlsetup** - Sets up the IBM DB2 database for xCAT to use. The db2sqlsetup script is run on the Management Node, after the DB2 Server code has been installed, to setup the DB2 Server (-S). The xcatd daemon will be stopped during migration on the MN. No xCAT commands should be run during the init process, because we will be migrating the xCAT database to DB2 and restarting the xcatd daemon.

The db2sqlsetup script must be run on each Service Node, after the DB2 Client code has been installed, to setup the DB2 Client (-C). There are two postscripts that are provided ( db2install and odbcsetup) that will automatically setup you Service Node as a DB2 client.

For full information on the setup of DB2, see Setting\_Up\_DB2\_as\_the\_xCAT\_DB.

When running of db2sqlsetup on the MN: One password must be supplied for the setup, a password for the xcatdb unix id which will be used as the DB2 instance id and database name. The password will be prompted for interactively or can be input with the XCATDB2PW environment variable. The script will create the xcat database instance (xcatdb) in the /var/lib/db2 directory unless overridden by setting the site.databasesloc attribute. This attribute should not be set to the directory that is defined in the installloc attribute and it is recommended that the databasesloc be a new filesystem dedicated to the DB2 database, especially in very large clusters.

When running db2sqlseutp on the SN: Not only will the password for the DB2 instance Id be prompted for and must match the one on the Management Node; but also the hostname or ip address of the Management Node as known by the Service Node must be supplied , unless the XCATDB2SERVER environment variable is set. You can automatically

install and setup of DB2 on the SN using the db2install and odbctestup postscripts and not need to manually run the command. See the full documentation.

Note: On AIX , root must be running ksh and on Linux, bash shell.

## OPTIONS

### **-h|--help**

Displays the usage message.

### **-v|--version**

Displays the release version of the code.

### **-V|--verbose**

Displays verbose messages.

### **-i|--init**

The init option is used to setup an installed DB2 database on AIX or Linux (p-Series) so that xCAT can use the database. This must be combined with either the -S or -C flag to indicate whether we are setting up the Server or the Client. With the -S flag, it involves creating the xcatdb database, the xcatdb instance id, allowing access to the xcatdb database by the Management Node. It also backs up the current xCAT database and restores it into the newly setup xcatdb DB2 database. It creates the /etc/xcat/cfgloc file to point the xcatd daemon to the DB2 database and restarts the xcatd daemon using the database.

### **-p|--passwd**

The password change option is to change the database access password for the DB2 xcatdb database. If -S is input then it will only change the password on the DB2 Server (MN). If -C is input it will only change on the DB2 clients (SN). If neither -S or -C are input with this flag, then it will change both the DB2 Server and Clients. When changing the password the xcatd daemon will be stopped and restarted. Any other tools accessing the database should also be stopped before changing and restarted after changing.

### **-S|-C**

This options says whether to setup the Server (-S) on the Management Node, or the Client (-C) on the Service Nodes.

### **-N|--nostart**

This option with the -S flag will create the database, but will not backup and restore xCAT tables into the database. It will create the cfgloc file such that the next start of xcatd will try and contact the database. This can be used to setup the xCAT DB2 database during or before install.

### **-o|--setupODBC**

This option sets up the ODBC /etc/./odbcinst.ini, /etc/./odbc.ini and the .odbc.ini file in roots home directory will be created and initialized to run off the xcatdb DB2 database.

## ENVIRONMENT VARIABLES

\* XCATDB2INSPATH overrides the default install path for DB2 which is /opt/ibm/db2/V9.7 for Linux and /opt/IBM/db2/V9.7 for AIX.

\* DATABASELOC override the where to create the xcat DB2 database, which is /var/lib/db2 by default of taken from the site.databasesloc attribute.

\* XCATDB2PW can be set to the password for the xcatdb DB2 instance id so that there will be no prompting for a password when the script is run.

## EXAMPLES

1. To setup DB2 Server for xCAT to run on the DB2 xcatdb database, on the MN:

```
db2sqlsetup -i -S
```

2. To setup DB2 Client for xCAT to run on the DB2 xcatdb database, on the SN:

```
db2sqlsetup -i -C
```

3. To setup the ODBC for DB2 xcatdb database access, on the MN :

```
db2sqlsetup -o -S
```

4. To setup the ODBC for DB2 xcatdb database access, on the SN :

```
db2sqlsetup -o -C
```

5. To setup the DB2 database but not start xcat running with it:

```
db2sqlsetup -i -S -N
```

6. To change the DB2 xcatdb password on both the Management and Service Nodes:

```
db2sqlsetup -p
```

## dumpxCATdb.1

### NAME

**dumpxCATdb** - dumps the xCAT db tables .

## SYNOPSIS

```
dumpxCATdb [-a] [-V] [{-p | --path} path]
```

```
dumpxCATdb [-b] [-V] [{-p | --path} path]
```

```
dumpxCATdb [-h | --help] [-v | --version]
```

## DESCRIPTION

If not using the binary dump option (-b), then the dumpxCATdb command creates .csv files for xCAT database tables and puts them in the directory given by the -p flag. These files can be used by the restorexCATdb command to restore the database. The command will read the list of tables in the site.skiptables attribute and not backup those tables. Supports using XCAT\_SKIPTABLES env variable to provide a list of skip tables. The command will never backup TEAL or ISNM tables, except isnm\_config. To dump TEAL tables use the documented process for TEAL. For ISNM use tabdump, after using tabprune to get to prune unnecessary records.

If using the binary dump option for the DB2 or PostgreSQL database, then the routine will use the Database provide utilities for backup of the entire database.

## OPTIONS

**-h** Display usage message.

**-v** Command Version.

**-V** Verbose.

**-a** All, without this flag the eventlog and auditlog will be skipped.

**-b** This flag is only used for the DB2 or PostgreSQL database. The routine will use the database backup utilities to create a binary backup of the entire database. Note to use this backup on DB2, you will have first had to modify the logging of the database and have taken an offline initial backup. Refer to the xCAT DB2 documentation for more instructions.

**-p** Path to the directory to dump the database. It will be created, if it does not exist.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To dump the xCAT database into the /tmp/db directory, enter:

```
dumpxCATdb -p /tmp/db
```

2. To dump the xCAT database into the /tmp/db directory, including the auditlog and eventlog enter:

```
dumpxCATdb -a -p /tmp/db
```

3. To have dumpxCATdb not backup the hosts or passwd table:

```
chtab key=skiptables site.value="hosts,passwd"
```

```
dumpxCATdb -p /tmp/db
```

4. To have dumpxCATdb not backup the hosts or passwd table:

```
export XCAT_SKIPTABLES="hosts,passwd"
```

```
dumpxCATdb -p /tmp/db
```

5. To have dumpxCATdb use DB2 utilities to backup the DB2 database:

```
dumpxCATdb -b -p /install/db2backup
```

## FILES

/opt/xcatsbin/dumpxCATdb

## SEE ALSO

restorexCATdb(1)|restorexCATdb.1

## genimage.1

## NAME

**genimage** - Generates a stateless image to be used for a diskless install.

## SYNOPSIS

### genimage

**genimage** [-o *osver*] [-a *arch*] [-p *profile*] [-i *nodebootif*] [-n *nodenetdrivers*] [--onlyinitrd] [-r *otherifaces*] [-k *kernelver*] [-g *krpmver*] [-m *statelite*] [-l *rootlimitsize*] [--permission *permission*] [--interactive] [--dryrun] [--ignorekernelchk] [--nouupdate] *imagename*

**genimage** [-h | --help | -v | --version]

## DESCRIPTION

Generates a stateless and a statelite image that can be used to boot xCAT nodes in a diskless mode.

**genimage** will use the *osimage* definition for information to generate this image. Additional options specified on the command line will override any corresponding previous *osimage* settings and will be written back to the *osimage* definition.

If **genimage** runs on the management node, both the *osimage* table and *linuximage* table will be updated with the given values from the options.

The **genimage** command will generate two initial ramdisks, **initrd-stateless.gz** for **stateless** mode, and **initrd-statelite.gz** for **statelite** mode.



After your image is generated, you can chroot to the image, install any additional software you would like, or make modifications to files, and then run the following command to prepare the image for deployment.

for stateless: **packimage**

for statelite: **liteimg**

Besides prompting for some parameter values, the **genimage** command takes default guesses for the parameters not specified or not defined in the *osimage* and *linuximage* tables. It also assumes default answers for questions from the yum/zypper command when installing rpms into the image. Use **--interactive** flag if you want the yum/zypper command to prompt you for the answers.

If **--onlyinitrd** is specified, genimage only regenerates the initrd for a stateless image to be used for a diskless install.

The **genimage** command must be run on a system that is the same architecture and same distro with same major release version as the nodes it will be used on. If the management node is not the same architecture or same distro level, copy the contents of /opt/xcat/share/xcat/netboot/<os> to a system that is the proper architecture, and mount /install from the management node to that system. Then change directory to /opt/xcat/share/xcat/netboot/<os> and run ./genimage.

## Parameters

*imagename* specifies the name of an os image definition to be used. The specification for the image is stored in the *osimage* table and *linuximage* table.

## OPTIONS

### -a *arch*

The hardware architecture of this node: ppc64le, x86\_64, ppc64, x86, ia64, etc. If omitted, the current hardware architecture will be used.

### -o *osver*

The operating system for the image: rhels8.2.0, sle15, ubuntu18.04.2, etc. The OS packages must be in /install/<osver>/<arch> (use copycds(8)|copycds.8).

### -p *profile*

The profile (e.g. compute, service) to use to create the image. This determines what package lists are used from /opt/xcat/share/xcat/netboot/<os> to create the image with. When deploying nodes with this image, the nodes' nodetype.profile attribute must be set to this same value.

### -i *nodebootif*

This argument is now optional, and allows you to specify the network boot interface to be configured in the image (e.g. eth0). If not specified, the interface will be determined and configured during the network boot process.

### -n *nodenetdrivers*

This argument is now optional, and allows you to specify the driver modules needed for the network interface(s) on your stateless nodes. If you do not specify this option, the default is to include all recent IBM xSeries network drivers.

If specified, *nodenetdrivers* should be a comma separated list of network drivers to be used by the stateless nodes (I.e.: -n tg3,e1000). Note that the drivers will be loaded in the order that you list them, which may prove important in some cases.

### -l *rootlimit*

The maximum size allowed for the root file system in the image. Specify in bytes, or can append k, m, or g.

**--onlyinitrd**

Regenerates the initrd for a stateless image to be used for a diskless install.

Regenerates the initrd that is part of a stateless/statelite image that is used to boot xCAT nodes in a stateless/statelite mode.

The **genimage --onlyinitrd** command will generate two initial ramdisks, **initrd-stateless.gz** for **stateless** mode, and **initrd-statelite.gz** for **statelite** mode.

**--permission *permission***

The mount permission of **/.statelite** directory for **statelite** mode, which is only used for **statelite** mode, and the default permission is 755.

**-r *otherifaces***

Other network interfaces (e.g. eth1) in the image that should be configured via DHCP.

**-k *kernelver***

Use this flag if you want to use a specific version of the kernel in the image. Defaults to the first kernel found in the install image.

**-g *krpmver***

Use this flag to specify the rpm version for kernel packages in the image. It must be present if -k flag is specified in the command for SLES. Generally, the value of -g is the part after **linux-** and before **.rpm** in a kernel rpm name.

**-m *statelite***

This flag is for Ubuntu, Debian and Fedora12 only. Use this flag to specify if you want to generate statelite image. The default is to generate stateless image for these three operating systems. For others, this flag is invalid because both stateless and statelite images will be generated with this command.

**--interactive**

This flag allows the user to answer questions from yum/zypper command when installing rpms into the image. If it is not specified, '-y' will be passed to the yum command and '--non-interactive --no-gpg-checks' will be passed to the zypper command as default answers.

**--dryrun**

This flag shows the underlying call to the os specific genimage function. The user can copy and the paste the output to run the command on another machine that does not have xCAT installed.

**-t *tmplimit***

(Deprecated) This flag allows the user to setup the /tmp and the /var/tmp file system sizes. This flag is no longer supported. You can overwrite any file system size using the .postinstall script where you can create a new /etc/fstab file.

**--ignorekernelchk**

Skip the kernel version checking when injecting drivers from osimage.driverupdatesrc. That means all drivers from osimage.driverupdatesrc will be injected to initrd for the specific target kernel.

**--noudate**

This flag allows the user to bypass automatic package updating when installing other packages.

**-v|--version**

Display version.

**-h|--help**

Display usage message.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1 To prompt the user for inputs:

```
genimage
```

2 To generate an image using information from an osimage definition:

```
genimage myimagename
```

3 To run genimage in test mode without actually generating an image:

```
genimage --dryrun myimagename
```

4 To generate an image and have yum/zypper prompt for responses:

```
genimage myimagename --interactive
```

5 To generate an image, replacing some values in the osimage definition:

```
genimage -i eth0 -n tg3 myimagename
```

## FILES

/opt/xcat/bin/genimage

/opt/xcat/share/xcat/netboot/<OS>/genimage

## SEE ALSO

packimage(1)|packimage.1, liteimg(1)|liteimg.1

## geninitrd.1

### NAME

**geninitrd** - Generate an initrd (initial ramfs) which to be used for stateful install or stateless netboot.

### SYNOPSIS

**geninitrd** *imagename* [--ignorekernelchk]

**geninitrd** [-h | --help]

### DESCRIPTION

Generate the initrd for the osimage: **imagename** which is an xCAT object of *osimage* type.

#### Diskful Osimage

If the **imagename** is a stateful one (The provmethod attribute for the osimage is 'install'), this command is used to rebuild the initrd to inject the new drivers from driver rpms or 'update distro' and copy the rebuilt initrd and new kernel (If there's new kernel in 'update distro') to the directory */tftpboot/xcat/<imagename>*.

If the initrd has been rebuilt by geninitrd, when run nodeset, the *-nouupdateinitrd* option should be used to skip the rebuilding of initrd to improve the performance.

Three attributes of osimage object can be used to specify the Driver RPM location and Driver names for injecting new drivers to initrd.

**netdrivers** - comma separated driver names that need to be injected to the initrd. The postfix '.ko' can be ignored. The netdrivers attribute must be set to specify the new driver list. If you want to load all the drivers from the driver rpms, using the keyword allupdate.

**driverupdatesrc** - comma separated driver rpm packages (full path should be specified)

**osupdatename** - comma separated 'osdistrouupdate' object. Each 'osdistrouupdate' object specifies a Linux distro update. When run geninitrd, 'kernel-\*.rpm' will be searched from osdistrouupdate.dirpath to get all the rpm packages and then search the drivers from the rpm packages.

Refer to the doc: *Using\_Linux\_Driver\_Update\_Disk*

#### Stateless Osimage

If the **imagename** is a stateless one (The provmethod attribute for the osimage is 'netboot'), this command is used to generate the initrd from the rooting which generated by 'genimage' command. So the 'genimage' must be run once before running the geninitrd command.

Two attributes of osimage object can be used to specify the Driver RPM location and Driver names for injecting new drivers to initrd.

**netdrivers** - comma separated driver names that need to be injected to the initrd. The postfix '.ko' can be ignored. The netdrivers attribute must be set to specify the new driver list. If you want to load all the drivers from the driver rpms, using the keyword allupdate.

**driverupdatesrc** - comma separated driver rpm packages (full path should be specified)

## Parameters

*imagename* specifies the name of an os image definition to be used. The specification for the image is stored in the *osimage* table and *linuximage* table.

### **--ignorekernelchk**

Skip the kernel version checking when injecting drivers from *osimage.driverupdatesrc*. That means all drivers from *osimage.driverupdatesrc* will be injected to *initrd* for the specific target kernel.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1 To generate *initrd* for the *osimage* **myimagename**:

```
geninitrd myimagename
```

## FILES

/opt/xcat/bin/geninitrd

/opt/xcat/bin/genimage

/opt/xcat/share/xcat/netboot/<OS>/genimage

## SEE ALSO

*geninitrd*(1)|*geninitrd*.1, *genimage*(1)|*genimage*.1

### **getmacs.1**

## NAME

**getmacs** - Collects node MAC address.

## SYNOPSIS

### **Common:**

**getmacs** [-h] --help | -v | --version]

### PPC specific:

**getmacs** *noderange* [-F *filter*]

**getmacs** *noderange* [-M]

**getmacs** *noderange* [-V] [--verbose] [-f] [-d] [--arp] | [-D {[**-S** *server*] [**-G** *gateway*] [**-C** *client*] [**-o**] | [--noping]}]

### blade specific:

**getmacs** *noderange* [-V] [--verbose] [-d] [--arp] [-i *ethN* | *enN*]

## DESCRIPTION

The **getmacs** command collects MAC address from a single or range of nodes. Note that on AIX systems, the returned MAC address is not colon-separated (for example 8ee2245cf004), while on Linux systems the MAC address is colon-separated (for example 8e:e2:24:5c:f0:04). If no ping test performed, **getmacs** writes the first adapter MAC to the xCAT database. If ping test performed, **getmacs** will write the first successfully pinged MAC to xCAT database.

For PPC (using Direct FSP Management) specific:

Note: If network adapters are physically assigned to LPARs, **getmacs** cannot read the MAC addresses unless perform **Discovery** with option “**-D**”, since there is no HMC command to read them and **getmacs** has to login to open firmware. And if the LPARs has never been activated before, **getmacs** need to be performed with the option “**-D**” to get their MAC addresses.

For PPC (using HMC) specific:

Note: The option “**-D**” **must** be used to get MAC addresses of LPARs.

For IBM Flex Compute Node (Compute Node for short) specific:

Note: If “**-d**” is specified, all the MAC of the blades will be displayed. If no option specified, the first MAC address of the blade will be written to mac table.

## OPTIONS

### --arp

Read MAC address with ARP protocol.

### -C

Specify the IP address of the partition for ping test. The default is to read from xCAT database if no **-C** specified.

### -d

Display MAC only. The default is to write the first valid adapter MAC to the xCAT database.

### -D

Perform discovery for mac address. By default, it will run ping test to test the connection between adapter and xCAT management node. Use ‘--noping’ can skip the ping test to save time. Be aware that in this way, the lpar will be reset.

### -f

Force immediate shutdown of the partition. This flag must be used with **-D** flag.

### -F

Specify filters to select the correct adapter. Acceptable filters are Type, MAC\_Address, Phys\_Port\_Loc, Adapter, Port\_Group, Phys\_Port, Logical\_Port, VLAN, VSwitch, Curr\_Conn\_Speed.

**-G**

Gateway IP address of the partition. The default is to read from xCAT database if no **-G** specified.

**-h**

Display usage message.

**-M**

Return multiple MAC addresses for the same adapter or port, if available from the hardware. For some network adapters (e.g. HFI) the MAC can change when there are some recoverable internal errors. In this case, the hardware can return several MACs that the adapter can potentially have, so that xCAT can put all of them in DHCP. This allows successful booting, even after a MAC change, but on Linux at this time, it can also cause duplicate IP addresses, so it is currently not recommended on Linux. By default (without this flag), only a single MAC address is returned for each adapter.

**--noping**

Only can be used with '**-D**' to display all the available adapters with mac address but do NOT run ping test.

**-o**

Read MAC address when the lpar is in openfirmware state. This option must be used with [**-D**] option to perform ping test. Before use **-o**, the lpar must be in openfirmware state.

**-S**

The IP address of the machine to ping. The default is to read from xCAT database if no **-S** specified.

**-v**

Command Version.

**-V**

Verbose output.

**-i**

Specify the interface whose mac address will be collected and written into mac table. If 4 mac addresses are returned by option '**-d**', they all are the mac addresses of the blade. The N can start from 0(map to the eth0 of the blade) to 3. If 5 mac addresses are returned, the 1st mac address must be the mac address of the blade's FSP, so the N will start from 1(map to the eth0 of the blade) to 4.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To retrieve the MAC address for the HMC-managed partition lpar4 and write the first valid adapter MAC to the xCAT database, enter:

```
getmacs lpar4
```

Output is similar to:

```
lpar4:
#Type  MAC_Address  Phys_Port_Loc  Adapter  Port_Group  Phys_Port  Logical_Port  Vlan  ↵
↵VSwitch  Curr_Conn_Speed
hea  7607DFB07F02  N/A  N/A  N/A  N/A  N/A  1  ETHERNET0  N/A
ent  U78A1.001.99203B5-P1-T6  00145eb55788  /lhea@23c00614/ethernet@23e00514  unsuccessful↵
↵physical
```

2. To retrieve the MAC address with ARP protocol:

```
getmacs lpar4 --arp
```

Output is similar to:

```
lpar4:
#IP          MAC_Address
192.168.0.10  00145eb55788
```

3. To retrieve the MAC address for the HMC-managed partition lpar4 and display the result only, enter:

```
getmacs lpar4 -d
```

Output is similar to:

```
lpar4:
#Type  MAC_Address  Phys_Port_Loc  Adapter  Port_Group  Phys_Port  Logical_Port  Vlan  ↵
↵VSwitch  Curr_Conn_Speed
hea  7607DFB07F02  N/A  N/A  N/A  N/A  N/A  1  ETHERNET0  N/A
ent  U78A1.001.99203B5-P1-T6  00145eb55788  /lhea@23c00614/ethernet@23e00514  unsuccessful↵
↵physical
```

4. To retrieve the MAC address for the HMC-managed partition lpar4 with filter Type=hea,VSwitch=ETHERNET0.

```
getmacs lpar4 -F Type=hea,VSwitch=ETHERNET0
```

Output is similar to:

```
lpar4:
#Type  MAC_Address  Phys_Port_Loc  Adapter  Port_Group  Phys_Port  Logical_Port  Vlan  ↵
↵VSwitch  Curr_Conn_Speed
hea  7607DFB07F02  N/A  N/A  N/A  N/A  N/A  1  ETHERNET0  N/A
```

5. To retrieve the MAC address while performing a ping test for the HMC-managed partition lpar4 and display the result only, enter:

```
getmacs lpar4 -d -D -S 9.3.6.49 -G 9.3.6.1 -C 9.3.6.234
```

Output is similar to:



```
lpar4:
#Type Location Code MAC Address Full Path Name Ping Result
ent U9133.55A.10B7D1G-V12-C4-T1 8e:e2:24:5c:f0:04 /vdevice/l-lan@30000004 successful
↳virtual
```

- To retrieve the MAC address for Power 775 LPAR using Direct FSP Management without ping test and display the result only, enter:

```
getmacs lpar4 -d
```

Output is similar to:

```
lpar4:
#Type Phys_Port_Loc MAC_Address Adapter Port_Group Phys_Port Logical_Port Vlan
↳VSwitch Curr_Conn_Speed
HFI N/A 02:00:02:00:00:04 N/A N/A N/A N/A N/A N/A
```

- To retrieve multiple MAC addresses from Power 775 HFI network adapter using Direct FSP Management, enter:

```
getmacs lpar4 -M
```

Output is similar to:

```
lpar4:
#Type Phys_Port_Loc MAC_Address Adapter Port_Group Phys_Port Logical_Port Vlan
↳VSwitch Curr_Conn_Speed
HFI N/A 02:00:02:00:00:04|02:00:02:00:00:05|02:00:02:00:00:06 N/A N/A N/A N/A N/A
↳ N/A N/A
```

- To retrieve the MAC address for Power Lpar by '-D' but without ping test.

```
getmacs lpar4 -D --noping
```

Output is similar to:

```
lpar4:
# Type Location Code MAC Address Full Path Name Device Type
ent U8233.E8B.103A4DP-V3-C3-T1 da:08:4c:4d:d5:03 /vdevice/l-lan@30000003 virtual
ent U8233.E8B.103A4DP-V3-C4-T1 da:08:4c:4d:d5:04 /vdevice/l-lan@30000004 virtual
ent U78A0.001.DNWHYT2-P1-C6-T1 00:21:5e:a9:50:42 /lhea@2000000000000000/
↳ethernet@20000000000000003 physical
```

## FILES

/opt/xcat/bin/getmacs

## SEE ALSO

makedhcp(8)|makedhcp.8

## getsnodes.1

## NAME

**getsnodes** - queries your SoftLayer account and gets attributes for each server.

## SYNOPSIS

**getsnodes** [-v | --verbose] [*hostname-match*]

**getsnodes** [-? | -h | --help]

## DESCRIPTION

The **getsnodes** command queries your SoftLayer account and gets attributes for each server. The attributes can be piped to 'mkdef -z' to define the nodes in the xCAT DB so that xCAT can manage them.

Before using this command, you must download and install the SoftLayer API perl module. For example:

```
cd /usr/local/lib
git clone https://github.com/softlayer/softlayer-api-perl-client.git
```

You also need to follow these directions to get your SoftLayer API key: <http://knowledgelayer.softlayer.com/procedure/retrieve-your-api-key>

**getsnodes** requires a .slconfig file in your home directory that contains your SoftLayer userid, API key, and location of the SoftLayer API perl module, in attr=val format. For example:

```
# Config file used by the xcat cmd getsnodes
userid = joe_smith
apikey = 1234567890abcdef1234567890abcdef1234567890abcdef
apidir = /usr/local/lib/softlayer-api-perl-client
```

## OPTIONS

**-?|-h|--help**

Display usage message.

**-v|--version**

Command Version.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. Display information about all of the nodes in your SoftLayer account:

```
getslnodes
```

2. Display information about all of the nodes whose hostname starts with foo:

```
getslnodes foo
```

3. Create xCAT node definitions in the xCAT DB for all of the nodes in your SoftLayer account:

```
getslnodes | mkdef -z
```

## FILES

/opt/xcat/bin/getslnodes

## SEE ALSO

pushinitrd(1)|pushinitrd.1

## gettab.1

## NAME

**gettab** - select table rows, based on attribute criteria, and display specific attributes.

## SYNOPSIS

**gettab** [-H | --with-fieldname] *key=value,... table.attribute ...*

**gettab** [-? | -h | --help]

## DESCRIPTION

The **gettab** command uses the specified key values to select a row in each of the tables requested. For each selected row, the specified attributes are displayed. The **gettab** command can be used instead of **nodels** for tables that are not keyed by nodename (e.g. the **site** table), or to select rows based on an attribute value other than nodename.

## OPTIONS

### **-H|--with-fieldname**

Always display table.attribute name next to result. By default, this is done only if more than one table.attribute is requested.

### **-?|-h|--help**

Display usage message.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To display setting for **master** (management node) in the site table:

```
gettab -H key=master site.value
```

The output would be similar to:

```
site.value: mgmtnode.cluster.com
```

2. To display the first node or group name that has **mgt** set to **blade** in the nodehm table:

```
gettab mgt=blade nodehm.node
```

The output would be similar to:

```
blades
```

## FILES

/opt/xcat/bin/gettab

## SEE ALSO

models(1)|models.1, chtab(8)|chtab.8, tabdump(8)|tabdump.8

## groupfiles4dsh.1

## NAME

**groupfiles4dsh** - Builds a directory of files for each defined nodegroup in xCAT.

## SYNOPSIS

**groupfiles4dsh** [{-p | --path} *path*]

**groupfiles4dsh** [-h | --help] [-v | --version]

## DESCRIPTION

This tool will build a directory of files, one for each defined nodegroup in xCAT. The file will be named the nodegroup name and contain a list of nodes that belong to the nodegroup. The file can be used as input to the AIX dsh command. The purpose of this tool is to allow backward compatibility with scripts that were created using the AIX or CSM dsh command

Reference: man dsh.

## OPTIONS

**-h** Display usage message.

**-v** Command Version.

**-p** Path to the directory to create the nodegroup files (must exist).

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To create the nodegroup files in directory /tmp/nodegroupfiles, enter:

```
groupfiles4dsh -p /tmp/nodegroupfiles
```

To use with dsh:

```
export DSH_CONTEXT=DSH ( default unless CSM is installed)
export DSH_NODE_RSH=/bin/ssh (default is rsh)
export DSH_NODEGROUP_PATH= /tmp/nodegroupfiles

dsh -N all date (where all is a group defined in xCAT)
dsh -a date (will look in all nodegroupfiles and build a list of all nodes)
```

## FILES

/opt/xcat/share/xcat/tools/groupfiles4dsh

## SEE ALSO

xdsh(1)|xdsh.1

## imgcapture.1

## NAME

**imgcapture** - Captures an image from a Linux diskful node and create a diskless or diskful image on the management node.

## SYNOPSIS

**imgcapture** *node* **-t** | **--type** {**diskless** | **sysclone**} **-o** | **--osimage** *osimage* [**-i** *nodebootif*] [**-n** *nodenetdrivers*] [**-V** | **--verbose**]

**imgcapture** [**-h** | **--help**] | [**-v** | **--version**]

## DESCRIPTION

The **imgcapture** command will capture an image from one running diskful Linux node and create a diskless or diskful image for later use.

The **node** should be one diskful Linux node, managed by the xCAT MN, and the remote shell between MN and the **node** should have been configured. AIX is not supported. VMs are not supported.

The **imgcapture** command supports two image types: **diskless** and **sysclone**. For the **diskless** type, it will capture an image from one running diskful Linux node, prepares the rootimg directory, kernel and initial ramdisks for the **liteimg/packimage** command to generate the statelite/stateless rootimg. For the **sysclone** type, it will capture an image from one running diskful Linux node, create an osimage which can be used to clone other diskful Linux nodes.

The **diskless** type:

The attributes of osimage will be used to capture and prepare the root image. The **osver**, **arch** and **profile** attributes for the stateless/statelite image to be created are duplicated from the **node**'s attribute. If the **-p|--profile** *profile* option is specified, the image will be created under “/<installroot>/netboot/<osver>/<arch>/<profile>/rootimg”.

The default files/directories excluded in the image are specified by `/opt/xcat/share/xcat/netboot/<os>/<profile>.<osver>.<arch>.imgcapture`, also, you can put your customized file (`<profile>.<osver>.<arch>.imgcapture.exlist`) to `/install/custom/netboot/<osplatform>`. The directories in the default `.imgcapture.exlist` file are necessary to capture the image from the diskful Linux node managed by xCAT, don't remove it.

The image captured will be extracted into the `/<installroot>/netboot/<osver>/<arch>/<profile>/rootimg` directory.

After the **imgcapture** command returns without any errors, you can customize the rooting and run the **liteimg/packimage** command with the options you want.

The **sysclone** type:

xCAT leverages the Open Source Tool - Systemimager to capture the osimage from the **node**, and put it into `/<installroot>/sysclone/images` directory.

The **imgcapture** command will create the *osimage* definition after the image is captured successfully, you can use this osimage and **nodeset** command to clone diskful nodes.

## OPTIONS

### **-t | --type**

Specify the osimage type you want to capture, two types are supported: diskless and sysclone.

### **-p|--profile *profile***

Assign *profile* as the profile of the image to be created.

### **-o|--osimage *osimage***

The osimage name.

### **-i *nodebootif***

The network interface the diskless node will boot over (e.g. eth0), which is used by the **genimage** command to generate initial ramdisks.

### **-n *nodenetdrivers***

The driver modules needed for the network interface, which is used by the **genimage** command to generate initial ramdisks.

By default, the **genimage** command can provide drivers for the following network interfaces:

For x86 or x86\_64 platform:

```
tg3 bnx2 bnx2x e1000 e1000e igb mlx_en
```

For ppc64 platform:

```
e1000 e1000e igb ibmveth ehea
```

For S390x:

```
qdio ccwgroup
```

If the network interface is not in the above list, you'd better specify the driver modules with this option.

### **-h|--help**

Display the usage message.

### **-v|--version**

Display the version.

**-V|--verbose**

Verbose output.

## RETRUN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

**node1** is one diskful Linux node, which is managed by xCAT.

1. There's one pre-defined *osimage*. In order to capture and prepare the diskless root image for *osimage*, run the command:

```
imgcapture node1 -t diskless -o osimage
```

2. In order to capture the diskful image from **node1** and create the *osimage* **img1**, run the command:

```
imgcapture node1 -t sysclone -o img1
```

## FILES

/opt/xcat/bin/imgcapture

## SEE ALSO

genimage(1)|genimage.1, imgimport(1)|imgimport.1, imgexport(1)|imgexport.1, packimage(1)|packimage.1, liteimg(1)|liteimg.1, nodeset(8)|nodeset.8

## imgexport.1

### NAME

**imgexport** - Exports an xCAT image.

## SYNOPSIS

**imgexport** [-h] --help]

**imgexport** *image\_name* [*destination*] [-e | --extra *file:dir*] ... ] [-p | --postscripts *node\_name*] [-R | --remotehost *user@host*] [-v | --verbose]



## DESCRIPTION

The **imgexport** command will export an image that is being used by xCAT. To export images, you must have the images defined in the *osimage* table. All the columns in the *osimage* and *linuximage* tables will be exported. If kits are used in stateful or stateless images, *kit*, *kitcomponent* and *kitrepo* tables will be exported. In addition, the following files will also be exported.

### For stateful:

```
x.pkglist
x.otherpkgs.pkglist
x.tmpl
x.synclist
kits related files
```

### For stateless:

```
kernel
initrd.gz
rootimg.cpio.xz or rootimg.cpio.gz or rootimg.tar.xz or rootimg.tar.gz or rootimg.gz(for_
↳backward-compatibility)
x.pkglist
x.otherpkgs.pkglist
x.synclist
x.postinstall
x.exlist
kits related files
```

### For statelite:

```
kernel
initrd.gz
root image tree
x.pkglist
x.synclist
x.otherpkgs.pkglist
x.postinstall
x.exlist
```

where x is the name of the profile.

Any files specified by the **-e** flag will also be exported. If **-p** flag is specified, the names of the postscripts and the postbootscripts for the given node will be exported. The postscripts themselves need to be manually exported using **-e** flag.

For statelite, the litefile table settings for the image will also be exported. The litetree and statelite tables are not exported.

## OPTIONS

**-e|--extra** *srcfile:destdir*

Pack up extra files. If *destdir* is omitted, the destination directory will be the same as the source directory.

**-h|--help**

Display usage message.

**-p|--postscripts** *node\_name*

Get the names of the postscripts and postbootscripts for the given node and pack them into the image.

**-R|--remotehost** *user@host*

Export the image to remote host. Passwordless ssh must be setup to the remote host.

**-v|--verbose**

Verbose output.

*image\_name*

The name of the image. Use **lsdef -t osimage** to find out all the image names.

*destination*

The output bundle file name. If remote host is specified with **--remotehost** option, the *destination* can include the path to the bundle file. If remote host is not specified, the bundle file is placed in a local working directory, even if *destination* includes a path.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. Simplest way to export an image. If there is an image in the osimage table named 'foo', then run:

```
imgexport foo
```

foo.tgz will be built in the current working directory. Make sure that you have enough space in the directory that you are in to run imgexport, if you have a big image to tar up.

2. To include extra files with your image:

```
imgexport Default_Stateless_1265981465 foo.tgz -e /install/postscripts/myscript1 -e /tmp/
↪mydir:/usr/mydir
```

In addition to all the default files, this will export */install/postscripts/myscript1* and the whole directory */tmp/dir* into the file called foo.tgz. And when imgimport is called */install/postscripts/myscript1* will be copied into the same directory and */tmp/mydir* will be copied to */usr/mydir*.

3. To include postscript with your image:

```
imgexport Default_Stateless_1265981465 foo.tgz -p node1 -e /install/postscripts/myscript1
```

The *postscripts* and the *postbootscripts* names specified in the *postscripts* table for node1 will be exported into the image. The postscript *myscript1* will also be exported.

## FILES

/opt/xcat/bin/imgexport

## SEE ALSO

imgimport(1)|imgimport.1

## imgimport.1

## NAME

**imgimport** - Imports an xCAT image or configuration file into the xCAT tables so that you can immediately begin deploying with it.

## SYNOPSIS

**imgimport** [-h|--help]

**imgimport** *bundle\_file\_name* [-p | --postscripts *nodelist*] [-f | --profile *new\_profile*] [-R | --remotehost *user@host*] [-v | --verbose]

## DESCRIPTION

The **imgimport** command will import an image that has been exported by **imgexport** from xCAT. This is the easiest way to transfer, backup, change or share images created by xCAT whether they be stateless or stateful. The bundle file will be unpacked in the current working directory. The xCAT configuration such as *osimage* and *linuximage* tables will then be updated.

**For stateful, the following files will be copied to the appropriate directories**

```
x.pkglist
x.otherpkgs.pkglist
x.tmpl
x.synclist
kits related files
```

**For stateless, the following files will be copied to the appropriate directories**

```
kernel
initrd.gz
rootimg.cpio.xz or rootimg.cpio.gz or rootimg.tar.xz or rootimg.tar.gz or rootimg.gz(for_
↳backward-compatibility)
x.pkglist
x.otherpkgs.pkglist
x.synclist
```

(continues on next page)

(continued from previous page)

```
x.postinstall
x.exlist
kits related files
```

For statelite, the following files will be copied to the appropriate directories

```
kernel
initrd.gz
root image tree
x.pkglist
x.synclist
x.otherpkgs.pkglist
x.postinstall
x.exlist
```

where x is the profile name.

Any extra files, included by **--extra** flag in the **imgexport** command, will also be copied to the appropriate directories.

For statelite, the litefile table will be updated for the image. The litetree and statelite tables are not imported.

If **-p** flag is specified, the *postscripts* table will be updated with the postscripts and the postbootscripts names from the image for the nodes given by this flag.

If **-f** flag is not specified, all the files will be copied to the same directories as the source. If it is specified, the old profile name x will be changed to the new and the files will be copied to the appropriate directories for the new profiles. For example, */opt/xcat/share/xcat/netboot/sles/x.pkglist* will be copied to */install/custom/netboot/sles/compute\_new.pkglist* and */install/netboot/sles11/ppc64/x/kernel* will be copied to */install/netboot/sles11/ppc64/compute\_new/kernel*. This flag is commonly used when you want to copy the image on the same xCAT mn so you can make modification on the new one.

After this command, you can run the **nodeset** command and then start deploying the nodes. You can also choose to modify the files and run the following commands before the node deployment.

**For stateful:**

```
nodeset
```

**For stateless:**

```
genimage
packimage
nodeset
```

**For statelite:**

```
genimage
liteimg
nodeset
```

## OPTIONS

**-f|--profile** *new\_profile*

Import the image with a new profile name.

**-h|--help**

Display usage message.

**-p|--postscripts** *nodelist*

Import the postscripts. The postscripts contained in the image will be set in the postscripts table for *nodelist*.

**-R|--remotehost** *user@host*

Import the image from remote host. Passwordless ssh must be setup to the remote host.

**-v|--verbose**

Verbose output.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. Simplest way to import an image. If there is a bundle file named 'foo.gz', then run:

```
imgimport foo.gz
```

2. Import the image with postscript names.

```
imgimport foo.gz -p node1,node2
```

The *postscripts* table will be updated with the name of the *postscripts* and the *postbootscripts* for node1 and node2.

3. Import the image with a new profile name

```
imgimport foo.gz -f compute_test
```

## FILES

/opt/xcat/bin/imgimport

## SEE ALSO

imgexport(1)|imgexport.1

## liteimg.1

## NAME

**liteimg** - Modify statelite image by creating a series of links.

## SYNOPSIS

**liteimg** [-h| --help]

**liteimg** [-v| --version]

**liteimg** *imagename*

## DESCRIPTION

This command modifies the statelite image by creating a series of links. It creates 2 levels of indirection so that files can be modified while in their image state as well as during runtime. For example, a file like <\$imgroot>/etc/ntp.conf will have the following operations done to it:

```
* mkdir -p $imgroot/.default/etc*
```

```
* mkdir -p $imgroot/.statelite/tmpfs/etc*
```

```
* mv $imgroot/etc/ntp.conf $imgroot/.default/etc*
```

```
* cd $imgroot/.statelite/tmpfs/etc*
```

```
* ln -sf ../../../../default/etc/ntp.conf .*
```

```
* cd $imgroot/etc*
```

```
* ln -sf ../.statelite/tmpfs/etc/ntp.conf .*
```

When finished, the original file will reside in *\$imgroot/.default/etc/ntp.conf*. *\$imgroot/etc/ntp.conf* will link to *\$imgroot/.statelite/tmpfs/etc/ntp.conf* which will in turn link to *\$imgroot/.default/etc/ntp.conf*

Note: If you make any changes to your litefile table after running liteimg then you will need to rerun liteimg again.

## PARAMETERS

*imagename* specifies the name of a os image definition to be used. The specification for the image is stored in the *osimage* table and *linuximage* table.

## OPTIONS

**-h|--help** Display usage message.

**-v|--version** Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To lite a RHEL 6.6 statelite image for a compute node architecture x86\_64 enter:

```
liteimg rhels6.6-x86_64-statelite-compute
```

## FILES

/opt/xcat/bin/liteimg

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

genimage(1)|genimage.1

## lsdef.1

## NAME

**lsdef** - Use this command to list xCAT data object definitions.

## SYNOPSIS

**lsdef** [-h | --help] [-t *object-types*] [-i *attr-list*]

**lsdef** [-V | --verbose] [-a | --all] [-S] [-t *object-types*] [-o *object-names*] [-z | --stanza] [-i *attr-list* | [-l | --long] | [-s | --short]] [-c | --compress] [--osimage] [--nics] [[-w *attr==val*] [-w *attr=~val*] ...] [*noderange*]

**lsdef** [-a | --all] [-t *object-types*] [-z | --stanza] [-i *attr-list* | [-l | --long] | [-s | --short]] [--template [*template-object-name*]]

## DESCRIPTION

This command is used to display xCAT object definitions which are stored in the xCAT database and xCAT object definition templates shipped in xCAT.

## OPTIONS

### **-a|--all**

Display all definitions. For performance consideration, the auditlog and eventlog objects will not be listed. To list auditlog or eventlog objects, use **lsdef -t auditlog** or **lsdef -t eventlog** instead.

### **-c|--compress**

Display information in compressed mode, each output line has format “<object name>: <data>”. The output can be passed to command **xcoll** or **xdshbak** for formatted output. The **-c** flag must be used with **-i** flag.

### **-h|--help**

Display usage message.

### **-i attr-list**

Comma separated list of attribute names to display.

### **-l|--long**

List the complete object definition.

### **-s|--short**

Only list the object names.

### **-S**

List all the hidden nodes (FSP/BPA nodes) with other ones.

### *noderange*

A set of comma delimited node names and/or group names. See the “noderange” man page for details on supported formats.

### **-o object-names**

A set of comma delimited object names.

### **--template [template-object-name]**

Show the object definition templates *template-object-name* shipped in xCAT. If no *template-object-name* is specified, all the object definition templates of the specified type **-t object-types** will be listed. Use **-a|--all** option to list all the object definition templates.

### **--osimage**

Show all the osimage information for the node.

### **--nics**

Show the nics configuration information for the node.

### **-t object-types**

A set of comma delimited object types. Use the help option to get a list of valid objects.

### **-V|--verbose**



Verbose mode.

**-w attr==val -w attr=~val ...**

Use one or multiple **-w** flags to specify the selection string that can be used to select objects. The operators **==**, **!=**, **=~** and **!~** are available. Use the help option to get a list of valid attributes for each object type.

Operator descriptions:

```
==      Select nodes where the attribute value is exactly this value.
!=      Select nodes where the attribute value is not this specific value.
=~      Select nodes where the attribute value matches this regular_
↪expression.
!~      Select nodes where the attribute value does not match this regular_
↪expression.
```

Note: if the “val” fields includes spaces or any other characters that will be parsed by shell, the “attr<operator>val” needs to be quoted. If the operator is “!~”, the “attr<operator>val” needs to be quoted using single quote.

**-z|--stanza**

Display output in stanza format. See the “xcatstanzafile” man page for details on using xCAT stanza files. And default is to list complete object definition, use **-i** to specify the attribute scope.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To display a description of all the valid attributes that could be used when defining an xCAT node.

```
lsdef -t node -h
```

2. To get a list of all the objects that have been defined.

```
lsdef -a
```

3. To get all the attributes of the node1

```
lsdef node1
OR
lsdef -t node node1
OR
lsdef -t node -o node1
```

4. To get the object name of node1 instead of all the attributes

```
lsdef -s node1
```

5. To get a list of all the network definitions.

```
lsdef -t network
```

6. To get a complete listing of all network definitions.

```
lsdef -l -t network
```

7. To list the whole xCAT database and write it to a stanza file. (backup database)

```
lsdef -a -l -z > mydbstanzafile
```

8. To list the MAC and install adapter name for each node.

```
lsdef -t node -i mac,installnic
```

9. To list an osimage definition named “aix53J”.

```
lsdef -t osimage -l -o aix53J
```

10. To list all node definitions that have a status value of “booting”.

```
lsdef -t node -w status==booting
```

11. To list all the attributes of the group “service”.

```
lsdef -l -t group -o service
```

12. To list all the attributes of the nodes that are members of the group “service”.

```
lsdef -t node -l service
```

13. To get a listing of object definitions that includes information about what xCAT database tables are used to store the data.

```
lsdef -V -l -t node -o node01
```

14. To list the hidden nodes that can’t be seen with other flags. The hidden nodes are FSP/BPAs.

```
lsdef -S
```

15. To list the nodes status and use **xcoll** to format the output.

```
lsdef -t node -i status -c | xcoll
```

16. To display the description for some specific attributes that could be used when defining an xCAT node.

```
lsdef -t node -h -i profile,pprofile
```

17. To display the nics configuration information for node cn1.

```
lsdef cn1 --nics
```

18. To list all the object definition templates shipped in xCAT.

```
lsdef --template -a
```

19. To display the details of “node” object definition template “ppc64le-template” shipped in xCAT.

```
lsdef -t node --template ppc64le-template
```

20. To list all the “node” object definition templates shipped in xCAT.

```
lsdef -t node --template
```

## FILES

/opt/xcat/bin/lsdef

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

mkdef(1)|mkdef.1, chdef(1)|chdef.1, rmdef(1)|rmdef.1, xcatstanzafile(5)|xcatstanzafile.5

## lsflexnode.1

## NAME

**lsflexnode** - Display the information of flexible node

## SYNOPSIS

**lsflexnode** [-h | --help]

**lsflexnode** [-v | --version]

**lsflexnode** *noderange*

## DESCRIPTION

IBM BladeCenter HX5 offers flexibility ideal that the blades can be combined together for scalability.

There are several concepts to support the HX5 multiple blades combination:

**Complex:** Multiple blades which combined by a scalability card is a complex.

**Partition:** A logic concept which containing part of the **Blade slot node** in a complex. Each partition can map to a system to install Operating System. Each partition could have 1HX5, 1HX5+1MD or 2HX5+2MD. (MD is the Memory Drawer)

**Blade slot node:** The physical blade which installed in the slot of a chassis. It can be a HX5 or MD.

A **Complex** will be created automatically when a multiple blades combination is installed. In this **Complex**, every blade belongs to it is working as a **Blade slot node**.

A **Partition** can be created base on the **Complex**, each **Partition** can have one or multiple **Blade slot node**.

The *noderange* in the **SYNOPSIS** can be a AMM node or a blade node.

## OPTIONS

### **-h | --help**

Display the usage message.

### **-v | --version**

Display the version information.

## ATTRIBUTES

The meaning of attributes which displayed by the **lsflexnode**. The word ‘node’ in this section means **Blade slot node**.

### **Complex**

The unique numeric identifier for a complex installed in the chassis.

### **Partition number**

The number of partitions currently defined for this complex.

### **Complex node number**

The number of nodes existing in this complex, regardless of their assignment to any given partition.

### **Partition**

The unique numeric identifier for a partition defined within a complex installed in the chassis.

### **Partition Mode**

The currently configured mode of this partition. It can be ‘partition’ or ‘standalone’.

### **Partition node number**

The number of nodes currently defined for this partition.

### **Partition status**

The current power status of this partition when the partition has a valid partition configuration. It can be ‘poweredoff’, ‘poweredon’, ‘resetting’ or ‘invalid’.

### **Node**

The unique numeric identifier for this node, unique within the partition. If this node does not belong to a partition, the slot number will be displayed.

### **Node state**

The physical power state of this node. It can be ‘poweredoff’, ‘poweredon’ or ‘resetting’.

### **Node slot**

The base slot number where the node exists in the chassis.

### **Node resource**

A string providing a summary overview of the resources provided by this node. It includes the CPU number, CPU frequency and Memory size.

### **Node type**

The general categorization of the node. It can be ‘processor’, ‘memory’ or ‘io’.

### **Node role**

Indicates if the node is assigned to a partition, and if so, provides an indication of whether the node is the primary node of the partition or not.

### Flexnode state

The state of a flexible node. It is the state of the partition which this node belongs to. If this node does NOT belong to a partition, the value should be 'invalid'.

It can be 'poweredoff', 'poweredon', 'resetting' or 'invalid'.

### Complex id

The identifier of the complex this node belongs to.

### Partition id

The identifier of the partition this node belongs to.

## EXAMPLES

1 Display all the **Complex**, **Partition** and **Blade slot node** which managed by a AMM.

```
lsflexnode amm1
```

The output:

```
amm1: Complex - 24068
amm1: ..Partition number - 1
amm1: ..Complex node number - 2
amm1: ..Partition = 1
amm1: ....Partition Mode - partition
amm1: ....Partition node number - 1
amm1: ....Partition status - poweredoff
amm1: ....Node - 0 (logic id)
amm1: .....Node state - poweredoff
amm1: .....Node slot - 14
amm1: .....Node type - processor
amm1: .....Node resource - 2 (1866 MHz) / 8 (2 GB)
amm1: .....Node role - secondary
amm1: ..Partition = unassigned
amm1: ....Node - 13 (logic id)
amm1: .....Node state - poweredoff
amm1: .....Node slot - 13
amm1: .....Node type - processor
amm1: .....Node resource - 2 (1866 MHz) / 8 (2 GB)
amm1: .....Node role - unassigned
```

2 Display a flexible node.

```
lsflexnode blade1
```

The output:

```
blade1: Flexnode state - poweredoff
blade1: Complex id - 24068
blade1: Partition id - 1
```

(continues on next page)

(continued from previous page)

```
blade1: Slot14: Node state - poweredoff
blade1: Slot14: Node slot - 14
blade1: Slot14: Node type - processor
blade1: Slot14: Node resource - 2 (1866 MHz) / 8 (2 GB)
blade1: Slot14: Node role - secondary
```

## FILES

/opt/xcat/bin/lsmflexnode

## SEE ALSO

mkflexnode(1)|mkflexnode.1, rmflexnode(1)|rmflexnode.1

## lshwconn.1

## NAME

**lshwconn** - Use this command to display the connection status for CEC and Frame nodes.

## SYNOPSIS

**lshwconn** [-h] --help]

**lshwconn** [-v] --version]

### PPC (with HMC) specific:

**lshwconn** [-V] --verbose] *noderange*

### PPC (without HMC, using FSPAPI) specific:

**lshwconn** *noderange* -T *tooltype*

## DESCRIPTION

This command is used to display the connection status for CEC and Frame node.

## OPTIONS

### **-h|--help**

Display usage message.

### **-V|--verbose**

Verbose output.

### **-T**

The tooltype is used to communicate to the CEC/Frame. The value could be lpar or fnm. The tooltype value lpar is for xCAT and fnm is for CNM.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To display connection status for all CEC nodes in node group CEC:

```
lshwconn cec
```

Output is similar to:

```
cec1: ipaddr=192.168.200.245,alt_ipaddr=unavailable,state=Connected
cec2: Connection not found
```

2. To display connection status for Frame node frame1:

```
lshwconn frame1
```

Output is similar to:

```
frame1: side=a,ipaddr=192.168.200.247,alt_ipaddr=unavailable,state=Connected
frame1: side=b,ipaddr=192.168.200.248,alt_ipaddr=unavailable,state=Connected
```

3. To display connection status for all CEC nodes in node group CEC to hardware server, and using lpar tooltype:

```
lshwconn cec -T lpar
```

Output is similar to:

```
cec1: sp=primary,ipadd=40.3.7.1,alt_ipadd=unavailable,state=LINE UP
cec2: Connection not found
```

## FILES

`$XCATROOT/bin/lshwconn`

(The `XCATROOT` environment variable is set when xCAT is installed. The default value is “`/opt/xcat`”.)

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

`rmhwconn(1)`|`rmhwconn.1`, `mkhwconn(1)`|`mkhwconn.1`

## lskit.1

## NAME

**lskit** - Lists information for one or more Kits.

## SYNOPSIS

**lskit** [-V | --verbose] [-F | --framework *kitattr\_names*] [-x | --xml | --XML] [-K | --kitattr *kitattr\_names*] [-R | --repoattr *repoattr\_names*] [-C | --compattr *compattr\_names*] [*kit\_names*]

**lskit** [-? | -h | --help | -v | --version]

**lskit** [-F | --framework *kit\_path\_name*]

## DESCRIPTION

The **lskit** command is used to list information for one or more kits. A kit is a special kind of package that is used to install a software product on one or more nodes in an xCAT cluster.

Note: The xCAT support for Kits is only available for Linux operating systems.

The **lskit** command outputs the following info for each kit: the kit’s basic info, the kit’s repositories, and the kit’s components. The command outputs the info in two formats: human-readable format (default), and XML format. Use the `-x` option to view the info in XML format.

Input to the command can specify any number or combination of the input options.



## OPTIONS

### **-F|--framework** *kit\_path\_name*

Use this option to display the framework values of the specified Kit tarfile. This information is retrieved directly from the tarfile and can be done before the Kit has been defined in the xCAT database. This option cannot be combined with other options.

### **-K|--kitattr** *kitattr\_names*

Where *kitattr\_names* is a comma-delimited list of kit attribute names. The names correspond to attribute names in the **kit** table. The **lskit** command will only display the specified kit attributes.

### **-R|--repoattr** *repoattr\_names*

Where *repoattr\_names* is a comma-delimited list of kit repository attribute names. The names correspond to attribute names in the **kitrepo** table. The **lskit** command will only display the specified kit repository attributes.

### **-C|--compattr** *compattr\_names*

where *compattr\_names* is a comma-delimited list of kit component attribute names. The names correspond to attribute names in the **kitcomponent** table. The **lskit** command will only display the specified kit component attributes.

### *kit\_names*

is a comma-delimited list of kit names. The **lskit** command will only display the kits matching these names.

### **-x|--xml|--XML**

Need XCATXMLTRACE=1 env when using -x|--xml|--XML, for example: XCATXMLTRACE=1 lskit -x testkit-1.0.0 Return the output with XML tags. The data is returned as:

```
<data>
  <kitinfo>
    ...
  </kitinfo>
</data>
...
<data>
  <kitinfo>
    ...
  </kitinfo>
</data>
```

Each <kitinfo> tag contains info for one kit. The info inside <kitinfo> is structured as follows:

```
The <kit> sub-tag contains the kit's basic info.
The <kitrepo> sub-tags store info about the kit's repositories.
The <kitcomponent> sub-tags store info about the kit's components.
```

The data inside <kitinfo> is returned as:

```
<kitinfo>
  <kit>
    ...
  </kit>
```

(continues on next page)

(continued from previous page)

```
<kitrepo>
...
</kitrepo>
...

<kitcomponent>
...
</kitcomponent>
...
</kitinfo>
```

### **-V|--verbose**

Display additional progress and error messages.

### **-v|--version**

Command Version.

### **-?|-h|--help**

Display usage message.

## **RETURN VALUE**

0 The command completed successfully.

1 An error has occurred.

## **EXAMPLES**

1. To list all kits, enter:

```
lskit
```

2. To list the kit “kit-test1-1.0-Linux”, enter:

```
lskit kit-test1-1.0-Linux
```

3. To list the kit “kit-test1-1.0-Linux” for selected attributes, enter:

```
lskit -K basename,description -R kitreponame -C kitcompname kit-test1-1.0-Linux
```

4. To list the framework value of a Kit tarfile.

```
lskit -F /myhome/mykits/pperte-1.3.0.2-0-x86_64.tar.bz2
```

Output is similar to:

```
Extracting the kit.conf file from /myhome/mykits/pperte-1.3.0.2-0-x86_64.tar.
↪bz2. Please wait.
```

```
kitframework=2
compatible_kitframeworks=0,1,2
```

5. To list kit “testkit-1.0-1” with XML tags, enter:

```
XCATXMLTRACE=1 lskit -x testkit-1.0-1
```

## FILES

/opt/xcat/bin/lskit

## SEE ALSO

lskitcomp(1)|lskitcomp.1, lskitdeployparam(1)|lskitdeployparam.1, addkit(1)|addkit.1, rmkit(1)|rmkit.1, addkitcomp(1)|addkitcomp.1, rmkitcomp(1)|rmkitcomp.1

## lskitcomp.1

## NAME

**lskitcomp** - Used to list information for one or more kit components.

## SYNOPSIS

**lskitcomp** [-V | --verbose] [-x | --xml | --XML] [-C | --compattr *compattr\_names*] [-O | --osdistro *os\_distro*] [-S | --serverrole *server\_role*] [*kitcomp\_names*]

**lskitcomp** [-? | -h | --help | -v | --version]

## DESCRIPTION

The **lskitcomp** command is used to list information for one or more kit components. A kit is made up of one or more kit components. Each kit component is a meta package used to install a software product component on one or more nodes in an xCAT cluster.

The **lskitcomp** command outputs the kit component info in two formats: human-readable format (default), and XML format. Use the -x option to view the info in XML format.

Input to the command can specify any number or combination of the input options.

Note: The xCAT support for Kits is only available for Linux operating systems.

## OPTIONS

**-C|--compattr** *compattr\_names*

where *compattr\_names* is a comma-delimited list of kit component attribute names. The names correspond to attribute names in the **kitcomponent** table. The **lskitcomp** command will only display the specified kit component attributes.

**-O|--osdistro** *os\_distro*

where *os\_distro* is the name of an osdistro in **osdistro** table. The **lskitcomp** command will only display the kit components matching the specified osdistro.

**-S|--serverrole** *server\_role*

where *server\_role* is the name of a server role. The typical server roles are: mgtnode, servicenode, computenode, loginnode, storagenode. The **lskitcomp** command will only display the kit components matching the specified server role.

*kitcomp\_names*

is a comma-delimited list of kit component names. The **lskitcomp** command will only display the kit components matching the specified names.

**-x|--xml|--XML**

Need XCATXMLTRACE=1 env when using -x|--xml|--XML. Return the output with XML tags. The data is returned as:

```
<data>
  <kitinfo>
    ...
  </kitinfo>
</data>
...
<data>
  <kitinfo>
    ...
  </kitinfo>
</data>
```

Each <kitinfo> tag contains info for a group of kit components belonging to the same kit. The info inside <kitinfo> is structured as follows:

The <kit> sub-tag contains the kit's name.  
The <kitcomponent> sub-tags store info about the kit's components.

The data inside <kitinfo> is returned as:

```
<kitinfo>
  <kit>
    ...
  </kit>

  <kitcomponent>
    ...
  </kitcomponent>
  ...
</kitinfo>
```

**-V|--verbose**

Display additional progress and error messages.

**-v|--version**

Command Version.

**-?|-h|--help**

Display usage message.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To list all kit components, enter:

```
lskitcomp
```

2. To list the kit component “comp-server-1.0-1-rhels-6-x86\_64”, enter:

```
lskitcomp comp-server-1.0-1-rhels-6-x86_64
```

3. To list the kit component “comp-server-1.0-1-rhels-6-x86\_64” for selected kit component attributes, enter:

```
lskitcomp -C kitcompname,desc comp-server-1.0-1-rhels-6-x86_64
```

4. To list kit components compatible with “rhels-6.2-x86\_64” osdistro, enter:

```
lskitcomp -O rhels-6.2-x86_64
```

5. To list kit components compatible with “rhels-6.2-x86\_64” osdistro and “computenode” server role, enter:

```
lskitcomp -O rhels-6.2-x86_64 -S computenode
```

6. To list the kit component “testkit-compute-1.0-1-ubuntu-14.04-ppc64el” with XML tags, enter:

```
XCATXMLTRACE=1 lskitcomp -x testkit-compute-1.0-1-ubuntu-14.04-ppc64el
```

## FILES

/opt/xcat/bin/lskitcomp

## SEE ALSO

lskit(1)|lskit.1, lskitdeployparam(1)|lskitdeployparam.1, addkit(1)|addkit.1, rmkit(1)|rmkit.1, addkit-comp(1)|addkitcomp.1, rmkitcomp(1)|rmkitcomp.1

### lskitdeployparam.1

## NAME

**lskitdeployparam** - Lists the deployment parameters for one or more Kits or Kit components

## SYNOPSIS

**lskitdeployparam** [-V | --verbose] [-x | --xml | --XML] [-k | --kitname *kit\_names*] [-c | --compname *comp\_names*]

**lskitdeployparam** [-? | -h | --help | -v | --version]

## DESCRIPTION

The **lskitdeployparam** command is used to list the kit deployment parameters for one or more kits, or one or more kit components. Kit deployment parameters are used to customize the installation or upgrade of kit components.

The **lskitdeployparam** command outputs the kit component information in two formats: human-readable format (default), and XML format. Use the -x option to view the information in XML format.

Input to the command can specify any combination of the input options.

Note: The xCAT support for Kits is only available for Linux operating systems.

## OPTIONS

**-k|--kitname** *kit\_names*

Where *kit\_names* is a comma-delimited list of kit names. The **lskitdeployparam** command will only display the deployment parameters for the kits with the matching names.

**-c|--compname** *comp\_names*

Where *comp\_names* is a comma-delimited list of kit component names. The **lskitdeployparam** command will only display the deployment parameters for the kit components with the matching names.

**-x|--xml|--XML**

Return the output with XML tags. The data is returned as:

```
<data>
  <kitdeployparam>
    <name>KIT_KIT1_PARAM1</name>
    <value>value11</value>
  </kitdeployparam>
</data>
<data>
  <kitdeployparam>
    <name>KIT_KIT1_PARAM2</name>
    <value>value12</value>
  </kitdeployparam>
</data>
...
```

**-V|--verbose**

Display additional progress and error messages.

**-v|--version**

Command Version.

**-?|-h|--help**

Display usage message.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To list kit deployment parameters for kit “kit-test1-1.0-Linux”, enter:

```
lskitdeployparam -k kit-test1-1.0-Linux
```

2. To list kit deployment parameters for kit component “comp-server-1.0-1-rhels-6-x86\_64”, enter:

```
lskitdeployparam -c comp-server-1.0-1-rhels-6-x86_64
```

## FILES

/opt/xcat/bin/lskitdeployparam

## SEE ALSO

lskit(1)|lskit.1, lskitcomp(1)|lskitcomp.1, addkit(1)|addkit.1, rmkit(1)|rmkit.1, addkitcomp(1)|addkitcomp.1, rmkitcomp(1)|rmkitcomp.1

## lskmodules.1

## NAME

**lskmodules** - list kernel driver modules in rpms or driver disk image files

## SYNOPSIS

**lskmodules** [-V | --verbose] [-i | --osimage *osimage\_names*] [-c | --kitcomponent *kitcomp\_names*] [-o | --osdistro *osdistro\_names*] [-u | --osdistropupdate *osdistroupdate\_names*] [-x | --xml | --XML]

**lskmodules** [-? | -h | --help | -v | --version]

## DESCRIPTION

The **lskmodules** command finds the kernel driver module files (\*.ko) in the specified input locations, runs the modinfo command against each file, and returns the driver name and description. If -x is specified, the output is returned with XML tags.

Input to the command can specify any number or combination of the input options.

## OPTIONS

**-i|--osimage** *osimage\_names*

where *osimage\_names* is a comma-delimited list of xCAT database osimage object names. For each *osimage\_name*, lskmodules will use the entries in *osimage.driverupdatesrc* for the rpms and driver disk image files to search.

**-c|--kitcomponent** *kitcomponent\_names*

where *kitcomponent\_names* is a comma-delimited list of xCAT database kitcomponent object names. For each *kitcomponent\_name*, lskmodules will use the entries in *kitcomponent.driverpacks* for the rpm list and the *repodir* of the *kitcomponent.kitreponame* for the location of the rpm files to search.

**-o|--osdistro** *osdistro\_names*

where *osdistro\_names* is a comma-delimited list of xCAT database osdistro object names. For each *osdistro\_name*, lskmodules will search each *<osdistro.dirpaths>/Packages/kernel-<kernelversion>.rpm* file.

**-u|--osdistroudate** *osdistroudate\_names*

where *osdistroudate\_names* is a comma-delimited list of xCAT database osdistroudate table entries. For each *osdistroudate\_name*, lskmodules will search the *<osdistroudate.dirpath>/kernel-<kernelversion>.rpm* file.

**-x|--xml|--XML**

Return the output with XML tags. The data is returned as:

```
<module>
  <name> xxx.ko </name>
  <description> this is module xxx </description>
</module>
```

This option is intended for use by other programs. The XML will not be displayed. To view the returned XML, set the *XCATSHOWXML=yes* environment variable before running this command.

**-V|--verbose**

Display additional progress and error messages.

**-v|--version**

Command Version.

**-?|-h|--help**

Display usage message.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.



## EXAMPLES

1. To list the kernel modules included in the driverpacks shipped with kitcomponent kit1\_comp1-x86\_64, enter:

```
lskmodules -c kit1_comp1-x86_64
```

## FILES

## SEE ALSO

**lslite.1**

## NAME

**lslite** - Display a summary of the statelite information.

## SYNOPSIS

**lslite** [-h | --help]

**lslite** [-V | --verbose] [-i *imagename*] | [*noderange*]

## DESCRIPTION

The **lslite** command displays a summary of the statelite information that has been defined for a noderange or an image.

## OPTIONS

**-h|--help**

Display usage message.

**-V|--verbose**

Verbose mode.

**-i** *imagename*

The name of an existing xCAT osimage definition.

*noderange*

A set of comma delimited node names and/or group names. See the “noderange” man page for details on additional supported formats.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To list the statelite information for an xCAT node named “node01”.

```
lslite node01
```

Output is similar to:

```
>>>Node: node01

Oimage: 61img

Persistent directory (statelite table):
    xcatmn1:/statelite

Litefiles (litefile table):
    tmpfs,rw      /etc/adjtime
    tmpfs,rw      /etc/lvm/.cache
    tmpfs,rw      /etc/mtab
    . . . . .

Litetree path (litetree table):
    1,MN:/etc
    2,server1:/etc
```

2. To list the statelite information for an xCAT oimage named “osimage01”.

```
lslite -i osimage01
```

Output is similar to:

```
tmpfs,rw      /etc/adjtime
tmpfs,rw      /etc/lvm/.cache
tmpfs,rw      /etc/mtab
. . . . .
```

## FILES

/opt/xcat/bin/lslite

## SEE ALSO

noderange(3)|noderange.3, tabdump(8)|tabdump.8

## lsslp.1

## NAME

**lsslp** - Discovers selected networked services information within the same subnet.

## SYNOPSIS

**lsslp** [-h] --help]

**lsslp** [-v] --version]

**lsslp** [noderange] [-V] [-i ip[,ip..]] [-w] [-r|-x|-z] [-n] [-s CEC|FRAME|MM|IVM|RSA|HMC|CMM|IMM2|FSP] [-t tries] [-I] [-C counts] [-T timeout] [--vpdtable]

## DESCRIPTION

The lsslp command discovers selected service types using the -s flag. All service types are returned if the -s flag is not specified. If a specific IP address is not specified using the -i flag, the request is sent out all available network adapters. The optional -r, -x, -z and --vpdtable flags format the output. If you can't receive all the hardware, use -T to increase the waiting time.

NOTE: SLP broadcast requests will propagate only within the subnet of the network adapter broadcast IPs specified by the -i flag.

## OPTIONS

**noderange** The nodes which the user wants to discover. If the user specifies the noderange, lsslp will just return the nodes in the node range. Which means it will help to add the new nodes to the xCAT database without modifying the existed definitions. But the nodes' name specified in noderange should be defined in database in advance. The specified nodes' type can be frame/cec/hmc/fsp/bpa. If the it is frame or cec, lsslp will list the bpa or fsp nodes within the nodes(bap for frame, fsp for cec). Do not use noderange with the flag -s.

**-i** IP(s) the command will send out (defaults to all available adapters).

**-h** Display usage message.

**-n** Only display and write the newly discovered hardware.

**-u** Do unicast to a specified IP range. Must be used with -s and --range. The -u flag is not supported on AIX.

**--range** Specify one or more IP ranges. Must be use in unicast mode. It accepts multiple formats. For example, 192.168.1.1/24, 40-41.1-2.3-4.1-100. If the range is huge, for example, 192.168.1.1/8, lsslp may take a very long time for node scan. So the range should be exactly specified.

**-r** Display Raw SLP response.

**-C** The number of the expected responses specified by the user. When using this flag, lsslp will not return until the it has found all the nodes or time out. The default max time is 3 seconds. The user can use -T flag the specify the time they want to use. A short time will limit the time costing, while a long time will help to find all the nodes.

- T** The number in seconds to limit the time of lsslp.
- s** Service type interested in discovering.
- t** Number or service-request attempts.
- vpdtable** Output the SLP response in vpdtable formatting. Easy for writing data to vpd table.
- v** Command Version.
- V** Verbose output.
- w** Writes output to xCAT database.
- x** XML format.
- z** Stanza formatted output.
- I** Give the warning message for the nodes in database which have no SLP responses. Note that this flag can only be used after the database migration finished successfully.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To list all discovered HMC service types in tabular format, enter:

```
lsslp -s HMC
```

Output is similar to:

device	type-model	serial-number	ip-addresses	hostname
HMC	7310CR2	103F55A	1.1.1.115	hmc01
HMC	7310CR2	105369A	3.3.3.103	hmc02
HMC	7310CR3	KPHHK24	3.3.3.154	hmc03

2. list all discovered FSP service types in raw response format on subnet 30.0.0.255, enter:

```
lsslp -i 3.0.0.255 -s CEC -r
```

Output is similar to:

```
(type=cec-service-processor),(serial-number=10A3AEB),(machinetype-model=9117-570),(fru-
↪ serial-number=YL11C5338102),(hostname=),(frame-number=0),(cage-number=0),(ip-address=3.
↪ 0.0.94,1.1.1.147),(web-url=https://3.0.0.94:473 ),(slot=1),(bpc-machinetype-model=0),
↪ (bpc-serial-number=0),(Image=fips240/b0630a_0623.240)
(type=cec-service-processor),(serial-number=10A3E2B),(machinetype-model=9117-570),(fru-
↪ serial- number=YL11C5338250),(hostname=),(frame-number=0),(cage-number=0),(ip-
↪ address=3.0.0.95,1.1.1.147),(web-url=https://3.0.0.95:473 ),(slot=1),(bpc-machinetype-
↪ model=0),(bpc-serial-number=0),(Image=fips240/b0630a_0623.240)
```

3. To list all discovered MM service types in XML format and write the output to the xCAT database, enter:

```
lsslp -s MM -x -w
```

Output is similar to:

```
<Node>
  <groups>mm,all</groups>
  <id>00:14:5E:E0:CB:1E</id>
  <mgt>blade</mgt>
  <mtm>029310C</mtm>
  <node>Server-029310C-SN100485A-A</node>
  <nodetype>mm</nodetype>
  <otherinterfaces>9.114.47.229</otherinterfaces>
  <serial>100485A</serial>
</Node>
```

4. To list all discovered service types in stanza format and write the output to the xCAT database, enter:

```
lsslp -z -w
```

Output is similar to:

```
c76v1hmc02:
  objtype=node
  hcp=c76v1hmc02
  nodetype=hmc
  mtm=7315CR2
  serial=10407DA
  ip=192.168.200.125
  groups=hmc,all
  mgt=hmc
  mac=00:1a:64:fb:7d:50
  hidden=0
192.168.200.244:
  objtype=node
  hcp=192.168.200.244
  nodetype=fsp
  mtm=9125-F2A
  serial=0262662
  side=A-0
  otherinterfaces=192.168.200.244
  groups=fsp,all
  mgt=fsp
  id=4
  parent=Server-9125-F2A-SN0262662
  mac=00:1a:64:fa:01:fe
  hidden=1
Server-8205-E6B-SN1074CDP:
  objtype=node
  hcp=Server-8205-E6B-SN1074CDP
  nodetype=cec
  mtm=8205-E6B
  serial=1074CDP
  groups=cec,all
```

(continues on next page)

(continued from previous page)

```

    mgt=fsp
    id=0
    hidden=0
192.168.200.33:
    objtype=node
    hcp=192.168.200.33
    nodetype=bpa
    mtm=9458-100
    serial=99201WM
    side=B-0
    otherinterfaces=192.168.200.33
    groups=bpa,all
    mgt=bpa
    id=0
    mac=00:09:6b:ad:19:90
    hidden=1
Server-9125-F2A-SN0262652:
    objtype=node
    hcp=Server-9125-F2A-SN0262652
    nodetype=frame
    mtm=9125-F2A
    serial=0262652
    groups=frame,all
    mgt=fsp
    id=5
    hidden=0

```

5. To list all discovered service types in stanza format and display the IP address, enter:

```
lsslp -w
```

Output is similar to:

```

mm01:
    objtype=node
    nodetype=fsp
    mtm=8233-E8B
    serial=1000ECP
    side=A-0
    groups=fsp,all
    mgt=fsp
    id=0
    mac=00:14:5E:F0:5C:FD
    otherinterfaces=50.0.0.5

bpa01:
    objtype=node
    nodetype=bpa
    mtm=9A01-100
    serial=0PIN746
    side=A-1
    groups=bpa,all

```

(continues on next page)

(continued from previous page)

```
mgt=bpa
id=0
mac=00:1A:64:54:8C:A5
otherinterfaces=50.0.0.1
```

6. To list all the CECs, enter:

```
lsslp -s CEC
```

device	type-model	serial-number	side	ip-addresses	hostname
FSP	9117-MMB	105EBEP	A-1	20.0.0.138	20.0.0.138
FSP	9117-MMB	105EBEP	B-1	20.0.0.139	20.0.0.139
CEC	9117-MMB	105EBEP			Server-9117-MMB-SN105EBEP

7. To list all the nodes defined in database which have no SLP response.

```
lsslp -I
```

Output is similar to:

These nodes defined in database but can't be discovered: f17c00bpcb\_b,f17c01bpcb\_a,  
↪ f17c01bpcb\_b,f17c02bpcb\_a,

device	type-model	serial-number	side	ip-addresses	hostname
bpa	9458-100	BPCF017	A-0	40.17.0.1	f17c00bpca_a
bpa	9458-100	BPCF017	B-0	40.17.0.2	f17c00bpcb_a

8. To find the nodes within the user specified. Make sure the noderange input have been defined in xCAT database.

```
lsslp CEC1-CEC3
```

```
or lsslp CEC1,CEC2,CEC3
```

device	type-model	serial-number	side	ip-addresses	hostname
FSP	9A01-100	0P1P336	A-0	192.168.200.34	192.168.200.34
FSP	9A01-100	0P1P336	B-0	192.168.200.35	192.168.200.35
FSP	9A01-100	0P1P336	A-1	50.0.0.27	50.0.0.27
FSP	9A01-100	0P1P336	B-1	50.0.0.28	50.0.0.28
CEC	9A01-100	0P1P336			CEC1
FSP	8233-E8B	1040C7P	A-0	192.168.200.36	192.168.200.36
FSP	8233-E8B	1040C7P	B-0	192.168.200.37	192.168.200.37
FSP	8233-E8B	1040C7P	A-1	50.0.0.29	50.0.0.29
FSP	8233-E8B	1040C7P	B-1	50.0.0.30	50.0.0.30
CEC	8233-E8B	1040C7P			CEC2
FSP	8205-E6B	1000ECP	A-0	192.168.200.38	192.168.200.38
FSP	8205-E6B	1000ECP	B-0	192.168.200.39	192.168.200.39
FSP	8205-E6B	1000ECP	A-1	50.0.0.31	50.0.0.27
FSP	8205-E6B	1000ECP	B-1	50.0.0.32	50.0.0.28
CEC	8205-E6B	1000ECP			CEC3

9. To list all discovered CMM in stanza format, enter: lsslp -s CMM -m -z

```
e114ngmm1:
  objtype=node
```

(continues on next page)

(continued from previous page)

```
mpa=e114ngmm1
nodetype=cmm
mtm=98939AX
serial=102537A
groups=cmm,all
mgt=blade
hidden=0
otherinterfaces=70.0.0.30
hwtype=cmm
```

10. To use lsslp unicast, enter:

```
lsslp -u -s CEC --range 40-41.1-2.1-2.1-2
```

## FILES

/opt/xcat/bin/lsslp

## SEE ALSO

rscan(1)|rscan.1

## lstree.1

## NAME

**lstree** - Display the tree of service node hierarchy, hardware hierarchy, or VM hierarchy.

## SYNOPSIS

**lstree** [-h | --help]

**lstree** [-s | --servicenode] [-H | --hardwaremgmt] [-v | --virtualmachine] [noderange]

## DESCRIPTION

The **lstree** command can display the tree of service node hierarchy for the xCAT nodes which have service node defined or which are service nodes, display the tree of hardware hierarchy only for the physical objects, display the tree of VM hierarchy for the xCAT nodes which are virtual machines or which are the hosts of virtual machines. If a noderange is specified, only show the part of the hierarchy that involves those nodes. For ZVM, we only support to display VM hierarchy. By default, lstree will show both the hardware hierarchy and the VM hierarchy for all the nodes.



## OPTIONS

### **-h|--help**

Display usage message.

### **-s|--servicenode**

Show the tree of service node hierarchy.

### **-H|--hardwaremgmt**

Show the tree of hardware hierarchy.

### **--v|--virtualmachine**

Show the tree of VM hierarchy.

### *noderange*

A set of comma delimited node names and/or group names. See the “noderange” man page for details on additional supported formats.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To display the tree of service node hierarchy for all the nodes.

```
lmtree -s
```

Output is similar to:

```
Service Node: mysn01
|__mycn01
|__mycn02
|__mycn03

Service Node: mysn02
|__mycn11
|__mycn12
|__mycn13
. . . . .
```

2. To display the tree of service node hierarchy for service node “mysn01”.

```
lmtree -s mysn01
```

Output is similar to:

```
Service Node: mysn01
|__mycn01
|__mycn02
|__mycn03
```

3. To display the tree of hardware hierarchy for all the nodes.

```
lmtree -H
```

Output is similar to:

```
HMC: myhmc01
|__Frame: myframe01
|__CEC: mycec01
|__CEC: mycec02
.....

Service Focal Point: myhmc02
|__Frame: myframe01
|__CEC: mycec01
|__CEC: mycec02
|__CEC: mycec03
.....

Management Module: mymm01
|__Blade 1: js22n01
|__Blade 2: js22n02
|__Blade 3: js22n03
.....

BMC: 192.168.0.1
|__Server: x3650n01
```

4. To display the tree of hardware hierarchy for HMC “myhmc01”.

```
lmtree -H myhmc01
```

Output is similar to:

```
HMC: myhmc01
|__Frame: myframe01
|__CEC: mycec01
|__CEC: mycec02
.....
```

5. To display the tree of VM hierarchy for all the nodes.

```
lmtree -v
```

Output is similar to:

```
Server: hs22n01
|__ hs22vm1
```

(continues on next page)

(continued from previous page)

```
Server: x3650n01
|__ x3650n01kvm1
|__ x3650n01kvm2
```

6. To display the tree of VM hierarchy for the node “x3650n01”.

```
lstree -v x3650n01
```

Output is similar to:

```
Server: x3650n01
|__ x3650n01kvm1
|__ x3650n01kvm2
```

7. To display both the hardware tree and VM tree for all nodes.

```
lstree
```

Output is similar to:

```
HMC: myhmc01
|__Frame: myframe01
|__CEC: mycec01
|__LPAR 1: node01
|__LPAR 2: node02
|__LPAR 3: node03
.....
|__CEC: mycec02
|__LPAR 1: node11
|__LPAR 2: node12
|__LPAR 3: node13
.....

Service Focal Point: myhmc02
|__Frame: myframe01
|__CEC: mycec01
|__LPAR 1: node01
|__LPAR 2: node02
|__LPAR 3: node03
.....
|__Frame: myframe02
|__CEC: mycec02
|__LPAR 1: node21
|__LPAR 2: node22
|__LPAR 3: node23
.....

Management Module: mymm01
|__Blade 1: hs22n01
|__hs22n01vm1
|__hs22n01vm2
|__Blade 2: hs22n02
|__hs22n02vm1
```

(continues on next page)

(continued from previous page)

```

|__hs22n02vm2
.....
BMC: 192.168.0.1
|__Server: x3650n01
|__ x3650n01kvm1
|__ x3650n01kvm2

```

## FILES

/opt/xcat/bin/lstree

## SEE ALSO

noderange(3)|noderange.3, tabdump(8)|tabdump.8

## lsve.1

## NAME

**lsve** - Lists detail attributes for a virtual environment.

## SYNOPSIS

**lsve** [-**t** *type*] [-**m** *manager*] [-**o** *object*]

## DESCRIPTION

The **lsve** command can be used to list a virtual environment for ‘Data Center’, ‘Cluster’, ‘Storage Domain’, ‘Network’ and ‘Template’ objects.

The mandatory parameter **-m** *manager* is used to specify the address of the manager of virtual environment. xCAT needs it to access the RHEV manager.

The mandatory parameter **-t** *type* is used to specify the type of the target object.

Basically, **lsve** command supports three types of object: **dc**, **cl**, **sd**, **nw** and **tpl**.

The parameter **-o** *object* is used to specify which object to list. If no **-o** is specified, all the objects with the **-t** type will be displayed.

## OPTIONS

**-h** Display usage message.

**-m** Specify the manager of the virtual environment.

For RHEV, the FQDN (Fully Qualified Domain Name) of the rhev manager have to be specified.

**-o** The target object to display.

**-t** Specify the **type** of the target object.

Supported types:

```
B<dc> - Data Center (For type of 'dc', all the elements belongs to the data_
      ↪center will be listed.)
B<cl> - Cluster
B<sd> - Storage Domain (To get the status of Storage Doamin, show it from I
      ↪<data center> it attached to.
B<nw> - Network
B<tpl> - Template
```

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To list the data center 'Default', enter:

```
lsve -t dc -m <FQDN of rhev manager> -o Default
```

Output is similar to:

```
datacenters: [Default]
description: The default Data Center
state: up
storageformat: v1
storagetype: nfs
  clusters: [Default]
    cpu: Intel Westmere Family
    description: The default server cluster
    memory_hugepage: true
    memory_overcommit: 100
  storagedomains: [image]
    available: 55834574848
    committed: 13958643712
    ismaster: true
    status: active
    storage_add: <Address of storage domain>
    storage_format: v1
    storage_path: /vfsimg
```

(continues on next page)

(continued from previous page)

```

storage_type: nfs
type: data
used: 9663676416
networks: [rhev2]
description:
state: operational
stp: false
networks: [rhev]
description: Management Network
state: operational
stp: false
templates: [Blank]
bootorder: hd
cpucore: 1
cpusocket: 1
creation_time: 2008-04-01T00:00:00.000-04:00
display: spice
memory: 536870912
state: ok
stateless: false
type: desktop

```

2. To list the cluster 'Default', enter:

```
lsve -t cl -m <FQDN of rhev manager> -o Default
```

Output is similar to:

```

cpu: Intel Westmere Family
description: The default server cluster
memory_hugepage: true
memory_overcommit: 10

```

3. To list the Storage Domain 'image', enter:

```
lsve -t sd -m <FQDN of rhev manager> -o image
```

Output is similar to:

```

storagedomains: [image]
available: 55834574848
committed: 13958643712
ismaster: true
status:
storage_add: <Address of storage domain>
storage_format: v1
storage_path: /vfsimg
storage_type: nfs
type: data
used: 9663676416

```

4. To list the network 'rhev', enter:

```
lsve -t nw -m <FQDN of rhev manager> -o rhevm
```

Output is similar to:

```
networks: [rhevm]
  description: Management Network
  state: operational
  stp: false
```

5. To list the template ‘tpl01’, enter:

```
lsve -t tpl -m <FQDN of rhev manager> -o tpl01
```

Output is similar to:

```
templates: [tpl01]
  bootorder: network
  cpucore: 2
  cpusocket: 2
  creation_time: 2012-08-22T23:52:35.953-04:00
  display: vnc
  memory: 1999634432
  state: ok
  stateless: false
  type: server
```

## FILES

/opt/xcat/bin/lsve

## SEE ALSO

cfgve(1)|cfgve.1

## lsvlan.1

## NAME

**lsvlan** - It lists the existing vlans for the cluster.

## SYNOPSIS

**lsvlan**

**lsvlan** [*vlanid*]

**lsvlan** [-h | --help]

**lsvlan** [-v | --version]

## DESCRIPTION

The **lsvlan** command lists all the vlans for the cluster. If *vlanid* is specified it will list more details about this vlan including the nodes in the vlan.

## PARAMETERS

*vlanid* is a unique vlan number. If it is omitted, all vlans will be listed.

## OPTIONS

**-h|--help** Display usage message.

**-v|--version** Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To list all the vlans in the cluster

```
lsvlan
```

Output is similar to:

```
vlan 3:
  subnet 10.3.0.0
  netmask 255.255.0.0

vlan 4:
  subnet 10.4.0.0
  netmask 255.255.0.0
```

2. To list the details for vlan3

```
lsvlan 3
```

Output is similar to:



```
vlan 3
  subnet 10.3.0.0
  netmask 255.255.0.0

hostname ip address node vm host
v3n1 10.3.0.1 c68m4hsp06
v3n2 10.3.0.2 x3455n01
v3n3 10.3.0.3 x3650n01
v3n4 10.3.0.4 x3650n01kvm1 x3650n01
v3n5 10.3.0.5 x3650n01kvm2 x3650n01
```

## FILES

/opt/xcat/bin/lsvlan

## SEE ALSO

mkvlan(1)|mkvlan.1, rmvlan(1)|rmvlan.1, chvlan(1)|chvlan.1

## lsvm.1

## NAME

**lsvm** - Lists information about HMC-, DFM-, IVM-, KVM-, VMware-, and zVM-managed partitions or virtual machines. For Power 775, it lists the LPARs' I/O slots information and CEC configuration.

## SYNOPSIS

**lsvm** [-h] --help]

**lsvm** [-v] --version]

**lsvm** [-V] --verbose] *noderange*

**lsvm** [-a] --all] *noderange*

### For PPC (using Direct FSP Management):

**lsvm** [-l] --long] --p775 *noderange*

**lsvm** *noderange*

## For KVM and VMware

**lsvm** *noderange*

### For zVM:

**lsvm** *noderange*

## DESCRIPTION

The **lsvm** command lists all profiles defined for the partitions or virtual machines specified in *noderange*. If *noderange* is a CEC, all the partitions associated with that CEC are displayed.

### For PPC (using Direct FSP Management):

For Power 775 (use option **--p775** to specify), **lsvm** lists all partition I/O slots information for the partitions specified in *noderange*. If *noderange* is a CEC, it gets the CEC's pump mode value, octant's memory interleaving value, the all the octants configure value, and all the I/O slots information.

For DFM-managed (short for Direct FSP Management mode) normal power machine, **lsvm** lists the processor, memory, physical I/O slots, hugepage and BSR info for the specified partitions or CEC.

#### The pump mode value has the valid options:

1 - Node Pump Mode 2 - Chip Pump Mode

#### The Memory Interleaving Mode has 3 valid options:

0 - not Applicable 1 - interleaved 2 - non-interleaved

More information about this part, refer to the section Using the \*vm commands to define partitions in xCAT DFM in the doc below.

XCAT\_Power\_775\_Hardware\_Management

## For KVM and VMware

If *noderange* is a hypervisor, virtual machines defined on that hypervisor will be displayed. If *noderange* is a VM, details for that VM will be displayed.

Note: Only the virtual machine which is in power on state can be listed by **lsvm** command.

### For zVM:

Show the directory entry for a given virtual machine.

## OPTIONS

**-h**

Display usage message.

**-v**

Command version.

**-V**

Verbose output.

**-a**

List all the profiles for one partition

**--p775**

Specify the operation is for Power 775 machines.

**-l**

Show lparnames for lpars. It shall work with option **--p775**.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To list all partition profiles defined for HMC-managed partition lpar3, enter:

```
lsvm lpar3
```

Output is similar to:

```
lpar3: name=lpar3,lpar_name=lpar3,lpar_id=4,lpar_env=aixlinux,all_resources=0,min_
↪mem=512, desired_mem=2048, max_mem=3072,min_num_huge_pages=0,desired_num_huge_pages=0,
↪max_num_huge_pages=0,proc_mode=shared, min_proc_units=0.5,desired_proc_units=0.5,max_
↪proc_units=0.5,min_procs=1,desired_procs=1,max_procs=1, sharing_mode=uncap,uncap_
↪weight=128,shared_proc_pool_id=0,shared_proc_pool_name=DefaultPool,io_slots=none, lpar_
↪io_pool_ids=none,max_virtual_slots=10, "virtual_serial_adapters=1/server/1/any//any/1,
↪0/server/1/any//any/1", virtual_scsi_adapters=2/client/1/p6vios/4/1,virtual_eth_
↪adapters=3/0/1//0/1,hca_adapters=none,boot_mode=norm,conn_monitoring=0,auto_start=0,
↪power_ctrl_lpar_ids=none,work_group_id=none,redundant_err_path_reporting=0, bsr_
↪arrays=0,lhea_logical_ports=none,lhea_capabilities=none,lpar_proc_compat_mode=default,
↪electronic_err_reporting=null
```

2. To list all IVM-managed partitions associated with CEC cec01, enter:

```
lsvm cec01
```

Output is similar to:

```
cec01: name=10-B7D1G,lpar_name=10-B7D1G,lpar_id=1,os_type=vioserver,all_resources=0,min_
→mem=512, desired_mem=2048,max_mem=2048,proc_mode=shared,min_proc_units=0.10,desired_
→proc_units=0.40, max_proc_units=4.00,min_procs=1,desired_procs=4,max_procs=4,sharing_
→mode=uncap,uncap_weight=128, "io_slots=21010002/none/0,21010003/none/0,21010004/none/0,
→21020003/none/0,21020004/none/0,21030003/none/0,21030004/none/0,21040003/none/0,
→21040004/none/0",lpar_io_pool_ids=none,max_virtual_slots=48, "virtual_serial_
→adapters=0/server/1/any//any/1,1/server/1/any//any/1,10/client/0/2/lp2/0/0,12/client/0/
→3/lp3/0/0,14/client/0/4/lp4/0/0","virtual_scsi_adapters=11/server/2/lp2/2/0,13/server/
→3/lp3/2/0,15/server/4/lp4/2/0","virtual_eth_adapters=3/0/1//1/0,4/0/2//1/0,5/0/3//1/0,
→6/0/4//1/0",boot_mode=norm,conn_monitoring=0,auto_start=0,power_ctrl_lpar_ids=none
name=lp2,lpar_name=lp2,lpar_id=2,os_type=aixlinux,all_resources=0,min_mem=128,desired_
→mem=1024,max_mem=1024,proc_mode=shared,min_proc_units=0.10,desired_proc_units=0.10,max_
→proc_units=4.00,min_procs=1,desired_procs=1,max_procs=4,sharing_mode=uncap,uncap_
→weight=128,io_slots=none,lpar_io_pool_ids=none,max_virtual_slots=6, "virtual_serial_
→adapters=0/server/1/any//any/1,1/server/1/any//any/1",virtual_scsi_adapters=2/client/1/
→10-7D1G/11/1,virtual_eth_adapters=4/0/1//0/0,boot_mode=norm,conn_monitoring=0,auto_
→start=0,power_ctrl_lpar_ids=none
name=lp3,lpar_name=lp3,lpar_id=3,os_type=aixlinux,all_resources=0,min_mem=128,desired_
→mem=128,max_mem=128,proc_mode=shared,min_proc_units=0.10,desired_proc_units=0.10,max_
→proc_units=4.00,min_procs=1,desired_procs=1,max_procs=4,sharing_mode=uncap,uncap_
→weight=128,io_slots=none,lpar_io_pool_ids=none,max_virtual_slots=6, "virtual_serial_
→adapters=0/server/1/any//any/1,1/server/1/any//any/1",virtual_scsi_adapters=2/client/1/
→10-B7D1G/13/1,virtual_eth_adapters=4/0/1//0/0,boot_mode=of,conn_monitoring=0,auto_
→start=1, power_ctrl_lpar_ids=none
```

3. For Power 775, to list the I/O slot information of lpar1, enter:

```
lsvm lpar1 --p775
```

Output is similar to:

```
1: 514/U78A9.001.0123456-P1-C17/0x21010202/2/1
1: 513/U78A9.001.0123456-P1-C15/0x21010201/2/1
1: 512/U78A9.001.0123456-P1-C16/0x21010200/2/1
```

4. To list the lparname of lpars, enter:

```
lsvm lpar1 -l --p775
```

Output is similar to:

```
lpar1: 1: 514/U78A9.001.0123456-P1-C17/0x21010202/2/1
lpar1: 1: 513/U78A9.001.0123456-P1-C15/0x21010201/2/1
lpar1: 1: 512/U78A9.001.0123456-P1-C16/0x21010200/2/1
```

5. For Power 775, to list the I/O slot information and octant configuration of cec1, enter:

```
lsvm cec1 --p775
```

Output is similar to:

```
1: 514/U78A9.001.0123456-P1-C17/0x21010202/2/1
1: 513/U78A9.001.0123456-P1-C15/0x21010201/2/1
1: 512/U78A9.001.0123456-P1-C16/0x21010200/2/1
```

(continues on next page)

(continued from previous page)

```
13: 537/U78A9.001.0123456-P1-C9/0x21010219/2/13
13: 536/U78A9.001.0123456-P1-C10/0x21010218/2/13
17: 545/U78A9.001.0123456-P1-C7/0x21010221/2/17
17: 544/U78A9.001.0123456-P1-C8/0x21010220/2/17
21: 553/U78A9.001.0123456-P1-C5/0x21010229/2/21
21: 552/U78A9.001.0123456-P1-C6/0x21010228/2/21
25: 569/U78A9.001.0123456-P1-C1/0x21010239/2/25
25: 561/U78A9.001.0123456-P1-C3/0x21010231/2/25
25: 560/U78A9.001.0123456-P1-C4/0x21010230/2/25
29: 568/U78A9.001.0123456-P1-C2/0x21010238/2/29
5: 521/U78A9.001.0123456-P1-C13/0x21010209/2/5
5: 520/U78A9.001.0123456-P1-C14/0x21010208/2/5
9: 529/U78A9.001.0123456-P1-C11/0x21010211/2/9
9: 528/U78A9.001.0123456-P1-C12/0x21010210/2/9
cec1: PendingPumpMode=1,CurrentPumpMode=1,OctantCount=8:
OctantID=0,PendingOctCfg=5,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↪CurrentMemoryInterleaveMode=2;
OctantID=1,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↪CurrentMemoryInterleaveMode=2;
OctantID=2,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↪CurrentMemoryInterleaveMode=2;
OctantID=3,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↪CurrentMemoryInterleaveMode=2;
OctantID=4,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↪CurrentMemoryInterleaveMode=2;
OctantID=5,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↪CurrentMemoryInterleaveMode=2;
OctantID=6,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↪CurrentMemoryInterleaveMode=2;
OctantID=7,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↪CurrentMemoryInterleaveMode=2;
```

6. To list the lparname of lpars, enter:

```
lsvm cec1 -l --p775
```

Output is similar to:

```
lpar1: 1: 514/U78A9.001.0123456-P1-C17/0x21010202/2/1: 32: 0/3/3
lpar1: 1: 513/U78A9.001.0123456-P1-C15/0x21010201/2/1: 32: 0/3/3
lpar1: 1: 512/U78A9.001.0123456-P1-C16/0x21010200/2/1: 32: 0/3/3
lpar13: 13: 537/U78A9.001.0123456-P1-C9/0x21010219/2/13: 32: 0/3/3
lpar13: 13: 536/U78A9.001.0123456-P1-C10/0x21010218/2/13: 32: 0/3/3
lpar17: 17: 545/U78A9.001.0123456-P1-C7/0x21010221/2/17: 32: 0/0/0
lpar17: 17: 544/U78A9.001.0123456-P1-C8/0x21010220/2/17: 32: 0/0/0
lpar21: 21: 553/U78A9.001.0123456-P1-C5/0x21010229/2/21: 32: 0/0/0
lpar21: 21: 552/U78A9.001.0123456-P1-C6/0x21010228/2/21: 32: 0/0/0
lpar24: 25: 569/U78A9.001.0123456-P1-C1/0x21010239/2/25: 32: 0/0/0
lpar25: 25: 561/U78A9.001.0123456-P1-C3/0x21010231/2/25: 32: 0/0/0
lpar25: 25: 560/U78A9.001.0123456-P1-C4/0x21010230/2/25: 32: 0/0/0
lpar29: 29: 568/U78A9.001.0123456-P1-C2/0x21010238/2/29: 32: 0/0/0
lpar5: 5: 521/U78A9.001.0123456-P1-C13/0x21010209/2/5: 32: 0/3/3
```

(continues on next page)

(continued from previous page)

```
lpar5: 5: 520/U78A9.001.0123456-P1-C14/0x21010208/2/5: 32: 0/3/3
lpar9: 9: 529/U78A9.001.0123456-P1-C11/0x21010211/2/9: 32: 0/3/3
lpar9: 9: 528/U78A9.001.0123456-P1-C12/0x21010210/2/9: 32: 0/3/3
cec1: PendingPumpMode=1,CurrentPumpMode=1,OctantCount=8:
OctantID=0,PendingOctCfg=5,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↔CurrentMemoryInterleaveMode=2;
OctantID=1,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↔CurrentMemoryInterleaveMode=2;
OctantID=2,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↔CurrentMemoryInterleaveMode=2;
OctantID=3,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↔CurrentMemoryInterleaveMode=2;
OctantID=4,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↔CurrentMemoryInterleaveMode=2;
OctantID=5,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↔CurrentMemoryInterleaveMode=2;
OctantID=6,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↔CurrentMemoryInterleaveMode=2;
OctantID=7,PendingOctCfg=1,CurrentOctCfg=1,PendingMemoryInterleaveMode=2,
↔CurrentMemoryInterleaveMode=2;
Number of BSR arrays: 256,Bytes per BSR array: 4096,Available BSR array: 0;
Available huge page memory(in pages): 0
Configurable huge page memory(in pages): 12
Page Size(in GB): 16
Maximum huge page memory(in pages): 24
Requested huge page memory(in pages): 15
Number of BSR arrays: 256,Bytes per BSR array: 4096,Available BSR array: 0;
Available huge page memory(in pages): 0
Configurable huge page memory(in pages): 12
Page Size(in GB): 16
Maximum huge page memory(in pages): 24
Requested huge page memory(in pages): 15
```

7. To list the virtual machine's directory entry:

```
lsvm gpok3
```

Output is similar to:

```
gpok3: USER LNX3 PWD 512M 1G G
gpok3: INCLUDE LNXDFLT
gpok3: COMMAND SET VSWITCH VSW2 GRANT LNX3
```

8. For DFM-managed normal power machine, list out the detailed resource information:

```
lsvm cec
```

Output is similar to:

```
cec: HYP Configurable Processors: 16, Avail Processors: 16.
HYP Configurable Memory:32.00 GB(128 regions).
HYP Available Memory: 31.25 GB(125 regions).
HYP Memory Region Size: 0.25 GB(256 MB).
```

(continues on next page)

(continued from previous page)

```
cec: All Physical I/O info:
65535,519,U78AA.001.WZSGVU7-P1-C7,0x21010207,0xffff(Empty Slot)
65535,518,U78AA.001.WZSGVU7-P1-C6,0x21010206,0xffff(Empty Slot)
65535,517,U78AA.001.WZSGVU7-P1-C5,0x21010205,0xffff(Empty Slot)
65535,516,U78AA.001.WZSGVU7-P1-C4,0x21010204,0xffff(Empty Slot)
65535,514,U78AA.001.WZSGVU7-P1-C19,0x21010202,0xffff(Empty Slot)
65535,513,U78AA.001.WZSGVU7-P1-T7,0x21010201,0xc03(USB Controller)
65535,512,U78AA.001.WZSGVU7-P1-T9,0x21010200,0x104(RAID Controller)
cec: Huge Page Memory
Available huge page memory(in pages):      2
Configurable huge page memory(in pages):    2
Page Size(in GB):                          16
Maximum huge page memory(in pages):         4
Requested huge page memory(in pages):       2
cec: Barrier Synchronization Register(BSR)
Number of BSR arrays: 256
Bytes per BSR array: 4096
Available BSR array: 256
```

Note: The lines listed in “All Physical I/O info” section represent all the physical I/O resource information. The format is like “owner\_lparid,slot\_id,physical resource name,drc\_index,slot\_class\_code(class description)”. The ‘drc index’ is short for Dynamic Resource Configuration Index, it uniquely indicates a physical I/O resource in a normal power machine.

9. For DFM-managed partition on normal power machine, list out the detailed information:

```
lsvm lpar1
```

Output is similar to:

```
lpar1: Lpar Processor Info:
Curr Processor Min: 1.
Curr Processor Req: 16.
Curr Processor Max: 16.
lpar1: Lpar Memory Info:
Curr Memory Min: 0.25 GB(1 regions).
Curr Memory Req: 30.75 GB(123 regions).
Curr Memory Max: 32.00 GB(128 regions).
lpar1: 1,519,U78AA.001.WZSGVU7-P1-C7,0x21010207,0xffff(Empty Slot)
lpar1: 1,518,U78AA.001.WZSGVU7-P1-C6,0x21010206,0xffff(Empty Slot)
lpar1: 1,517,U78AA.001.WZSGVU7-P1-C5,0x21010205,0xffff(Empty Slot)
lpar1: 1,516,U78AA.001.WZSGVU7-P1-C4,0x21010204,0xffff(Empty Slot)
lpar1: 1,514,U78AA.001.WZSGVU7-P1-C19,0x21010202,0xffff(Empty Slot)
lpar1: 1,513,U78AA.001.WZSGVU7-P1-T7,0x21010201,0xc03(USB Controller)
lpar1: 1,512,U78AA.001.WZSGVU7-P1-T9,0x21010200,0x104(RAID Controller)
lpar1: 1/2/2
lpar1: 256.
```

## FILES

/opt/xcat/bin/lsvm

## SEE ALSO

mkvm(1)|mkvm.1, chvm(1)|chvm.1, rmvm(1)|rmvm.1

## lsxcatd.1

## NAME

**lsxcatd** - lists xCAT daemon information.

## SYNOPSIS

**lsxcatd** [-h | --help | -v | --version | -d | --database | -t | --nodetype | -a | --all ]

## DESCRIPTION

The **lsxcat** command lists important xCAT daemon (xcatd) information.

## OPTIONS

**-v|--version**

Command Version.

**-h|--help**

Display usage message.

**-d|--database**

Displays information about the current database being used by xCAT.

**-t|--nodetype**

Displays whether the node is a Management Node or a Service Node.

**-a|--all**

Displays all information about the daemon supported by the command.



## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To display information about the current database:

```
lsxcatd -d
```

Output is similar to:

```
cfgloc=Pg:dbname=xcatdb;host=7.777.47.250|xcatadm
dbengine=Pg
dbname=xcatdb
dbhost=7.777.47.250
dbadmin=xcatadm
```

2. To display all information:

```
lsxcatd -a
```

Output is similar to:

```
Version 2.8.5 (git commit 0d4888af5a7a96ed521cb0e32e2c918a9d13d7cc, built Tue Jul
→29 02:22:47 EDT 2014)
This is a Management Node
cfgloc=mysql:dbname=xcatdb;host=9.114.34.44|xcatadmin
dbengine=mysql
dbname=xcatdb
dbhost=9.114.34.44
dbadmin=xcatadmin
```

## FILES

/opt/xcat/bin/lsxcatd

## SEE ALSO

**makentp.1**

## SYNOPSIS

**makentp** [-h|--help]

**makentp** [-v|--version]

**makentp** [-a|--all] [-V|--verbose]

## DESCRIPTION

**makentp** command sets up the NTP server on the xCAT management node and the service node.

By default, it sets up the NTP server for xCAT management node. If **-a** flag is specified, the command will setup the ntp servers for management node as well as all the service nodes that have *servicenode.ntpserver* set. It honors the site table attributes *extntpservers* and *ntpservers* described below:

*site.extntpservers* – the NTP servers for the management node to sync with. If it is empty then the NTP server will use the management node's own hardware clock to calculate the system date and time.

*site.ntpservers* – the NTP servers for the service node and compute node to sync with. The keyword <xcatmaster> means that the node's NTP server is the node that is managing it (either its service node or the management node).

To setup NTP on the compute node, add *setupntp* postscript to the *postscripts* table and run **updatenode node -P setupntp** command.

## OPTIONS

### **-a|--all**

Setup NTP servers for both management node and the service node. If management node has SLES installed and used as *ntpservers*, it is recommended to use the *setupntp* postscript to set up NTP server for service nodes.

### **-h|--help**

Display usage message.

### **-v|--version**

Command Version.

### **-V|--verbose**

Verbose output.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To setup NTP server on the management node:

```
makentp
```

2. To setup NTP servers on both management node and the service node:

```
makentp -a
```

## FILES

/opt/xcats/bin/makentp

## SEE ALSO

mkdef.1

## NAME

**mkdef** - Use this command to create xCAT data object definitions.

## SYNOPSIS

**mkdef** [-h | --help] [-t *object-types*]

**mkdef** [-V | --verbose] [-t *object-types*] [--template *template-object-name*] [-o *object-names*] [-z | --stanza] [-d | --dynamic] [-f | --force] [[-w *attr==val*] [-w *attr=~val*] ...] [*noderange*] [*attr=val* [*attr=val...*]] [-u *provmethod*={install | netboot | statelite} *profile*= *xxx* [*osvers*= *value*] [*osarch*= *value*]]

## DESCRIPTION

The **mkdef** command is used to create xCAT object definitions which are stored in the xCAT database. If the definition already exists it will return an error message. The **--force** option may be used to re-create a definition. In this case the old definition will be remove and the new definition will be created.

## OPTIONS

*attr=val* [*attr=val...*]

Specifies one or more “attribute equals value” pairs, separated by spaces. *attr=val* pairs must be specified last on the command line. Use the **--help** option to get a list of valid attributes for each object type.

Note: when creating node object definitions, the **groups** attribute is required.

### -d|--dynamic

Use the dynamic option to create dynamic node groups. This option must be used with **-w** option.

### -f|--force

Use the force option to re-create object definitions. This option removes the old definition before creating the new one.

### -h|--help

Display usage message.

*noderange*

A set of comma delimited node names and/or group names. (must be the first parameter) See the <noderange> man page for details on supported formats.

**-o** *object-names*

A set of comma delimited object names.

**-t** *object-types*

A set of comma delimited object types. Use the **--help** option to get a list of valid object types.

**--template** *template-object-name*

Name of the xCAT shipped object definition template or an existing object, from which the new object definition will be created. The newly created object will inherit the attributes of the template definition unless the attribute is specified in the arguments of **mkdef** command. If there are a template and an existing object with the same name *template-object-name*, the template object takes precedence over the existing object. For the details of xCAT shipped object definition templates, refer to the manpage of **--template** option in `lsdef(1)|lsdef.1`.

**-V|--verbose**

Verbose mode.

**-w attr==val -w attr=~val ...**

Use one or multiple **-w** flags to specify the selection string that can be used to select objects. The operators **==**, **!=**, **==~** and **!~** are available. For **mkdef** command, the **-w** flag only makes sense for creating dynamic node group. Use the **--help** option to get a list of valid attributes for each object type.

Operator descriptions:

**==** Select nodes where the attribute value is exactly this value.

**!=** Select nodes where the attribute value is not this specific value.

**==~** Select nodes where the attribute value matches this regular expression.

**!~** Select nodes where the attribute value does not match this regular expression.

Note: if the “val” field includes spaces or any other characters that will be parsed by shell, the “attr<operator>val” needs to be quoted. If the operator is **!~**, the “attr<operator>val” needs to be quoted using single quote.

**-z|--stanza**

Indicates that the file being piped to the command is in stanza format. See the **xcatzstanzafile** man page for details on using xCAT stanza files.

**-u**

Fill in the attributes such as template file, pkglist file and otherpkglist file of **osimage** object based on the specified parameters. It will search **/install/custom/** directory first, and then **/opt/xcats/share/**. The *provmethod* and *profile* must be specified. If *osvers* or *osarch* is not specified, the corresponding value of the management node will be used.

Note: this option only works for object type **osimage**.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To create a site definition.

```
mkdef -t site -o clustersite installdir=/xcatinstall
```

2. To create a basic node definition.

```
mkdef -t node -o node01 groups="all,aix"
```

3. To re-create the current definition of “node01”.

```
mkdef -f -t node -o node01 nodetype=osi groups="linux"
```

(The group definitions are also created if they don’t already exist.)

4. To create a set of different types of definitions based on information contained in a stanza file.

```
cat defstanzafile | mkdef -z
```

5. To create a group definition called LinuxNodes containing the nodes clstrn01 and clstrn02.

```
mkdef -t group -o LinuxNodes members="clstrn01,clstrn02"
```

6. To create a node definition for an FSP node using the attributes provided by the group fspnodes.

```
mkdef -t node fspn1 groups=fspnodes nodetype=fsp
```

7. To create node definitions for a set of node host names contained in the node range “node1,node2,node3”

```
mkdef -t node node1,node2,node3 power=hmc groups="all,aix"
```

8. To create a dynamic node group definition called HMCgtNodes containing all the HMC managed nodes”

```
mkdef -t group -o HMCgtNodes -d -w mgt==hmc -w cons==hmc
```

9. To create a dynamic node group definition called SLESNodes containing all the SLES nodes

```
mkdef -t group -o SLESNodes -d -w "os=~^sles[0-9]+$"
```

10. To create a entry (7.0) in the policy table for user admin1

```
mkdef -t policy -o 7.0 name=admin1 rule=allow
```

11. To create a node definition with nic attributes

```
mkdef -t node cn1 groups=all nicips.eth0="1.1.1.1|1.2.1.1" nicnetworks.eth0=
↪ "net1|net2" nictypes.eth0="Ethernet"
```

12. To create an osimage definition and fill in attributes automatically.

```
mkdef redhat6img -u profile=compute provmethod=statelite
```

13. To create a PowerLE kvm node definition with the xCAT shipped template “ppc64lekvmguest-template”.

```
mkdef -t node cn1 --template ppc64lekvmguest-template ip=1.1.1.1  
↪mac=42:3d:0a:05:27:0b vmhost=1.1.0.1 vmnics=br0
```

14. To create a node definition from an existing node definition “cn1”

```
mkdef -t node cn2 --template cn1 ip=1.1.1.2 mac=42:3d:0a:05:27:0c
```

15. To create a dynamic group definition based on nodename, which will include all nodes named “fs2vm\*”

```
mkdef -t group mg_vms -d -w node=~"/fs2vm.*"
```

## FILES

\$XCATROOT/bin/mkdef

(The XCATROOT environment variable is set when xCAT is installed. The default value is “/opt/xcat”.)

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

chdef(1)|chdef.1, lsdef(1)|lsdef.1, rmdef(1)|rmdef.1, xcatstanzafile(5)|xcatstanzafile.5

## mkdsklsnode.1

### NAME

**mkdsklsnode** - Use this xCAT command to define and initialize AIX/NIM diskless machines.

## SYNOPSIS

**mkdsklsnode** [-h|--help]

**mkdsklsnode** [-V|--verbose] [-f|--force] [-n|--newname] [-i *osimage\_name*] [-l *location*] [-u | --updateSN] [-k | --skipsync] [-p | --primarySN] [-b | --backupSN] [-S | --setuphanfs] *noderange* [*attr=val* [*attr=val* ...]]

## DESCRIPTION

This xCAT command can be used to define and/or initialize AIX/NIM diskless machines. Once this step is completed you can use either the xCAT **rnetboot** command or the **rbootseq/rpower** commands to initiate a network boot of the nodes.

The command can be used to define and initialize a new NIM machine object or it can be used to reinitialize an existing machine to use a different operating system image.

This command will also create a NIM `resolv_conf` resource to be used when installing the node. If a `resolv_conf` resource is not already included in the xCAT `osimage` definition and if the “domain” and “nameservers” values are set then a new NIM `resolv_conf` resource will be created and allocated to the nodes.

The “domain” and “nameservers” attributes can be set in either the xCAT “network” definition used by the nodes or in the xCAT cluster “site” definition. The setting in the “network” definition will take priority.

The “search” field of the `resolv.conf` file will contain a list all the domains listed in the xCAT network definitions and the xCAT site definition.

The “nameservers” value can either be set to a specific IP address or the “<xcatmaster>” key word. The “<xcatmaster>” key word means that the value of the “xcatmaster” attribute of the node definition will be used in the `/etc/resolv.conf` file. (I.e. The name of the install server as known by the node.)

You can set the “domain” and “nameservers” attributes by using the **chdef** command. For example:

```
chdef -t network -o clstr_net domain=cluster.com nameservers=<xcatmaster>
```

If the “domain” and “nameservers” attributes are not set in either the nodes “network” definition or the “site” definition then no new NIM `resolv_conf` resource will be created.

If you are using xCAT service nodes the **mkdsklsnode** command will automatically determine the correct server(s) for the node and create the NIM definitions on that server(s).

When creating a new NIM machine definition the default is to use the same name as the xCAT node name that is provided.

You can use the “-n” option of the **mkdsklsnode** command to create and initialize an alternate NIM machine definition for the same physical nodes. This option allows you to set up a new image to use when a node is next rebooted while the node is currently running. This is possible because the NIM name for a machine definition does not have to be the hostname of the node. This allows you to have multiple NIM machine definitions for the same physical node. The naming convention for the new NIM machine name is “<xcat\_node\_name>\_<image\_name>”, (Ex. “node01\_61spot”). Since all the NIM initialization can be done while the node is running the downtime for the node is reduced to the time it takes to reboot.

**Note:** When using the “-n” option make sure that the new `osimage` you specify and all the NIM resources that are used are different than what are currently being used on the nodes. The NIM resources should not be shared between the old `osimage` and the new `osimage`.

You can use the force option to reinitialize a node if it already has resources allocated or it is in the wrong NIM state. This option will reset the NIM node and deallocate resources before reinitializing. Use this option with caution since reinitializing a node will stop the node if it is currently running.

After the **mkdsklsnode** command completes you can use the **lsnim** command to check the NIM node definition to see if it is ready for booting the node. (“**lsnim** -l <nim\_node\_name>”).

You can supply your own scripts to be run on the management node or on the service node (if there is hierarchy) for a node during the **mkdsklsnode** command. Such scripts are called **prescripts**. They should be copied to `/install/prescripts` directory. A table called *prescripts* is used to specify the scripts and their associated actions. The scripts to be run at the beginning of the **mkdsklsnode** command are stored in the ‘begin’ column of *prescripts* table. The scripts to be run at the end of the **mkdsklsnode** command are stored in the ‘end’ column of *prescripts* table. Run ‘**tabdump** prescripts -d’ command for details. An example for the ‘begin’ or the ‘end’ column is: *diskless:myscript1,myscript2*.

The following two environment variables will be passed to each script: `NODES` contains all the names of the nodes that need to run the script for and `ACTION` contains the current nodeset action, in this case “diskless”. If `#xCAT setting:MAX_INSTANCE=number` is specified in the script, the script will get invoked for each node in parallel, but no more than *number* of instances will be invoked at a time. If it is not specified, the script will be invoked once for all the nodes.

## OPTIONS

*attr=val [attr=val ...]*

Specifies one or more “attribute equals value” pairs, separated by spaces. `Attr= val` pairs must be specified last on the command line. These are used to specify additional values that can be passed to the underlying NIM commands.

Valid values:

### **duplex**

Specifies the duplex setting (optional). Used when defining the NIM machine. Use this setting to configure the client’s network interface. This value can be full or half. The default is full. (ex. “duplex=full”)

### **speed**

Specifies the speed setting (optional). Used when defining the NIM machine. This is the communication speed to use when configuring the client’s network interface. This value can be 10, 100, or 1000. The default is 100. (ex. “speed=100”)

### **psize**

Specifies the size in Megabytes of the paging space for the diskless node.(optional) Used when initializing the NIM machine. The minimum and default size is 64 MB of paging space. (ex. “psize=256”)

### **sparse\_paging**

Specifies that the paging file should be created as an AIX sparse file, (ex. “sparse\_paging=yes”). The default is “no”.

### **dump\_iscsi\_port**

The tcpip port number to use to communicate dump images from the client to the dump resource server. Normally set by default. This port number is used by a dump resource server.

### **configdump**

Specifies the type dump to be collected from the client. The values are “selective”, “full”, and “none”. If the configdump attribute is set to “full” or “selective” the client will automatically be configured to dump to an iSCSI target device. The “selective” memory dump will avoid dumping user data. The “full” memory dump will dump all the memory of the client partition. Selective and full memory dumps will be stored in subdirectory of the dump resource allocated to the client. This attribute is saved in the xCAT osimage definition.

### **-b|--backupSN**

When using backup service nodes only update the backup. The default is to update both the primary and backup service nodes.

### **-f|--force**

Use the force option to reinitialize the NIM machines.



**-h|--help**

Display usage message.

**-i *image\_name***

The name of an existing xCAT osimage definition. If this information is not provided on the command line the code checks the node definition for the value of the “provmethod” attribute. If the “-i” value is provided on the command line then that value will be used to set the “provmethod” attribute of the node definitions.

**-k|--skipsync**

Use this option to have the mkdsklnode command skip the NIM sync\_roots operation. This option should only be used if you are certain that the shared\_root resource does not have to be updated from the SPOT. Normally, when the SPOT is updated, you should do a sync\_roots on the shared\_root resource.

**-l|--location**

The directory location to use when creating new NIM resolv\_conf resources. The default location is /install/nim.

**-n|--newname**

Create a new NIM machine object name for the xCAT node. Use the naming convention “<xcat\_node\_name>\_<image\_name>” for the new NIM machine definition.

**-p|--primarySN**

When using backup service nodes only update the primary. The default is to update both the primary and backup service nodes.

**-S|--setupnfs**

Setup NFSv4 replication between the primary service nodes and backup service nodes to provide high availability NFS for the compute nodes. This option only exports the /install directory with NFSv4 replication settings, the data synchronization between the primary service nodes and backup service nodes needs to be taken care of through some mechanism.

**-u|--updateSN**

Use this option if you wish to update the osimages but do not want to define or initialize the NIM client definitions. This option is only valid when the xCAT “site” definition attribute “sharedinstall” is set to either “sns” or “all”.

*noderange*

A set of comma delimited node names and/or group names. See the “noderange” man page for details on additional supported formats.

**-V|--verbose**

Verbose mode.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. Initialize an xCAT node named “node01” as an AIX diskless machine. The xCAT osimage named “61spot” should be used to boot the node.

```
mkdsklsnode -i 61spot node01
```

2. Initialize all AIX diskless nodes contained in the xCAT node group called “aixnodes” using the image definitions pointed to by the “provmethod” attribute of the xCAT node definitions.

```
mkdsklsnode aixnodes
```

3. Initialize diskless node “clstrn29” using the xCAT osimage called “61dskls”. Also set the paging size to be 128M and specify the paging file be an AIX sparse file.

```
mkdsklsnode -i 61dskls clstrn29 psize=128 sparse_paging=yes
```

4. Initialize an xCAT node called “node02” as an AIX diskless node. Create a new NIM machine definition name with the osimage as an extension to the xCAT node name.

```
mkdsklsnode -n -i 61spot node02
```

## FILES

/opt/xcat/bin/mkdsklsnode

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

rmnsklsnode(1)|rmnsklsnode.1

**mkflexnode.1**

## NAME

**mkflexnode** - Create a flexible node.

## SYNOPSIS

**mkflexnode** [-h | --help]

**mkflexnode** [-v | --version]

**mkflexnode** *noderange*

## DESCRIPTION

A flexible node is a **Partition** in a complex. Creating a flexible node is to create a partition which including all the slots defined in the xCAT blade node.

Before creating a flexible node, a general xCAT blade node should be defined. The *id* attribute of this node should be a node range like 'a-b', it means the blades installed in slots 'a-b' need to be assigned to the partition. 'a' is the start slot, 'b' is the end slot. If this partition only have one slot, the slot range can be 'a'.

The action of creating flexible node will impact the hardware status. Before creating it, the blades in the slot range should be in **power off** state.

After the creating, use the **lsflexnode** to check the status of the node.

The *noderange* only can be a blade node.

## OPTIONS

**-h | --help**

Display the usage message.

**-v | --version**

Display the version information.

## EXAMPLES

1. Create a flexible node base on the xCAT node blade1.

The blade1 should belong to a complex, the *id* attribute should be set correctly and all the slots should be in **power off** state.

```
mkflexnode blade1
```

## FILES

/opt/xcat/bin/mkflexnode

## SEE ALSO

lsflexnode(1)|lsflexnode.1, rmflexnode(1)|rmflexnode.1

## mkhwconn.1

## NAME

**mkhwconn** - Sets up connections for CEC and Frame nodes to HMC nodes or hardware server.

## SYNOPSIS

**mkhwconn** [-h| --help]

**mkhwconn** [-v| --version]

### PPC (with HMC) specific:

**mkhwconn** [-V| --verbose] *noderange* -t [--port *port\_value*]

**mkhwconn** [-V| --verbose] *noderange* -s [*hmcnode* --port *port\_value*]

**mkhwconn** [-V| --verbose] *noderange* -p *hmc* [-P *passwd*] [--port *port\_value*]

### PPC (using Direct FSP Management) specific:

**mkhwconn** *noderange* -t [-T *tooltype*] [--port *port\_value*]

## DESCRIPTION

For PPC (with HMC) specific:

This command is used to set up connections for CEC and Frame nodes to HMC nodes. (If the connection already exists, it will not break it.) This command is useful when you have multiple HMCs, each of which will manage a subset of the CECs/Frames. Use **mkhwconn** to tell each HMC which CECs/Frames it should manage. When using this, you should turn off the self-discovery on each HMC. You also need to put all the HMCs and all the Frames on a single flat service network.

When **-t** is specified, this command reads the connection information from the xCAT *ppc* table (e.g. the parent attribute), and read the user/password from the *ppcdirect* table. Then this command will assign CEC nodes and Frame nodes to HMC nodes.

When **-p** is specified, this command gets the connection information from command line arguments. If **-P** is not specified, the default password for CEC and Frame nodes is used.

The flag **-s** is used to make the connection between the frame and its Service focal point(HMC). Makehwconn will also set the connections between the CECs within this Frame and the HMC. The *sfp* of the frame/CEC can either be defined in *ppc* table beforehand or specified in command line after the flag **-s**. If the user use **mkhwconn noderange -s HMC\_name**, it will not only make the connections but also set the *sfp* attributes for these nodes in *PPC* table.

In any case, before running this command, the CEC and Frame nodes need be defined with correct *nodetype*.*nodetype* value (cec or frame) and *nodehm.mgt* value (hmc).

Note: If a CEC belongs to a frame, which has a BPA installed, this CEC should not be assigned to an HMC individually. Instead, the whole frame should be assigned to the HMC.

For PPC (using Direct FSP Management) specific:

It is used to set up connections for CEC and Frame node to Hardware Server on management node (or service node ). It only could be done according to the node definition in xCAT DB. And this command will try to read the user/password from the **ppcdirect** table first. If fails, then read them from **passwd** table. Commonly , the username is **HMC**. If using the **ppcdirect** table, each CEC/Frame and user/password should be stored in **ppcdirect** table. If using the **passwd** table, the key should be “**cec**” or “**frame**”, and the related user/password are stored in **passwd** table.

When **--port** is specified, this command will create the connections for CECs/Frames whose side in **vpd** table is equal to port value.

## OPTIONS

### **-h|--help**

Display usage message.

### **-t**

Read connection information from xCAT DB (**ppc** and **ppcdirect** tables). Use this option if you need to connect multiple CECs/Frames to multiple HMCs in a single command.

### **-p**

The HMC node name. Only one HMC nodes can be specified by this flag. To setup connection for multiple HMC nodes, use flag **-t**.

### **-P**

The password of HMC based CEC/Frame login user(Default user name is ‘HMC’). This flag is optional.

### **-T**

The tooltype is used to communicate to the CEC/Frame. The value could be **lpar** or **fnm**. The tooltype value **lpar** is for xCAT and **fnm** is for CNM. The default value is “**lpar**”.

### **--port**

The port value specifies which special side will be used to create the connection to the CEC/Frame. The value could only be specified as “**0**” or “**1**” and the default value is “**0,1**”. If the user wants to use all ports to create the connection, he should not specify this value. If the port value is specified as “**0**”, in the **vpd** table, the side column should be **A-0** and **B-0**; If the port value is specified as “**1**”, the side column should be **A-1** and **B-1**. When making hardware connection between CEC/Frame and HMC, the value is used to specify the fsp/bpa port of the cec/frame and will be organized in order of “**A-0,A-1,B-0,B-1**”. If any side does not exist, the side would simply be ignored. Generally, only one port of a fsp/bap can be connected while another port be used as backup.

### **-s**

The flag **-s** is used to make the connection between the frame and its Service Focal Point(HMC). **-s** flag is not supposed to work with other functional flags.

### **-V|--verbose**

Verbose output.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To setup the connection for all CEC nodes in node group cec to HMC node, according to the definition in xCAT DB:

```
mkhwconn cec -t
```

2. To setup the connection for Frame nodes in node group frame to HMC node hmc1, with password 'abc123':

```
mkhwconn frame -p hmc1 -P abc123
```

3. To setup the connections for all CEC nodes in node group cec to hardware server, and the tooltype value is lpar:

```
mkhwconn cec -t -T lpar
```

4. To setup the connections for all cecs nodes in node group cec to hardware server, and the tooltype value is lpar, and the port value is 1:

```
mkhwconn cec -t -T lpar --port 1
```

5. To setup the connection between the frame and it's SFP node. This command will also set the connections between the CECs within this frame and their SFP node. User need to define HMC\_name in the database in advance, but no need to set the sfp attribute for these node, xCAT will set the HMC\_name as ppc.sfp for these nodes. The CECs within this frame should have the same sfp attribute as the frame.

```
mkhwconn cec -s HMC_name -P HMC_passwd
```

## FILES

\$XCATROOT/bin/mkhwconn

(The XCATROOT environment variable is set when xCAT is installed. The default value is "/opt/xcat".)

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

lshwconn(1)|lshwconn.1, rmhwconn(1)|rmhwconn.1

## mknimimage.1

### NAME

**mknimimage** - Use this xCAT command to create xCAT osimage definitions and related AIX/NIM resources. The command can also be used to update an existing AIX diskless image(SPOT).

### SYNOPSIS

**mknimimage** [-h | --help ]

**mknimimage** [-V] -u *osimage\_name* [attr=val [attr=val ... ]]

**mknimimage** [-V] [-f|--force] [-r|--sharedroot] [-D|--mkdumpres] [-l *location*] [-c | --completeosimage] [-s *image\_source*] [-i *current\_image*] [-p | --cplpp] [-t *nimtype*] [-m *nimmethode*] [-n *mksysbnode*] [-b *mksysbfile*] *osimage\_name* [attr=val [attr=val ... ]]

### DESCRIPTION

This command will create both an xCAT osimage definition and the corresponding NIM resource definitions. The command can also be used to update an existing AIX diskless image(SPOT).

The command will also install the NIM master software and configure NIM if needed.

The naming convention for the NIM SPOT resource definition is to use the same name as the xCAT osimage. The naming convention for any other NIM resources that are created is "<osimage\_name>\_<resource\_type>". (ex. "6limage\_lpp\_source" )

When creating a mksysb image definition you must specify either the "-n" or the "-b" option. The "-n" option can be used to create a mksysb image from an existing NIM client machine. The "-b" option can be used to specify an existing mksysb backup file.

#### Adding software and configuration files to the osimage.

When creating a diskless osimage definition you also have the option of automatically updating the NIM SPOT resource. You can have additional software installed or you can have configuration files added or updated. To have software installed you must provide either the names of NIM installp\_bundle resources or filesset names on the command line using the "attr=val" option. You may also supply the installp flags, RPM flags, emgr flags to use when installing the software.

To have configuration files updated you must provide the full path name of a "synclists" file which contains the list of actual files to update. The xCAT osimage definition that is created will contain the installp\_bundle, otherpkgs, and synclists files that are provided on the command line.

#### Updating an existing xCAT osimage

If you wish to update an existing diskless image after it has already been created you can use the "-u" (update) option. In this case the xCAT osimage definition will not be updated.

There are two ways to use the update feature.

You can update the osimage definition and run the **mknimimage** command with no “installp\_bundle”, “otherpkgs”, or “synclists” command line values. The information for updating the SPOT will come from the osimage definition only. This has the advantage of keeping a record of any changes that were made to the SPOT.

Or, you could do a more ad hoc update by providing one or more of the “installp\_bundle”, “otherpkgs”, or “synclists” values on the command line. If any of these values are provided the **mknimimage** command will use those values only. The osimage definition will not be used or updated.

**WARNING:** Installing random RPM packages in a SPOT may have unpredictable consequences. The SPOT is a very restricted environment and some RPM packages may corrupt the SPOT or even hang your management system. Try to be very careful about the packages you install. When installing RPMs, if the **mknimimage** command hangs or if there are file systems left mounted after the command completes you may need to reboot your management node to recover. This is a limitation of the current AIX support for diskless systems

### **Copying an xCAT osimage.**

You can use the “-i” and “-p” options to copy an existing diskless osimage. To do this you must supply the name of an existing xCAT osimage definition and the name of the new osimage you wish to create. The **mknimimage** command will do the following:

- create a new xCAT osimage definition using the new name that was specified.
- copy the NIM SPOT resource to a new location and define it to NIM using a new name.
- if the original osimage included a NIM “shared\_root” resource then a new shared\_root resource will be created for the new SPOT.
- any other resources (or attributes) included in the original osimage will be included in the new osimage definition.
- if the “-p” option is specified then the original NIM lpp\_source resource will be copied to a new location and redefined to NIM. (The default would be to use the original lpp\_source - to save file system space.)

### **Additional information**

**IMPORTANT:** The NIM lpp\_source and SPOT resources can get quite large. Always make sure that you have sufficient file system space available before running the **mknimimage** command.

To list the contents of the xCAT osimage definition use the xCAT **lsdef** command (“lsdef -t osimage -l -o <osimage\_name>”).

To check the validity of a SPOT or lpp\_source resource

To remove an xCAT osimage definition along with the associated NIM resource definitions use the **rmnimimage** command. Be careful not to accidentally remove NIM resources if they are still needed.

To list a NIM resource definition use the AIX **lsnim** command (“lsnim -l <resource\_name>”).

To check the validity of a SPOT or lpp\_source resource use the AIX **nim** command (“nim -o check <resource-name>”).

To remove specific NIM resource definitions use the AIX **nim** command. (“nim -o remove <resource-name>”).

## **OPTIONS**

*attr=val [attr=val ...]*

Specifies one or more “attribute equals value” pairs, separated by spaces. Attr=val pairs must be specified last on the command line.

Currently supported attributes:

### **bosinst\_data**

The name of a NIM bosinst\_data resource.



**dump**

The name of the NIM dump resource.

**fb\_script**

The name of a NIM fb\_script resource.

**home**

The name of the NIM home resource.

**installp\_bundle**

One or more comma separated NIM installp\_bundle resources.

**lpp\_source**

The name of the NIM lpp\_source resource.

**mksysb**

The name of a NIM mksysb resource.

**otherpkgs**

One or more comma separated installp, emgr, or rpm packages. The packages must have prefixes of 'I:', 'E:', or 'R:', respectively. (ex. R:foo.rpm)

**paging**

The name of the NIM paging resource.

**resolv\_conf**

The name of the NIM resolv\_conf resource.

**root**

The name of the NIM root resource.

**script**

The name of a NIM script resource.

**shared\_home**

The name of the NIM shared\_home resource.

**shared\_root**

A shared\_root resource represents a directory that can be used as a / (root) directory by one or more diskless clients.

**spot**

The name of the NIM SPOT resource.

**synclists**

The fully qualified name of a file containing a list of files to synchronize on the nodes.

**tmp**

The name of the NIM tmp resource.

**installp\_flags**

The alternate flags to be passed along to the AIX installp command. (The default for installp\_flags is "-abgQXY".)

**rpm\_flags**

The alternate flags to be passed along to the AIX rpm command. (The default for rpm\_flags is “-Uvh “.) The mknimimage command will check each rpm to see if it is installed. It will not be reinstalled unless you specify the appropriate rpm option, such as ‘-replacepkgs’.

**emgr\_flags**

The alternate flags to be passed along to the AIX emgr command. (There is no default flags for the emgr command.)

**dumpsize**

The maximum size for a single dump image the dump resource will accept. Space is not allocated until a client starts to dump. The default size is 50GB. The dump resource should be large enough to hold the expected AIX dump and snap data.

**max\_dumps**

The maximum number of archived dumps for an individual client. The default is one.

**snapcollect**

Indicates that after a dump is collected then snap data should be collected. The snap data will be collected in the clients dump resource directory. Values are “yes” or “no”. The default is “no”.

**nfs\_vers**

Value Specifies the NFS protocol version required for NFS access.

**nfs\_sec**

Value Specifies the security method required for NFS access.

Note that you may specify multiple “script”, “otherpkgs”, and “installp\_bundle” resources by using a comma separated list. (ex. “script=ascript,bscript”). RPM names may be included in the “otherpkgs” list by using a “R:” prefix(ex. “R:whatever.rpm”). epkg (AIX interim fix package) file names may be included in the “otherpkgs” using the ‘E:’ prefix. (ex. “otherpkgs=E:IZ38930TL0.120304.epkg.Z”).

**-b mksysbfile**

Used to specify the path name of a mksysb file to use when defining a NIM mksysb resource.

**-c|--completeosimage**

Complete the creation of the osimage definition passed in on the command line. This option will use any additional values passed in on the command line and/or it will attempt to create required resources in order to complete the definition of the xCAT osimage. For example, if the osimage definition is missing a spot or shared\_root resource the command will create those resources and add them to the osimage definition.

**-f|--force**

Use the force option to re-create xCAT osimage definition. This option removes the old definition before creating the new one. It does not remove any of the NIM resource definitions named in the osimage definition. Use the **rmnimimage** command to remove the NIM resources associated with an xCAT osimage definition.

**-h|--help**

Display usage message.

*osimage\_name*

The name of the xCAT osimage definition. This will be used as the name of the xCAT osimage definition as well as the name of the NIM SPOT resource.

**-D|--mkdumpres**

Create a diskless dump resource.

**-i *current\_image***

The name of an existing xCAT osimage that should be copied to make a new xCAT osimage definition. Only valid when defining a “diskless” or “dataless” type image.

**-l *location***

The directory location to use when creating new NIM resources. The default location is /install/nim.

**-m *nimmethode***

Used to specify the NIM installation method to use. The possible values are “rte” and “mksysb”. The default is “rte”.

**-n *mksysbnode***

The xCAT node to use to create a mksysb image. The node must be defined as a NIM client machine.

**-p|--cplpp**

Use this option when copying existing diskless osimages to indicate that you also wish to have the lpp\_resource copied. This option is only valid when using the “-i” option.

**-r|--sharedroot**

Use this option to specify that a NIM “shared\_root” resource be created for the AIX diskless nodes. The default is to create a NIM “root” resource. This feature is only available when using AIX version 6.1.4 or beyond. See the AIX/NIM documentation for a description of the “root” and “shared\_root” resources.

**-s *image\_source***

The source of software to use when creating the new NIM lpp\_source resource. This could be a source directory or a previously defined NIM lpp\_source resource name.

**-t *nimtype***

Used to specify the NIM machine type. The possible values are “standalone”, “diskless” or “dataless”. The default is “standalone”.

**-u**

Used to update an AIX/NIM SPOT resource with additional software and configuration files. This option is only valid for xCAT diskless osimage objects. The SPOT resource associated with the xCAT osimage definition will be updated. This option can also be used to update the nfs\_vers attribute from NFSv3 to NFSv4 for the NIM resources associated with diskful or diskless image.

**-V |--verbose**

Verbose mode.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## EXAMPLES

- 1) Create an osimage definition and the basic NIM resources needed to do a NIM “standalone” “rte” installation of node “node01”. Assume the software contained on the AIX product media has been copied to the /AIX/instimages directory.

```
mknimimage -s /AIX/instimages 61image
```

- 2) Create an osimage definition that includes some additional NIM resources.

```
mknimimage -s /AIX/instimages 61image installp_bundle=mybndlres,addswbnd
```

This command will create lpp\_source, spot, and bosinst\_data resources using the source specified by the “-s” option. The installp\_bundle information will also be included in the osimage definition. The mybndlres and addswbnd resources must be created before using this osimage definition to install a node.

- 3) Create an osimage definition that includes a mksysb image and related resources.

```
mknimimage -m mksysb -n node27 newsysb spot=myspot bosinst_data=mybdata
```

This command will use node27 to create a mksysb backup image and use that to define a NIM mksysb resource. The osimage definition will contain the name of the mksysb resource as well as the spot and bosinst\_data resource.

- 4) Create an osimage definition using a mksysb image provided on the command line.

```
mknimimage -m mksysb -b /tmp/backups/mysysbimage newsysb spot=myspot bosinst_data=mybdata
```

This command defines a NIM mksysb resource using mysysbimage.

- 5) Create an osimage definition and create the required spot definition using the mksysb backup file provided on the command line.

```
mknimimage -m mksysb -b /tmp/backups/mysysbimage newsysb bosinst_data=mybdata
```

This command defines a NIM mksysb resource and a spot definition using mysysbimage.

- 6) Create a diskless image called 61dskls using the AIX source files provided in the /AIX/instimages directory.

```
mknimimage -t diskless -s /AIX/instimages 61dskls
```

- 7) Create a diskless image called “614dskls” that includes a NIM “shared\_root” and a “dump” resource. Use the existing NIM lpp\_resource called “614\_lpp\_source”. Also specify verbose output.

```
mknimimage -V -r -D -t diskless -s 614_lpp_source 614dskls snapcollect=yes
```

The “snapcollect” attribute specifies that AIX “snap” data should be include when a system dump is initiated.

- 8) Create a new diskless image by copying an existing image.

```
mknimimage -t diskless -i 61cosi 61cosi_updt1
```

Note: If you also wish to have the original lpp\_source copied and defined use the -p option.

```
mknimimage -t diskless -i 61cosi -p 61cosi_updt1
```

- 9) Create a diskless image using an existing lpp\_source resource named “61cosi\_lpp\_source” and include NIM tmp and home resources. This assumes that the “mytmp” and “myhome” NIM resources have already been created by using NIM commands.

```
mknimimage -t diskless -s 61cosi_lpp_source 611cosi tmp=mytmp home=myhome
```

- 10) Create a diskless image and update it with additional software using rpm flags and configuration files.

```
mknimimage -t diskless -s 61cosi_lpp_source 61dskls otherpkgs=I:fset1,R:foo.rpm,  
↪E:IZ38930TL0.120304.epkg.Z synclists=/install/mysyncfile rpm_flags="-i --nodeps"
```

The xCAT osimage definition created by this command will include the “otherpkgs” and “synclists” values. The NIM SPOT resource associated with this osimage will be updated with the additional software using rpm flags “-i --nodeps” and configuration files.

- 11) Update an existing diskless image (AIX/NIM SPOT) using the information saved in the xCAT “61dskls” osimage definition. Also specify verbose messages.

```
mknimimage -V -u 61dskls
```

- 12) Update an existing diskless image called “61dskls”. Install the additional software specified in the NIM “bndres1” and “bndres2” installp\_bundle resources using the installp flags “-agcQX”. (The NIM “bndres1” and “bndres2” definitions must be created before using them in this command.)

```
mknimimage -u 61dskls installp_bundle=bndres1,bndres2 installp_flags="-agcQX"
```

Note that when “installp\_bundle”, “otherpkgs”, or “synclists” values are specified with the “-u” option then the xCAT osimage definition is not used or updated.

- 13) Update an existing image to support NFSv4. Also specify verbose messages.

```
mknimimage -V -u 61dskls nfs_vers=4
```

## FILES

/opt/xcat/bin/mknimimage

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

`rmnimimage(1)``|rmnimimage.1`

## mkvlan.1

## NAME

**mkvlan** - It takes a list of nodes and create a private tagged vlan for them.

## SYNOPSIS

**mkvlan** [*vlanid*] **-n** | **--nodes** *noderange* [**-t** | **--net** *subnet*] [**-m** | **--mask** *netmask*] [**-p** | **--prefix** *hostname\_prefix*] [**-i** | **--interface** *nic*]

**mkvlan** [**-h** | **--help**]

**mkvlan** [**-v** | **--version**]

## DESCRIPTION

The **mkvlan** command takes a list of nodes and move them to a private vlan.

This command will configure the switch to create a new tagged vlan on the given nic. The primary nic will be used if the nic is not specified. The new vlan ID is given by the command. However, if it is omitted, xCAT will automatically generate the new vlan ID by querying all the switches involved and finding out the smallest common number that is not used by any existing vlans. The subnet and the netmask for the vlan will be derived from the value of “vlannets” and “vlanmasks” from the *site* table if -t and -m are not specified. The following are the default site table entires:

```
vlannets="|(\d+)|10.($1+0).0.0|";
vlanmask="255.255.0.0";
```

The vlan network will be entered in the *networks* table. The nodes will be added to the vlan using the vlan tagging technique. And the new IP addresses and new hostnames will be assigned to the nodes. The -p flag specifies the node hostname prefix for the nodes. If it is not specified, by default, the hostnames for the nodes are having the following format:

v<vlanid>nY where Y is the node number. For example, the hostname for node 5 on vlan 10 is v10n5.

The *switch.vlan* will be updated with the new vlan id for the node for standalone nodes. For KVM guests, the *vm.nics* identifies which vlan this node belongs to. For example: v13 means this node is in vlan 3.

If there are more than one switches involved in the vlan, the ports that connect to the switches need to entered in *switches.linkports* with the following format:

```
<port numner>:switch,<port number>:switch....
```

For example:

```
"42:switch1,43:switch2"
```

This command will automatically configure the cross-over ports if the given nodes are on different switches.

For added security, the root guard and bpd guard will be enabled for the ports in this vlan. However, the guards will not be disabled if the ports are removed from the vlan using *chvlan* or *rmvlan* commands. To disable them, you need

to use the switch command line interface. Refer to the switch command line interface manual to see how to disable the root guard and bpdu guard for a port.

## PARAMETERS

*vlanid* is a unique vlan number. If it is omitted, xCAT will automatically generate the new vlan ID by querying all the switches involved and finding out the smallest common number that is not used by any existing vlans. Use **lsvlan** to find out the existing vlan ids used by xCAT.

## OPTIONS

**-n|--nodes** The nodes or groups to be included in the vlan. It can be stand alone nodes or KVM guests. It takes the noderange format. Check the man page for noderange for details.

**-t|--net** The subnet for the vlan.

**-m|--mask** The netmask for the vlan

**-p|--prefix** The prefix the the new hostnames for the nodes in the vlan.

**-i|--interface** The interface name where the vlan will be tagged on. If omitted, the xCAT management network will be assumed. For FVM, this is the interface name on the host.

**-h|--help** Display usage message.

**-v|--version** The Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

To start, the xCAT switches and switches table needs to be filled with switch and port info for the nodes. For example, the swith table will look like this:

```
#node,switch,port,vlan,interface,comments,disable "node1","switch1","10",,, "node1","switch2","1","eth1",,
"node2","switch1","11","primary",, "node2","switch2","2","eth1",, "node3","switch1","12","primary:eth0",,
"node3","switch2","3","eth1",,
```

Note that the interface value for the management (primary) network can be empty, the word "primary" or "primary:ethx". For other networks, the interface attribute must be specified.

The following is an example of the switches table

```
#switch,snmpversion,username,password,privacy,auth,linkports,sshusername,sshpasword,switchtype,comments,disable
"switch1","3","username","passw0rd","sha","48:switch2",,,, "switch2","2",,,, "43:switch1",,,,
```

1. To make a private vlan for node1, node2 and node3

```
mkvlan -n node1,node2,node3
```

The vlan will be created on eth0 for the nodes.

2. To make a private vlan for node1, node2 and node3 on eth1,

```
mkvlan -n node1,node2,node3 -i eth1
```

3. To make a private vlan for node1, node2 with given subnet and netmask.

```
mkvlan -n node1,node2,node3 -t 10.3.2.0 -m 255.255.255.0
```

4. To make a private vlan for KVM guests node1 and node2

```
chtab key=usexhnm site.vlaue=1

mkdef node1 arch=x86_64 groups=kvm,all installnic=mac primarynic=mac mgt=kvm
↪netboot=pxe nfsserver=10.1.0.204 os=rhels6 profile=compute provmethod=install
↪serialport=0 serialspeed=115200 vmcpus=1 vmhost=x3650n01 vmmemory=512 vmnics=br0
↪vmstorage=nfs://10.1.0.203/vms

mkdef node2 arch=x86_64 groups=kvm,all installnic=mac primarynic=mac mgt=kvm
↪netboot=pxe nfsserver=10.1.0.204 os=rhels6 profile=compute provmethod=install
↪serialport=0 serialspeed=115200 vmcpus=1 vmhost=x3650n01 vmmemory=512 vmnics=br0
↪vmstorage=nfs://10.1.0.203/vms

mkvlan -n node1,node2

mkvm node1,node2 -s 20G

rpower node1,node2 on

rinstall node1,node2
```

## FILES

/opt/xcat/bin/mkvlan

## SEE ALSO

chvlan(1)|chvlan.1, rmvlan(1)|rmvlan.1, lsvlan(1)|lsvlan.1

## mkvm.1

## NAME

**mkvm** - Creates HMC-, DFM-, IVM-, KVM-, VMware-, and zVM-managed partitions or virtual machines.



## SYNOPSIS

### Common:

```
mkvm [-h] --help]
```

```
mkvm [-v] --version]
```

### For PPC (with HMC) specific:

```
mkvm [-V] --verbose] noderange -i id -l singlenode
```

```
mkvm [-V] --verbose] noderange -c destcec -p profile
```

```
mkvm [-V] --verbose] noderange --full
```

### For PPC (using Direct FSP Management) specific:

```
mkvm noderange [--full]
```

```
mkvm noderange [vmcpus= min/req/max] [vmmemory= min/req/max] [vmphyslots= drc_index1,drc_index2...]  
[vmothersetting= hugepage:N,bsr:N] [vmnics= vlan1[,vlan2..]] [vmstorage= N|viosnode:slotid] [--vios]
```

### For KVM:

```
mkvm noderange [-s|--size disksize] [--mem memsize] [--cpus cpucount] [-f|--force]
```

### For VMware:

```
mkvm noderange [-s | --size disksize] [--mem memsize] [--cpus cpucount]
```

### For zVM:

```
mkvm noderange [directory_entry_file_path]
```

```
mkvm noderange [source_virtual_machine] [pool= disk_pool]
```

## DESCRIPTION

### For PPC (with HMC) specific:

The first form of **mkvm** command creates new partition(s) with the same profile/resources as the partition specified by *singlenode*. The **-i** and *noderange* specify the starting numeric partition number and the *noderange* for the newly created partitions, respectively. The LHEA port numbers and the HCA index numbers will be automatically increased if they are defined in the source partition.

The second form of this command duplicates all the partitions from the source specified by *profile* to the destination specified by *destcec*. The source and destination CECs can be managed by different HMCs.

Make sure the nodes in the *noderange* is defined in the *nodelist* table and the *mgt* is set to 'hmc' in the *nodehm* table before running this command.

Note that the **mkvm** command currently only supports creating standard LPARs, not virtual LPARs working with VIOS server.

### For PPC (using Direct FSP Management) specific:

With option **--full**, a partition using all the resources on a normal power machine will be created.

If no option is specified, a partition using the parameters specified with attributes such as **vmcpus**, **vmmemory**, **vmphyslots**, **vmothersetting**, **vmnics**, **vmstorage** will be created. Those attributes can either be specified with ‘\*def’ commands running before or be specified with this command.

### For KVM and VMware:

The **mkvm** command creates a new virtual machine with *disksize* GB of storage space, *memsize* MB of memory, and *cpucount* cpu(s).

### For zVM:

The first form of **mkvm** creates a new virtual machine based on a directory entry.

The second form of this creates a new virtual machine with the same profile/resources as the specified node (cloning).

## OPTIONS

### **-h|--help**

Display usage message.

### **-c**

The cec (fsp) name for the destination.

### **--cpus**

Number of CPUs for the kvm/vmware virtual machine being created.

### **--full**

Request to create a new full system partition for each CEC.

**vmcpus= value vmmemory= value vmphyslots= value vmothersetting= value vmnics= value vmstorage= value [--vios]**

To specify the parameters which are used to create a partition. The **vmcpus**, **vmmemory** are necessary, and the value specified with this command have a more high priority. If the value of any of the three options is not specified, the corresponding value specified for the node object will be used. If any of the three attributes is neither specified with this command nor specified with the node object, error information will be returned. To reference to `lsvm(1)|lsvm.1` for more information about ‘drc\_index’ for *vmphyslots*.

The option **vios** is used to specify the partition that will be created is a VIOS partition. If specified, the value for **vmstorage** shall be number which indicate the number of vSCSI server adapter will be created, and if no value specified for **vmphyslots**, all the physical slot of the power machine will be assigned to VIOS partition. If not specified, it shall be in form of **vios\_name:server\_slotid** to specify the vios and the virtual slot id of the vSCSI server adapter that will be connected from the Logical partition.

### **-f|--force**

If the storage already exists, remove it before creating a new virtual machine.

**-i**

Starting numeric id of the newly created partitions.

**-l**

The partition name of the source.

**--mem**

Set the memory size for kvm/vmware virtual machines, default unit is MB. Specify in MB or append K for KB, M for MB, or G for GB.

**-p**

The file that contains the profiles for the source partitions.

**-s|--size**

Set the storage size for kvm/vmware virtual machines, default unit is GB. Specify in GB or append K for KB, M for MB, G for GB.

**-v|--version**

Command Version.

**-V|--verbose**

Verbose output.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To create a new HMC-managed partition lpar5 based on the profile/resources of lpar4, enter:

```
mkdef -t node -o lpar5 mgt=hmc groups=all
```

then:

```
mkvm lpar5 -i 5 -l lpar4
```

Output is similar to:

```
lpar5: Success
```

2. To create new HMC-managed partitions lpar5-lpar8 based on the profile/resources of lpar4, enter:

```
mkdef -t node -o lpar5-lpar8 mgt=hmc groups=all
```

then:

```
mkvm lpar5-lpar8 -i 5 -l lpar4
```

Output is similar to:

```
lpar5: Success
lpar6: Success
lpar7: Success
lpar8: Success
```

3. To duplicate all the HMC-managed partitions associated with cec01 on cec02, first save the lpars from cec01 to a file:

```
lsvm lpar01-lpar04 > /tmp/myprofile
```

then create lpars on cec02:

```
mkvm lpar05-lpar08 -c cec02 -p /tmp/myprofile
```

Output is similar to:

```
lpar5: Success
lpar6: Success
lpar7: Success
lpar8: Success
```

4. To duplicate all the HMC-managed partitions associated with cec01 on cec02, one is for cec01, the other is for cec02:

```
mkdef -t node -o lpar5,lpar6 mgt=hmc groups=all
chtab node=lpar5 ppc.parent=cec01
chtab node=lpar6 ppc.parent=cec02
```

then create lpars on cec01 and cec02:

```
mkvm lpar5,lpar6 --full
```

Output is similar to:

```
lpar5: Success
lpar6: Success
```

5. To create a new zVM virtual machine (gpok3) based on a directory entry:

```
mkvm gpok3 /tmp/dirEntry.txt
```

Output is similar to:

```
gpok3: Creating user directory entry for LNX3... Done
```

6. To clone a new zVM virtual machine with the same profile/resources as the specified node:

```
mkvm gpok4 gpok3 pool=POOL1
```

Output is similar to:

```
gpok4: Cloning gpok3
gpok4: Linking source disk (0100) as (1100)
```

(continues on next page)

(continued from previous page)

```
gpok4: Linking source disk (0101) as (1101)
gpok4: Stopping LNX3... Done
gpok4: Creating user directory entry
gpok4: Granting VSwitch (VSW1) access for gpok3
gpok4: Granting VSwitch (VSW2) access for gpok3
gpok4: Adding minidisk (0100)
gpok4: Adding minidisk (0101)
gpok4: Disks added (2). Disks in user entry (2)
gpok4: Linking target disk (0100) as (2100)
gpok4: Copying source disk (1100) to target disk (2100) using FLASHCOPY
gpok4: Mounting /dev/dasdg1 to /mnt/LNX3
gpok4: Setting network configuration
gpok4: Linking target disk (0101) as (2101)
gpok4: Copying source disk (1101) to target disk (2101) using FLASHCOPY
gpok4: Powering on
gpok4: Detaching source disk (0101) at (1101)
gpok4: Detaching source disk (0100) at (1100)
gpok4: Starting LNX3... Done
```

7. To create a new kvm/vmware virtual machine with 20 GB of storage, 4096 MB of memory, and 2 cpus.

```
mkvm vm1 -s 20 --mem 4096 --cpus 2
```

or

```
mkvm vm1 -s 20G --mem 4194304K --cpus 2
```

or

```
mkvm vm1 -s 20480M --mem 4096M --cpus 2
```

or

```
mkvm vm1 -s 20971520K --mem 4G --cpus 2
```

8. To create a full partition on normal power machine.

First, define a node object:

```
mkdef -t node -o lpar1 mgt=fsp cons=fsp nodetype=ppc,osi id=1 hcp=cec parent=cec
↪ hwtype=lpar groups=lpar,all
```

Then, create the partition on the specified cec.

```
mkvm lpar1 --full
```

The output is similar to:

```
lpar1: Done
```

To query the resources allocated to node 'lpar1'

```
lsvm lpar1
```

The output is similar to:

```
lpar1: Lpar Processor Info:
Curr Processor Min: 1.
Curr Processor Req: 16.
Curr Processor Max: 16.
lpar1: Lpar Memory Info:
Curr Memory Min: 0.25 GB(1 regions).
Curr Memory Req: 30.75 GB(123 regions).
Curr Memory Max: 32.00 GB(128 regions).
lpar1: 1,519,U78AA.001.WZSGVU7-P1-C7,0x21010207,0xffff(Empty Slot)
lpar1: 1,518,U78AA.001.WZSGVU7-P1-C6,0x21010206,0xffff(Empty Slot)
lpar1: 1,517,U78AA.001.WZSGVU7-P1-C5,0x21010205,0xffff(Empty Slot)
lpar1: 1,516,U78AA.001.WZSGVU7-P1-C4,0x21010204,0xffff(Empty Slot)
lpar1: 1,514,U78AA.001.WZSGVU7-P1-C19,0x21010202,0xffff(Empty Slot)
lpar1: 1,513,U78AA.001.WZSGVU7-P1-T7,0x21010201,0xc03(USB Controller)
lpar1: 1,512,U78AA.001.WZSGVU7-P1-T9,0x21010200,0x104(RAID Controller)
lpar1: 1/2/2
lpar1: 256.
```

Note: The ‘parent’ attribute for node ‘lpar1’ is the object name of physical power machine that the full partition will be created on.

9. To create a partition using some of the resources on normal power machine.

Option 1:

After a node object is defined, the resources that will be used for the partition shall be specified like this:

```
chdef lpar1 vmcpus=1/4/16 vmmemory=1G/4G/32G vmphyslots=0x21010201,0x21010200,
↳vmothersetting=bsr:128,hugepage:2
```

Then, create the partition on the specified cec.

```
mkvm lpar1
```

Option 2:

```
mkvm lpar1 vmcpus=1/4/16 vmmemory=1G/4G/32G vmphyslots=0x21010201,0x21010200,
↳vmothersetting=bsr:128,hugepage:2
```

The output is similar to:

```
lpar1: Done
```

Note: The ‘vmphyslots’ specify the drc index of the physical slot device. Every drc index shall be delimited with ‘,’. The ‘vmothersetting’ specify two kinds of resource, bsr(Barrier Synchronization Register) specified the num of BSR arrays, hugepage(Huge Page Memory) specified the num of huge pages.

To query the resources allocated to node ‘lpar1’

```
lsvm lpar1
```

The output is similar to:

```
lpar1: Lpar Processor Info:
Curr Processor Min: 1.
Curr Processor Req: 4.
```

(continues on next page)

(continued from previous page)

```

Curr Processor Max: 16.
lpar1: Lpar Memory Info:
Curr Memory Min: 1.00 GB(4 regions).
Curr Memory Req: 4.00 GB(16 regions).
Curr Memory Max: 32.00 GB(128 regions).
lpar1: 1,513,U78AA.001.WZSGVU7-P1-T7,0x21010201,0xc03(USB Controller)
lpar1: 1,512,U78AA.001.WZSGVU7-P1-T9,0x21010200,0x104(RAID Controller)
lpar1: 1/2/2
lpar1: 128.

```

10. To create a vios partition using some of the resources on normal power machine.

```

mkvm viosnode vmcpus=1/4/16 vmmemory=1G/4G/32G vmphyslots=0x21010201,0x21010200,
↪vmnics=vlan1 vmstorage=5 --vios

```

The resources for the node is similar to:

```

viosnode: Lpar Processor Info:
Curr Processor Min: 1.
Curr Processor Req: 4.
Curr Processor Max: 16.
viosnode: Lpar Memory Info:
Curr Memory Min: 1.00 GB(4 regions).
Curr Memory Req: 4.00 GB(16 regions).
Curr Memory Max: 32.00 GB(128 regions).
viosnode: 1,513,U78AA.001.WZSGVU7-P1-T7,0x21010201,0xc03(USB Controller)
viosnode: 1,512,U78AA.001.WZSGVU7-P1-T9,0x21010200,0x104(RAID Controller)
viosnode: 1,0,U8205.E6B.0612BAR-V1-C,0x30000000,vSerial Server
viosnode: 1,1,U8205.E6B.0612BAR-V1-C1,0x30000001,vSerial Server
viosnode: 1,3,U8205.E6B.0612BAR-V1-C3,0x30000003,vEth (port_vlanid=1,mac_
↪addr=4211509276a7)
viosnode: 1,5,U8205.E6B.0612BAR-V1-C5,0x30000005,vSCSI Server
viosnode: 1,6,U8205.E6B.0612BAR-V1-C6,0x30000006,vSCSI Server
viosnode: 1,7,U8205.E6B.0612BAR-V1-C7,0x30000007,vSCSI Server
viosnode: 1,8,U8205.E6B.0612BAR-V1-C8,0x30000008,vSCSI Server
viosnode: 1,9,U8205.E6B.0612BAR-V1-C9,0x30000009,vSCSI Server
viosnode: 0/0/0
viosnode: 0.

```

## FILES

/opt/xcat/bin/mkvm

## SEE ALSO

chvm(1)|chvm.1, lsvm(1)|lsvm.1, rmvm(1)|rmvm.1

## mkzone.1

## NAME

**mkzone** - Defines a new zone in the cluster.

## SYNOPSIS

**mkzone** *zonename* [--defaultzone] [-k *full path to the ssh RSA private key*] [-a *noderange*] [-g] [-f] [-s {yes|no}] [-V]

**mkzone** [-h | -v]

## DESCRIPTION

The **mkzone** command is designed to divide the xCAT cluster into multiple zones. The nodes in each zone will share common root ssh keys. This allows the nodes in a zone to be able to as root ssh to each other without password, but cannot do the same to any node in another zone. All zones share a common xCAT Management Node and database including the site table, which defines the attributes of the entire cluster. The mkzone command is only supported on Linux ( No AIX support). The nodes are not updated with the new root ssh keys by mkzone. You must run updatenode -k or xdsh -K to the nodes to update the root ssh keys to the new generated zone keys. This will also sync any service nodes with the zone keys, if you have a hierarchical cluster. Note: if any zones in the zone table, there must be one and only one defaultzone. Otherwise, errors will occur.

## OPTIONS

**-h | --help**

Displays usage information.

**-v | --version**

Displays command version and build date.

**-k | --sshkeypath** *full path to the ssh RSA private key*

This is the path to the id\_rsa key that will be used to build root's ssh keys for the zone. If -k is used, it will generate the ssh public key from the input ssh RSA private key and store both in /etc/xcat/sshkeys/<zonename>/.ssh directory. If -f is not used, then it will generate a set of root ssh keys for the zone and store them in /etc/xcat/sshkeys/<zonename>/.ssh.

**--default**



if `--defaultzone` is input, then it will set the zone `defaultzone` attribute to yes; otherwise it will set to no. if `--defaultzone` is input and another zone is currently the default, then the `-f` flag must be used to force a change to the new defaultzone. If `-f` flag is not use an error will be returned and no change made. Note: if any zones in the zone table, there must be one and only one defaultzone. Otherwise, errors will occur.

#### **-a | --addnoderange** *noderange*

For each node in the noderange, it will set the `zonename` attribute for that node to the input `zonename`. If the `-g` flag is also on the command, then it will add the group name “`zonename`” to each node in the noderange.

#### **-s | --sshbetweennodes** *yes|no*

If `-s` entered, the zone `sshbetweennodes` attribute will be set to yes or no. It defaults to yes. When this is set to yes, then ssh will be setup to allow passwordless root access between nodes. If no, then root will be prompted for a password when running ssh between the nodes in the zone.

#### **-f | --force**

Used with the (`--defaultzone`) flag to override the current default zone.

#### **-g | --assigngroup**

Used with the (`-a`) flag to create the group `zonename` for all nodes in the input noderange.

#### **-V | --verbose**

Verbose mode.

## EXAMPLES

1. To make a new zone1 using defaults, enter:

```
mkzone zone1
```

Note: with the first **mkzone**, you will automatically get the `xcatdefault` zone created as the default zone. This zone uses ssh keys from `<roothome>/ssh` directory.

2. To make a new zone2 using defaults and make it the default zone enter:

```
mkzone> zone2 --defaultzone -f
```

3. To make a new zone2A using the ssh `id_rsa` private key in `/root/.ssh`:

```
mkzone zone2A -k /root/.ssh
```

4. To make a new zone3 and assign the noderange `compute3` to the zone enter:

```
mkzone zone3 -a compute3
```

5. To make a new zone4 and assign the noderange `compute4` to the zone and add zone4 as a group to each node enter:

```
mkzone zone4 -a compute4 -g
```

6. To make a new zone5 and assign the noderange `compute5` to the zone and add zone5 as a group to each node but not allow passwordless ssh between the nodes enter:

```
mkzone zone5 -a compute5 -g -s no
```

## Files

/opt/xcat/bin/mkzone/

Location of the mkzone command.

## SEE ALSO

chzone(1)|chzone.1, rmzone(1)|rmzone.1, xdsh(1)|xdsh.1, updatenode(1)|updatenode.1

## monadd.1

### NAME

**monadd** - Registers a monitoring plug-in to the xCAT cluster.

### SYNOPSIS

**monadd** [-h] --help]

**monadd** [-v] --version]

**monadd** name [-n|--nodestatmon] [-s|--settings *settings*]

### DESCRIPTION

This command is used to register a monitoring plug-in module to monitor the xCAT cluster. The plug-in module will be added to the xCAT *monitoring* database table and the configuration scripts for the monitoring plug-in, if any, will be added to the *postscripts* table. A monitoring plug-in module acts as a bridge that connects a 3rd party monitoring software and the xCAT cluster. A configuration script is used to configure the 3rd party software. Once added to the <postscripts> table, it will be invoked on the nodes during node deployment stage.

### Parameters

*name* is the name of the monitoring plug-in module. For example, if the *name* is called *xxx*, then the actual file name that the xcatd looks for is */opt/xcat/lib/perl/xCAT\_monitoring/xxx.pm*. Use *monls -a* command to list all the monitoring plug-in modules that can be used.

*settings* is the monitoring plug-in specific settings. It is used to customize the behavior of the plug-in or configure the 3rd party software. Format: *-s key-value -s key=value ...*. Note that the square brackets are needed here. Use *monls name -d* command to look for the possible setting keys for a plug-in module.

## OPTIONS

### **-h | --help**

Display usage message.

### **-n | --nodestatmon**

Indicate that this monitoring plug-in will be used for feeding the node liveness status to the xCAT *nodelist* table.

### **-s | --settings**

Specifies the plug-in specific settings. These settings will be used by the plug-in to customize certain entities for the plug-in or the third party monitoring software. e.g. `-s mon_interval=10 -s toggle=1`.

### **-v | --version**

Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To register gangliamon plug-in module (which interacts with Ganglia monitoring software) to monitor the xCAT cluster, enter:

```
monadd gangliamon
```

2. To register rmcmon plug-in module (which interacts with IBM's RSCT monitoring software) to monitor the xCAT cluster and have it feed the node liveness status to xCAT's *nodelist* table, enter:

```
monadd rmcmon -n
```

This will also add the *configrmcnode* to the *postscripts* table. To view the content of the *postscripts* table, enter:

```
tabdump postscripts
#node,postscripts,comments,disable
"service","servicenode",,
"xcatdefaults","syslog,remoteshell,configrmcnode",,
```

3. To register xcatmon plug-in module to feed the node liveness status to xCAT's *nodelist* table, enter:

```
monadd xcatmon -n -s ping-interval=2
```

where 2 is the number of minutes between the pings.

## FILES

/opt/xcat/bin/monadd

## SEE ALSO

monls(1)|monls.1, monrm(1)|monrm.1, monstart(1)|monstart.1, monstop(1)|monstop.1, moncfg(1)|moncfg.1, mon-decfg(1)|mondecfg.1

## moncfg.1

## NAME

**moncfg** - Configures a 3rd party monitoring software to monitor the xCAT cluster.

## SYNOPSIS

**moncfg** [-h] --help]

**moncfg** [-v] --version]

**moncfg** *name* [*noderange*] [-r|--remote]

## DESCRIPTION

This command is used to configure a 3rd party monitoring software to monitor the xCAT cluster. For example, it modifies the configuration file for the monitoring software so that the nodes can be included in the monitoring domain. The operation is performed on the management node and the service nodes of the given nodes. The operation will also be performed on the nodes if the *-r* option is specified, though the configuration of the nodes is usually performed during the node deployment stage.

## Parameters

*name* is the name of the monitoring plug-in module. For example, if the *name* is called *xxx*, then the actual file name that the xcatd looks for is */opt/xcat/lib/perl/xCAT\_monitoring/xxx.pm*. Use *monls -a* command to list all the monitoring plug-in modules that can be used.

*noderange* specifies the nodes to be monitored. If omitted, all nodes will be monitored.

## OPTIONS

**-h | --help** Display usage message.

**-r | --remote** Specifies that the operation will also be performed on the nodes.

**-v | --version** Command Version.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To configure the management node and the service nodes for ganglia monitoring, enter:

```
moncfg gangliamon
```

2. To configure the management node, nodes and their service nodes for ganglia monitoring, enter:

```
moncfg gangliamon -r
```

## FILES

/opt/xcat/bin/moncfg

## SEE ALSO

monls(1)|monls.1, mondecfg(1)|mondecfg.1, monadd(1)|monadd.1, monrm(1)|monrm.1, monstart(1)|monstart.1, monstop(1)|monstop.1

## mondecfg.1

## NAME

**mondecfg** - Deconfigures a 3rd party monitoring software from monitoring the xCAT cluster.

## SYNOPSIS

**mondecfg** [-h] --help]

**mondecfg** [-v] --version]

**mondecfg** *name* [*noderange*] [-r|--remote]

## DESCRIPTION

This command is used to deconfigure a 3rd party monitoring software from monitoring the xCAT cluster. The operation is performed on the management node and the service nodes of the given nodes. The operation will also be performed on the nodes if the *-r* option is specified. The deconfiguration operation will remove the nodes from the 3rd party software's monitoring domain.

## PARAMETERS

*name* is the name of the monitoring plug-in module. Use *monls* command to list all the monitoring plug-in modules that can be used.

*nodrange* specified the nodes to be deconfigured. If omitted, all nodes will be deconfigured.

## OPTIONS

**-h | --help** Display usage message.

**-r | --remote** Specifies that the operation will also be performed on the nodes.

**-v | --version** Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To deconfigure the management node and the service nodes from the ganglia monitoring, enter:

```
mondecfg gangliamon
```

2. To deconfigure the management node, nodes and their service nodes from the ganglia monitoring, enter:

```
mondecfg gangliamon -r
```

## FILES

/opt/xcat/bin/mondecfg

## SEE ALSO

monls(1)|monls.1, moncfg(1)|moncfg.1, monadd(1)|monadd.1, monrm(1)|monrm.1, monstart(1)|monstart.1, monstop(1)|monstop.1

### monls.1

## NAME

**monls** - Lists monitoring plug-in modules that can be used to monitor the xCAT cluster.

## SYNOPSIS

**monls** [-h] --help]

**monls** [-v] --version]

**monls** *name* [-d|--description]

**monls** [-a|--all] [-d|--description]

## DESCRIPTION

This command is used to list the status, description, the configuration scripts and the settings of one or all of the monitoring plug-in modules.

### Parameters

*name* is the name of the monitoring plug-in module.

## OPTIONS

**-a | --all** Searches the *XCATROOT/lib/perl/xCAT\_monitoring* directory and reports all the monitoring plug-in modules. If nothing is specified, the list is read from the *monitoring* table.

**-d | --description** Display the description of the plug-in modules. The description usually contains the possible settings.

**-h | --help** Display usage message.

**-v | --version** Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To list the status of all the monitoring plug-in modules from the *monitoring* table, enter:

```
monls
```

The output looks like this:

xcatmon	monitored	node-status-monitored
snmpmon	<b>not</b> -monitored	

2. To list the status of all the monitoring plug-in modules including the ones that are not in the monitoring table, enter

```
monls -a
```

The output looks like this:

xcatmon	monitored	node-status-monitored
snmpmon	not-monitored	
gangliamon	not-monitored	
rmcmmon	monitored	
nagiosmon	not-monitored	

3. To list the status and the description for *snmpmon* module, enter:

```
monls snmpmon -d
```

## FILES

/opt/xcat/bin/monls

## SEE ALSO

monadd(1)|monadd.1, monrm(1)|monrm.1, monstart(1)|monstart.1, monstop(1)|monstop.1, moncfg(1)|moncfg.1, mondecfg(1)|mondecfg.1

## monrm.1

## NAME

**monrm** - Unregisters a monitoring plug-in module from the xCAT cluster.

## SYNOPSIS

**monrm** [-h] --help]

**monrm** [-v] --version]

**monrm** *name*

## DESCRIPTION

This command is used to unregister a monitoring plug-in module from the *monitoring* table. It also removes any configuration scripts associated with the monitoring plug-in from the *postscripts* table. A monitoring plug-in module acts as a bridge that connects a 3rd party monitoring software and the xCAT cluster. A configuration script is used to configure the 3rd party software. Once added to the *postscripts* table, it will be invoked on the nodes during node deployment stage.



## PARAMETERS

*name* is the name of the monitoring plug-in module in the *monitoring* table. Use *monls* command to list all the monitoring plug-in modules that can be used.

## OPTIONS

**-h | --help** Display usage message.

**-v | --version** Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1.To unregister gangliamon plug-in module (which interacts with Ganglia monitoring software) from the xCAT cluster, enter:

```
monrm gangliamon
```

Note that gangliamon must have been registered in the xCAT *monitoring* table. For a list of registered plug-in modules, use command **monls**.

## FILES

/opt/xcat/bin/monrm

## SEE ALSO

monls(1)|monls.1, monadd(1)|monadd.1, monstart(1)|monstart.1, monstop(1)|monstop.1, moncfg(1)|moncfg.1, mondecfg(1)|mondecfg.1

## monshow.1

## NAME

**monshow** - Shows event data for monitoring.

## SYNOPSIS

**monshow** [-h] --help]

**monshow** [-v] --version]

**monshow** *name* [*noderange*] [-s] [-t *time*] [-a *attributes*] [-w *attr* < *operator* > *val*] [-w *attr* < *operator* > *val*] ... ][-o {p|e}]

## DESCRIPTION

This command displays the events that happened on the given nodes or the monitoring data that is collected from the given nodes for a monitoring plugin.

## PARAMETERS

*name* is the name of the monitoring plug-in module to be invoked.

*noderange* is a list of nodes to be showed for. If omitted, the data for all the nodes will be displayed.

## OPTIONS

**-h | --help** Display usage message.

**-v | --version** Command Version.

**-s** shows the summary data.

**-t** specifies a range of time for the data, The default is last 60 minutes. For example -t 6-4, it will display the data from last 6 minutes to 4 minutes; If it is -t 6, it will display the data from last 6 minutes until now.

**-a** specifies a comma-separated list of attributes or metrics names. The default is all.

**-w** specify one or multiple selection string that can be used to select events. The operators ==, !=, =, !, >, <, >=, <= are available. Wildcards % and \_ are supported in the pattern string. % allows you to match any string of any length(including zero length) and \_ allows you to match on a single character. The valid attributes are eventtype, monitor, monnode, application, component, id, severity, message, rawdata, comments. Valid severity are: Informational, Warning, Critical.

Operator descriptions:

**==** Select event where the attribute value is exactly this value.

**!=** Select event where the attribute value is not this specific value.

**=~** Select event where the attribute value matches this pattern string. Not work with severity.

**!~>** Select event where the attribute value does not match this pattern string. Not work with severity.

**>** Select event where the severity is higher than this value. Only work with severity.

**<** Select event where the severity is lower than this value. Only work with severity.

**>=** Select event where the severity is higher than this value(include). Only work with severity.

**<=** Select event where the severity is lower than this value(include). Only work with severity.

Note: if the “val” or “operator” fields includes spaces or any other characters that will be parsed by shell, the “attr<operator>val” needs to be quoted. If the operator is “!~”, the “attr<operator>val” needs to be quoted using single quote.

**-o** specifies montype, it can be p or e. p means performance, e means events.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To show summary data about PctRealMemFree and PctTotalTimeIdle of cluster in last 6 minutes, enter:

```
monshow rmcmon -s -a PctRealMemFree,PctTotalTimeIdle -t 6
```

2. To show all data of node1 and node2, enter:

```
monshow rmcmon node1,node2
```

3. To show summary data of nodes which managed by servicenode1, enter:

```
monshow rmcmon servicenode1 -s
```

4. To show RMC event with severity Critical, enter:

```
monshow rmcmon -w severity==Critical
```

## FILES

/opt/xcat/bin/monshow

## SEE ALSO

monls(1)|monls.1, monstart(1)|monstart.1, monstop(1)|monstop.1, monadd(1)|monadd.1, monrm(1)|monrm.1, mon-cfg(1)|moncfg.1, mondecfg(1)|mondecfg.1

## monstart.1

## NAME

**monstart** - Starts a plug-in module to monitor the xCAT cluster.

## SYNOPSIS

**monstart** [-h] --help]

**monstart** [-v] --version]

**monstart** *name* [*nodestring*] [-r|--remote]

## DESCRIPTION

This command is used to start a 3rd party software, (for example start the daemons), to monitor the xCAT cluster. The operation is performed on the management node and the service nodes of the given nodes. The operation will also be performed on the nodes if the **-r** option is specified.

## PARAMETERS

*name* is the name of the monitoring plug-in module. For example, if the *name* is called *xxx*, then the actual file name that the xcatd looks for is */opt/xcata/lib/perl/xcata\_monitoring/xxx.pm*. Use **monls -a** command to list all the monitoring plug-in modules that can be used.

*nodestring* is the nodes to be monitored. If omitted, all nodes will be monitored.

## OPTIONS

**-h | --help** Display usage message.

**-r | --remote** Specifies that the operation will also be performed on the nodes. For example, the 3rd party monitoring software daemons on the nodes will also be started.

**-v | --version** Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To start gangliamon plug-in module (which interacts with Ganglia monitoring software) to monitor the xCAT cluster, enter:

```
monstart gangliamon -r
```

2. To start xcatmon plug-in module to feed the node liveness status to xCAT's *nodelist* table, enter:

```
monstart rmcmon
```

## FILES

/opt/xcat/bin/monstart

## SEE ALSO

monls(1)|monls.1, monstop(1)|monstop.1, monadd(1)|monadd.1, monrm(1)|monrm.1, moncfg(1)|moncfg.1, mon-decfg(1)|mondecfg.1

## monstop.1

## NAME

**monstop** - Stops a monitoring plug-in module to monitor the xCAT cluster.

## SYNOPSIS

**monstop** [-h] --help]

**monstop** [-v] --version]

**monstop** *name* [*noderange*] [-r|--remote]

## DESCRIPTION

This command is used to stop a 3rd party software, (for example stop the daemons), from monitoring the xCAT cluster. The operation is performed on the management node and the service nodes of the given nodes. The operation will also be performed on the nodes if the **-r** option is specified.

## PARAMETERS

*name* is the name of the monitoring plug-in module in the *monitoring* table. Use **monls** command to list all the monitoring plug-in modules that can be used.

*noderange* is the nodes to be stopped for monitoring. If omitted, all nodes will be stopped.

## OPTIONS

**-h | -help** Display usage message.

**-r | --remote** Specifies that the operation will also be performed on the nodes. For example, the 3rd party monitoring software daemons on the nodes will also be stopped.

**-v | -version** Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1.To stop gangliamon plug-in module (which interacts with Ganglia monitoring software) to monitor the xCAT cluster, enter:

```
monstop gangliamon
```

Note that gangliamon must have been registered in the xCAT *monitoring* table. For a list of registered plug-in modules, use command *monls*.

## FILES

/opt/xcat/bin/monstop

## SEE ALSO

monls(1)|monls.1, monstart(1)|monstart.1, monadd(1)|monadd.1, monrm(1)|monrm.1, moncfg(1)|moncfg.1, mon-decfg(1)|mondecfg.1

## mysqlsetup.1

### NAME

**mysqlsetup** - Sets up the MySQL or MariaDB database for xCAT to use.

### SYNOPSIS

**mysqlsetup** { -h | --help }

**mysqlsetup** { -v | --version }

**mysqlsetup** { -i | --init } [-f | --hostfile] [-o | --odbc] [-L | --LL] [-V | --verbose]

**mysqlsetup** { -u | --update } [-f | --hostfile] [-o | --odbc] [-L | --LL] [-V | --verbose]

**mysqlsetup** { -o | --odbc } [-V | --verbose]

**mysqlsetup** { -L | --LL } [-V | --verbose]

## DESCRIPTION

**mysqlsetup** - Sets up the MySQL or MariaDB database (linux only for MariaDB) for xCAT to use. The **mysqlsetup** script is run on the Management Node as root after the MySQL or MariaDB packages have been installed. Before running the **--init** option, the MySQL server should be stopped, if it is running. The xCAT daemon, **xcatd**, must be running, do not stop it. No xCAT commands should be run during the init process, because we will be migrating the xCAT database to MySQL or MariaDB and restarting the **xcatd** daemon as well as the MySQL daemon. For more information, see <https://xcat-docs.readthedocs.io/en/stable/advanced/hierarchy/databases/index.html#mysql-mariadb>

Two passwords must be supplied for the setup, a password for the xcatadmin id and a password for the root id in the MySQL database. These will be prompted for interactively, unless the environment variables **XCAT-MYSQLADMIN\_PW** and **XCATMYSQLROOT\_PW** are set to the passwords for the xcatadmin id and root id in the database, respectively.

Note below we refer to MySQL but it works the same for MariaDB.

## OPTIONS

### **-h|--help**

Displays the usage message.

### **-v|--version**

Displays the release version of the code.

### **-V|--verbose**

Displays verbose messages.

### **-i|--init**

The **--init** option is used to setup a xCAT database on an installed MySQL or MariaDB server for xCAT to use. The **mysqlsetup** script will check for the installed MariaDB server rpm first and will use MariaDB if it is installed. This involves creating the xcatdb database, the xcatadmin id, allowing access to the xcatdb database by the Management Node. It customizes the **my.cnf** configuration file for xcat and starts the MySQL server. It also backs up the current xCAT database and restores it into the newly setup xcatdb MySQL database. It creates the **/etc/xcat/cfgloc** file to point the xcatd daemon to the MySQL database and restarts the xcatd daemon using the database. On AIX, it additionally setup the mysql id and group and corrects the permissions in the MySQL install directories. For AIX, you should be using the MySQL rpms available from the xCAT website. For Linux, you should use the MySQL or MariaDB rpms shipped with the OS. You can chose the **-f** and/or the **-o** option, to run after the **<\*\*\*\*\*-init>**.

### **-u|--update**

To run the update option, you must first have run the **-i** option and have xcat successfully running on the MySQL database. You can chose the **-f** and/or the **-o** option, to update.

### **-f|--hostfile**

This option runs during update, it will take all the host from the input file (provide a full path) and give them database access to the xcatdb in MySQL for the xcatadmin id. Wildcards and ipaddresses may be used. xCAT must have been previously successfully setup to use MySQL. xcatadmin and MySQL root password are required.

### **-o|--odbc**

This option sets up the ODBC **/etc/./odbcinst.ini**, **/etc/./odbc.ini** and the **.odbc.ini** file in roots home directory will be created and initialized to run off the xcatdb MySQL database.

### **-L|--LL**

Additional database configuration specifically for the LoadLeveler product.

## ENVIRONMENT VARIABLES

\* **XCATMYSQLADMIN\_PW** - the password for the xcatadmin id that will be assigned in the MySQL database.

\* **XCATMYSQLROOT\_PW** - the password for the root id that will be assigned to the MySQL root id, if the script creates it. The password to use to run MySQL command to the database as the MySQL root id. This password may be different than the unix root password on the Management Node.

## EXAMPLES

1. To setup MySQL for xCAT to run on the MySQL xcatdb database :

```
mysqlsetup -i
```

2. Add hosts from /tmp/xcat/hostlist that can access the xcatdb database in MySQL:

```
mysqlsetup -u -f /tmp/xcat/hostlist
```

Where the file contains a host per line, for example:

```
node1
1.115.85.2
10.%[.%.%
nodex.cluster.net
```

3. To setup the ODBC for MySQL xcatdb database access :

```
mysqlsetup -o
```

4. To setup MySQL for xCAT and add hosts from /tmp/xcat/hostlist and setup the ODBC in Verbose mode:

```
mysqlsetup -i -f /tmp/xcat/hostlist -o -V
```

## nimnodecust.1

### NAME

**nimnodecust** - Use this xCAT command to customize AIX/NIM standalone machines.

### SYNOPSIS

**nimnodecust** [-h|--help ]

**nimnodecust** [-V] -s *lpp\_source\_name* [-p *packages*] [-b *installp\_bundles*] *noderange* [*attr=val* [*attr=val* ...]]



## DESCRIPTION

This xCAT command can be used to customize AIX/NIM standalone machines.

The software packages that you wish to install on the nodes must be copied to the appropriate directory locations in the NIM lpp\_source resource provided by the “-s” option. For example, if the location of your lpp\_source resource is “/install/nim/lpp\_source/61lpp/” then you would copy RPM packages to “/install/nim/lpp\_source/61lpp/RPMS/ppc” and you would copy your installp packages to “/install/nim/lpp\_source/61lpp/installp/ppc”. Typically you would want to copy the packages to the same lpp\_source that was used to install the node. You can find the location for an lpp\_source with the AIX lsnim command. (Ex. “lsnim -l <lpp\_source\_name>”)

The packages you wish to install on the nodes may be specified with either a comma-separated list of package names or by a comma-separated list of installp\_bundle names. The installp\_bundle names are what were used when creating the corresponding NIM installp\_bundle definitions. The installp\_bundle definitions may also be used when installing the nodes.

A bundle file contains a list of package names. The RPMs must have a prefix of “R:” and the installp packages must have a prefix of “I:”. For example, the contents of a simple bundle file might look like the following.

```
# RPM
R:expect-5.42.1-3.aix5.1.ppc.rpm
R:ping-2.4b2_to-1.aix5.3.ppc.rpm

#installp
I:openssh.base
I:openssh.license
```

To create a NIM installp\_bundle definition you can use the “nim -o define” operation. For example, to create a definition called “mypackages” for a bundle file located at “/install/nim/mypkgs.bnd” you could issue the following command.

```
nim -o define -t installp_bundle -a server=master -a location=/install/nim/mypkgs.bnd
↪mypackages
```

See the AIX documentation for more information on using installp\_bundle files.

The xCAT nimnodecust command will automatically handle the distribution of the packages to AIX service nodes when using an xCAT hierarchical environment.

## OPTIONS

*attr=val [attr=val ...]*

Specifies one or more “attribute equals value” pairs, separated by spaces. Attr=val pairs must be specified last on the command line. These are used to specify additional values that can be passed to the underlying NIM commands, (“nim -o cust...”). See the NIM documentation for valid “nim” command line options.

**-b** *installp\_bundle\_names*

A comma separated list of NIM installp\_bundle names.

**-h** | **--help**

Display usage message.

**-p** *package\_names*

A comma-separated list of software packages to install. Packages may be RPM or installp.

*noderange*

A set of comma delimited node names and/or group names. See the “noderange” man page for details on additional supported formats.

**-V |--verbose**

Verbose mode.

**RETURN VALUE**

0 The command completed successfully.

1 An error has occurred.

**EXAMPLES**

- 1) Install the installp package “openssh.base.server” on an xCAT node named “node01”. Assume that the package has been copied to the NIM lpp\_source resource called “61lppsource”.

```
nimnodecust -s 61lppsource -p openssh.base.server node01
```

- 2) Install the product software contained in the two bundles called “llbnd” and “pebnd” on all AIX nodes contained in the xCAT node group called “aixnodes”. Assume that all the software packages have been copied to the NIM lpp\_source resource called “61lppsource”.

```
nimnodecust -s 61lppsource -b llbnd,pebnd aixnodes
```

**FILES**

/opt/xcat/bin/nimnodecust

**NOTES**

This command is part of the xCAT software product.

**nimnodeset.1****NAME**

**nimnodeset** - Use this xCAT command to initialize AIX/NIM standalone machines.

## SYNOPSIS

**nimnodeset** [-h|--help ]

**nimnodeset** [-V|--verbose] [-f|--force] [-i *osimage\_name*] [-l *location*] [-p|--primarySN] [-b | --backupSN] *nodename* [*attr=val* [*attr=val* ... ]]

## DESCRIPTION

This xCAT command can be used to initialize AIX/NIM standalone machines. Once this step is completed the either the xCAT **rnetboot** command or the **rbootseq/rpower** commands to initiate a network boot of the nodes.

If you are using xCAT service nodes the **nimnodeset** command will automatically determine the correct server(s) for the node and do the initialization on that server(s).

The *osimage\_name* is the name of an xCAT osimage definition that contains the list of NIM resources to use when initializing the nodes. If the *osimage\_name* is not provided on the command line the code checks the node definition for the value of the “provmethod” attribute (which is the name of an osimage definition). If the *osimage\_image* is provided on the command line then the code will also set the “provmethod” attribute of the node definitions.

This command will also create a NIM *resolv\_conf* resource to be used when installing the node. If a *resolv\_conf* resource is not already included in the xCAT osimage definition and if the “domain” and “nameservers” values are set then a new NIM *resolv\_conf* resource will be created and allocated to the nodes.

The “domain” and “nameservers” attributes can be set in either the xCAT “network” definition used by the nodes or in the xCAT cluster “site” definition. The setting in the “network” definition will take priority.

The “search” field of the *resolv.conf* file will contain a list all the domains listed in the xCAT network definitions and the xCAT site definition.

The “nameservers” value can either be set to a specific IP address or the “<xcatmaster>” key word. The “<xcatmaster>” key word means that the value of the “xcatmaster” attribute of the node definition will be used in the */etc/resolv.conf* file. (I.e. The name of the install server as known by the node.)

You can set the “domain” and “nameservers” attributes by using the **chdef** command. For example:

```
chdef -t network -o clstr_net domain=cluster.com nameservers=<xcatmaster>
```

If the “domain” and “nameservers” attributes are not set in either the nodes “network” definition or the “site” definition then no new NIM *resolv\_conf* resource will be created.

You can specify additional attributes and values using the “attr=val” command line option. This information will be passed on to the underlying call to the NIM “nim -o bos\_inst” command. See the NIM documentation for information on valid command line options for the nim command. The “attr” must correspond to a NIM attribute supported for the NIM “bos\_inst” operation. Information provided by the “attr=val” option will take precedence over the information provided in the osimage definition.

The force option can be used to reinitialize a node if it already has resources allocated or it is in the wrong NIM state. This option will reset the NIM node and deallocate resources before reinitializing.

This command will also create a NIM script resource to enable the xCAT support for user-provided customization scripts.

After the **nimnodeset** command completes you can use the **lsnim** command to check the NIM node definition to see if it is ready for booting the node. (“lsnim -l <nim\_node\_name>”).

You can supply your own scripts to be run on the management node or on the service node (if there is hierarchy) for a node during the **nimnodeset** command. Such scripts are called **prescripts**. They should be copied to */install/prescripts* directory. A table called *prescripts* is used to specify the scripts and their associated actions. The scripts to be run

at the beginning of the **nimnodeset** command are stored in the ‘begin’ column of *prescripts* table. The scripts to be run at the end of the **nimnodeset** command are stored in the ‘end’ column of *prescripts* table. Run ‘`tabdump prescripts -d`’ command for details. An example for the ‘begin’ or the ‘end’ column is: *standalone:myscript1,myscript2*. The following two environment variables will be passed to each script: **NODES** contains all the names of the nodes that need to run the script for and **ACTION** contains the current nodeset action, in this case “standalone”. If *#xCAT setting:MAX\_INSTANCE=number* is specified in the script, the script will get invoked for each node in parallel, but no more than *number* of instances will be invoked at a time. If it is not specified, the script will be invoked once for all the nodes.

## OPTIONS

*attr=val [attr=val ...]*

Specifies one or more “attribute equals value” pairs, separated by spaces. Attr= val pairs must be specified last on the command line. These are used to specify additional values that can be passed to the underlying NIM commands, (“nim -o bos\_inst ...”). See the NIM documentation for valid “nim” command line options. Note that you may specify multiple “script” and “installp\_bundle” values by using a comma separated list. (ex. “script=ascript,bscript”).

### **-b|--backupSN**

When using backup service nodes only update the backup. The default is to update both the primary and backup service nodes

### **-f|--force**

Use the force option to reinitialize the NIM machines.

### **-h|--help**

Display usage message.

### **-i image\_name**

The name of an existing xCAT osimage definition.

### **-l|--location**

The directory location to use when creating new NIM resolv\_conf resources. The default location is /install/nim.

### **-p|--primarySN**

When using backup service nodes only update the primary. The default is to update both the primary and backup service nodes.

### *noderange*

A set of comma delimited node names and/or group names. See the “noderange” man page for details on additional supported formats.

### **-V|--verbose**

Verbose mode.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

- 1) Initialize an xCAT node named “node01”. Use the xCAT osimage named “61gold” to install the node.

```
nimnodeset -i 61gold node01
```

- 2) Initialize all AIX nodes contained in the xCAT node group called “aixnodes” using the image definitions pointed to by the “provmethod” attribute of the xCAT node definitions.

```
nimnodeset aixnodes
```

- 3) Initialize an xCAT node called “node02”. Include installp\_bundle resources that are not included in the osimage definition. This assumes the NIM installp\_bundle resources have already been created.

```
nimnodeset -i 611image node02 installp_bundle=sshbundle,addswbundle
```

## FILES

/opt/xcat/bin/nimnodeset

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

mknimimage(1)|mknimimage.1, rnetboot(1)|rnetboot.1

## nodeaddunmged.1

## NAME

**nodeaddunmged** - Create a unmanaged node.

## SYNOPSIS

**nodeaddunmged** [-h| --help | -v | --version]

**nodeaddunmged** hostname=*node-name* ip=*ip-address*

## DESCRIPTION

The **nodeaddunmged** command adds an unmanaged node to the \_\_Unmanaged group. You can specify the node name and IP address of the node.

## OPTIONS

**-h|--help**

Display usage message.

**-v|--version**

Command Version.

**hostname**=*node-name*

Sets the name of the new unmanaged node, where <node-name> is the name of the node.

**ip**=*ip-address*

Sets the IP address of the unmanaged node, where *ip-address* is the IP address of the new node in the form xxx.xxx.xxx.xxx

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

To add an unmanaged node, use the following command:

```
nodeaddunmged hostname=unmanaged01 ip=192.168.1.100
```

## SEE ALSO

**nodech.1**

## NAME

**nodech** - Changes nodes' attributes in the xCAT cluster database.

## SYNOPSIS

**nodech** *noderange table.column=value* [...]

**nodech** {**-d** | **--delete**} *noderange table* [...]

**nodech** {**-v** | **--version**}

**nodech** [**-?** | **-h** | **--help**]

## DESCRIPTION

The **nodech** command changes the specified attributes for the given nodes. Normally, the given value will completely replace the current attribute value. But if “,” is used instead of “=”, the specified value will be prepended to the attribute’s comma separated list, if it is not already there. If “^=” is used, the specified value will be removed from the attribute’s comma separated list, if it is there. You can also use “^=” and “,” in the same command to essentially replace one item in the list with another. (See the Examples section.)

Additionally, as in **nodels**, boolean expressions can be used to further limit the scope of **nodech** from the given *noderange*. The operators supported are the same as **nodels** (=~, !~, ==, and !=).

With these operators in mind, the unambiguous assignment operator is ‘=@’. If you need, for example, to set the *nodelist.comments* to =foo, you would have to do *nodech n1 nodelist.comments=@=foo*.

See the **xcatdb** man page for an overview of each table.

The **nodech** command also supports some short cut names as aliases to common attributes. See the **nodels** man page for details.

## OPTIONS

**-d|--delete**

Delete the nodes’ row in the specified tables.

**-v|--version**

Command Version.

**-?|-h|--help**

Display usage message.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To update nodes in noderange node1-node4 to be in only group all:

```
nodech node1-node4 groups=all
```

2. To put all nodes with nodepos.rack value of 2 into a group called rack2:

```
nodech all nodepos.rack==2 groups,=rack2
```

3. To add nodes in noderange node1-node4 to the nodetype table with os=rhel5:

```
nodech node1-node4 groups=all,rhel5 nodetype.os=rhel5
```

4. To add node1-node4 to group1 in addition to the groups they are already in:

```
nodech node1-node4 groups,=group1
```

5. To put node1-node4 in group2, instead of group1:

```
nodech node1-node4 groups^=group1 groups,=group2
```

## FILES

/opt/xcat/bin/nodech

## SEE ALSO

nodes(1)|nodes.1, nodeadd(8)|nodeadd.8, noderange(3)|noderange.3

## nodechmac.1

## NAME

**nodechmac** - Updates the MAC address for a node.

## SYNOPSIS

**nodechmac** [-h | --help | -v | --version]

**nodechmac** *node-name* **mac**=*mac-address*



## DESCRIPTION

**Note:** The command **nodechmac** has been deprecated. To change the MAC address of the node:

```
makedhcp -d <nodename>
chdef -t node -o <nodename> mac=<new-mac>
makedhcp <nodename>
```

The **nodechmac** command changes the MAC address for provisioned node's network interface.

You can use this command to keep an existing node configuration. For example, if an existing node has hardware problems, the replacement node can use the old configurations. By using the **nodechmac** command, the node name and network settings of the old node can be used by the new node.

## OPTIONS

### **-h|--help**

Display usage message.

### **-v|--version**

Command Version.

*node-name*

Specifies the name of the node you want to update, where <node-name> is the node that is updated.

**mac=***mac-address*

Sets the new MAC address for the NIC used by the provisioning node, where <mac-address> is the NICs new MAC address.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

You can update the MAC address for a node, by using the following command:

```
nodechmac compute-000 mac=2F:3C:88:98:7E:01
```

## SEE ALSO

### nodechprofile.1

## NAME

**nodechprofile** - updates a profile used by a node

## SYNOPSIS

**nodechprofile** [-h| --help | -v | --version]

**nodechprofile** *noderange* [**imageprofile=** *image-profile*] [**networkprofile=** *network-profile*] [**hardwareprofile=** *hardware-profile*]

## DESCRIPTION

The **nodechprofile** command updates the profiles used by a node, including: the image profile, network profile, and hardware management profile.

If you update the image profile for a node, the operating system and provisioning settings for the node are updated.

If you update the network profile, the IP address and network settings for the node are updated.

If you update the hardware management profile, the hardware settings for the node are updated.

After nodes' hardware profile or image profile are updated, the status for each node is changed to "defined". A node with a "defined" status must be reinstalled

After nodes' network profile updated, the status for nodes is not changed. You'll need to run **noderegenips** to re-generate the nodes' IP address and nodes' status may also be updated at this stage.

## OPTIONS

### -h|--help

Display usage message.

### -v|--version

Command Version.

*noderange*

The nodes to be removed.

**imageprofile=** *image-profile*

Sets the new image profile name used by the node, where <image-profile> is the new image profile. An image profile defines the provisioning method, OS information, kit information, and provisioning parameters for a node. If the "\_\_ImageProfile\_imgprofile" group already exists in the nodehm table, then "imgprofile" is used as the image profile name.

**networkprofile=** *network-profile*

Sets the new network profile name used by the node, where <network-profile> is the new network profile. A network profile defines the network, NIC, and routes for a node. If the "\_\_NetworkProfile\_netprofile" group already exists in the nodehm table, then "netprofile" is used as the network profile name.

**hardwareprofile=** *hardware-profile*

Sets the new hardware profile name used by the node, where <hardware-profile> is the new hardware management profile used by the node. If a “\_\_HardwareProfile\_hwprofile” group exists, then “hwprofile” is the hardware profile name. A hardware profile defines hardware management related information for imported nodes, including: IPMI, HMC, CEC, CMM.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To change the image profile to rhels6.3\_packaged for compute nodes compute-000 and compute-001, use the following command:

```
nodechprofile compute-000,compute-001 imageprofile=rhels6.3_packaged
```

2. To change all of the profiles for compute node compute-000, enter the following command:

```
nodechprofile compute-000 imageprofile=rhels6.3_packaged_
↪networkprofile=default_cn hardwareprofile=default_ipmi
```

## SEE ALSO

nodepurge(1)|nodepurge.1, noderefresh(1)|noderefresh.1, nodeimport(1)|nodeimport.1, noderange(3)|noderange.3

## nodediscoverdef.1

## NAME

**nodediscoverdef** - Define the undefined discovery request to a predefined xCAT node, or clean up the discovery entries from the discoverydata table (which can be displayed by nodediscoverls command)

## SYNOPSIS

**nodediscoverdef** -u *uuid* -n *node*

**nodediscoverdef** -r -u *uuid*

**nodediscoverdef** -r -t {seq | profile | switch | blade | manual | undef | all}

**nodediscoverdef** [-h | --help | -v | --version]

## DESCRIPTION

The **nodediscoverdef** command defines the discovery entry from the discoverydata table to a predefined xCAT node. The discovery entry can be displayed by **nodediscoverls** command.

The options **-u** and **-n** have to be used together to define a discovery request to a node.

The **nodediscoverdef** command also can be used to clean up the discovery entries from the discoverydata table.

The option **-r** is used to remove discovery entries. If working with **-u**, the specific entry which uuid specified by **-u** will be removed.

You also can use the **-r -t** option to limit that only remove the nodes that were discovered in a particular method of discovery.

## OPTIONS

### **-t seq|profile|switch|blade>manual|undef|all**

Specify the nodes that have been discovered by the specified discovery method:

- \* **seq** - Sequential discovery (started via nodediscoverstart noderange=<noderange> ...).
- \* **profile** - Profile discovery (started via nodediscoverstart networkprofile=<network-profile> ...).
- \* **switch** - Switch-based discovery (used when the switch and switches tables are filled in).
- \* **blade** - Blade discovery (used for IBM Flex blades).
- \* **manual** - Manually discovery (used when defining node by nodediscoverdef command).
- \* **undef** - Display the nodes that were in the discovery pool, but for which xCAT has not yet received a discovery request.
- \* **all** - All discovered nodes.

### **-n node**

The xCAT node that the discovery entry will be defined to.

### **-r**

Remove the discovery entries from discoverydata table.

### **-u uuid**

The uuid of the discovered entry.

### **-h|--help**

Display usage message.

### **-v|--version**

Command version.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. Define the discovery entry which uuid is 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB4 to node node1

```
nodediscoverdef -u 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB4 -n node1
```

Output is similar to:

```
Defined [51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB4] to node node1.
```

2. Remove the discovery entry which uuid is 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB4 from the discovery-data table

```
nodediscoverdef -r -u 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB4
```

Output is similar to:

```
Removing discovery entries finished.
```

3. Remove the discovery entries which discover type is **seq** from the discoverydata table

```
nodediscoverdef -r -t seq
```

Output is similar to:

```
Removing discovery entries finished.
```

## SEE ALSO

nodediscoverstart(1)|nodediscoverstart.1, nodediscoverstatus(1)|nodediscoverstatus.1, nodediscoverstop(1)|nodediscoverstop.1, nodediscoverls(1)|nodediscoverls.1

## nodediscoverls.1

## NAME

**nodediscoverls** - List the discovered nodes

## SYNOPSIS

**nodediscoverls** [-t seq | profile | switch | blade | manual | undef | all] [-l]

**nodediscoverls** [-u *uuid*] [-l]

**nodediscoverls** [-h | --help | -v | --version]

## DESCRIPTION

The **nodediscoverls** command lists nodes that have recently been discovered. If discovery is currently in progress (i.e. **nodediscoverstart** has been run, but **nodediscoverstop** has not been), then **nodediscoverls** will list the nodes that have been discovered so far in this session. If discovery is not currently in progress, **nodediscoverls** will list all of the nodes that were discovered in the last discovery session.

You can use the **-t** option to limit the output to just the nodes that were discovered in a particular method of discovery.

## OPTIONS

**-t seq|profile|switch|blade|manual|undef|all**

Display the nodes that have been discovered by the specified discovery method:

- \* **seq** - Sequential discovery (started via **nodediscoverstart noderange=<noderange> ...**).
- \* **profile** - Profile discovery (started via **nodediscoverstart networkprofile=<network-profile> ...**).
- \* **switch** - Switch-based discovery (used when the switch and switches tables are filled in).
- \* **blade** - Blade discovery (used for IBM Flex blades).
- \* **manual** - Manually discovery (used when defining node by **nodediscoverdef** command).
- \* **undef** - Display the nodes that were in the discovery pool, but for which xCAT has not yet received a discovery request.
- \* **all** - All discovered nodes.

**-l**

Display more detailed information about the discovered nodes.

**-u *uuid***

Display the discovered node that has this uuid.

**-h|--help**

Display usage message.

**-v|--version**

Command version.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. Display the discovered nodes when sequential discovery is running:

```
nodediscoverls
```

Output is similar to:

UUID	SERIAL	NODE	METHOD	MTM
↪ 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB2	↪ 786310X 1052EF2	distest1	sequential	↪
↪ 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB3	↪ 786310X 1052EF3	distest2	sequential	↪

2. Display the nodes that were in the discovery pool, but for which xCAT has not yet received a discovery request:

```
nodediscoverls -t undef
```

Output is similar to:

UUID	SERIAL	NODE	METHOD	MTM
↪ 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB0	↪ 786310X 1052EF0	undef	undef	↪
↪ 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB1	↪ 786310X 1052EF1	undef	undef	↪

3. Display all the discovered nodes:

```
nodediscoverls -t all
```

Output is similar to:

UUID	SERIAL	NODE	METHOD	MTM
↪ 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB0	↪ 786310X 1052EF0	undef	undef	↪
↪ 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB1	↪ 786310X 1052EF1	undef	undef	↪
↪ 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB2	↪ 786310X 1052EF2	distest1	sequential	↪
↪ 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB3	↪ 786310X 1052EF3	distest2	sequential	↪

4. Display the discovered node whose uuid is **51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB2**, with detailed information:

```
nodediscoverls -u 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB2 -l
```

Output is similar to:

```
Object uuid: 51E5F2D7-0D59-11E2-A7BC-3440B5BEDBB2
node=distest1
method=sequential
discoverytime=03-31-2013 17:05:12
arch=x86_64
cpucount=32
cputype=Intel(R) Xeon(R) CPU E5-2690 0 @ 2.90GHz
memory=198460852
mtm=786310X
serial=1052EF2
nicdriver=eth0!be2net,eth1!be2net
nicipv4=eth0!10.0.0.212/8
nichwaddr=eth0!34:40:B5:BE:DB:B0,eth1!34:40:B5:BE:DB:B4
nicpci=eth0!0000:0c:00.0,eth1!0000:0c:00.1
nicloc=eth0!Onboard Ethernet 1,eth1!Onboard Ethernet 2
niconboard=eth0!1,eth1!2
nicfirm=eth0!ServerEngines BE3 Controller,eth1!ServerEngines BE3 Controller
switchname=eth0!c909f06sw01
switchaddr=eth0!192.168.70.120
switchdesc=eth0!IBM Flex System Fabric EN4093 10Gb Scalable Switch, flash_
image: version 7.2.6, boot image: version 7.2.6
switchport=eth0!INTA2
```

## SEE ALSO

nodediscoverstart(1)|nodediscoverstart.1, nodediscoverstatus(1)|nodediscoverstatus.1, nodediscoverstop(1)|nodediscoverstop.1, nodediscoverdef(1)|nodediscoverdef.1

## nodediscoverstart.1

### NAME

**nodediscoverstart** - starts the node discovery process

### SYNOPSIS

**nodediscoverstart** [-h | --help | -v | --version]

#### Sequential Discovery Specific:

**nodediscoverstart** *noderange=noderange* [*hostiprange=imageprofile*] [*bmciprange=bmciprange*] [*groups=groups*] [*rack=rack*] [*chassis=chassis*] [*height=height*] [*unit=unit*] [*osimage= osimagename*>] [-n | --dns] [-s | --skipbmcsetup] [-V|--verbose]

#### Profile Discovery Specific:



```
nodediscoverstart networkprofile=network-profile imageprofile=image-profile hostnameformat=host-name-format [hardwareprofile=hardware-profile] [groups=node-groups] [rack=rack-name] [chassis=chassis-name] [height=rack-server-height] [unit=rack-server-unit-location] [rank=rank-num]
```

## DESCRIPTION

The **nodediscoverstart** command starts either the **Sequential Discovery** or **Profile Discovery** process. They can not both be running at the same time.

### Sequential Discovery Specific:

This is the simplest discovery approach. You only need to specify the **noderange**, **hostiprange** and **bmciprange** that should be given to nodes that are discovered. (If you pre-define the nodes (via **nodeadd** or **mkdef**) and specify their host and BMC IP addresses, then you only need to specify the **noderange** to the **nodediscoverstart** command.) Once you have run **nodediscoverstart**, then physically power on the nodes in the sequence that you want them to receive the node names and IPs, waiting a short time (e.g. 30 seconds) between each node.

### Profile Discovery Specific:

This is the PCM discovery approach. *networkprofile*, *imageprofile*, *hostnameformat* arguments must be specified to start the **Profile Discovery**. All nodes discovered by this process will be associated with specified profiles and rack/chassis/unit locations.

When the nodes are discovered, PCM updates the affected configuration files on the management node automatically. Configuration files include the */etc/hosts* service file, DNS configuration, and DHCP configuration. Kit plug-ins are automatically triggered to update kit related configurations and services.

When you power on the nodes, they PXE boot and DHCP/TFTP/HTTP on the management node give each node the xCAT genesis boot image, which inventories the node hardware and sends data to the management node. There, either the sequential discovery process or the profile discovery process assigns node attributes and defines the node in the database.

## OPTIONS

**noderange**=*noderange*

The set of node names that should be given to nodes that are discovered via the **Sequential Discovery** method. This argument is required to **Sequential Discovery**. Any valid xCAT **noderange** is allowed, e.g. *node[01-10]*.

**hostiprange**=*ip range*

The ip range which will be assigned to the host of new discovered nodes in the **Sequential Discovery** method. The format can be: *start\_ip-end\_ip* or *noderange*, e.g. 192.168.0.1-192.168.0.10 or 192.168.0.[1-10].

**bmciprange**=*ip range*

The ip range which will be assigned to the bmc of new discovered nodes in the **Sequential Discovery** method. The format can be: *start\_ip-end\_ip* or *noderange*, e.g. 192.168.1.1-192.168.1.10 or 192.168.1.[1-10].

**imageprofile**=*image-profile*

Sets the new image profile name used by the discovered nodes in the **Profile Discovery** method. An image profile defines the provisioning method, OS information, kit information, and provisioning parameters for a node. If the “\_\_ImageProfile\_imgprofile” group already exists in the *nodehm* table, then “imgprofile” is used as the image profile name.

**networkprofile=network-profile**

Sets the new network profile name used by the discovered nodes in the **Profile Discovery** method. A network profile defines the network, NIC, and routes for a node. If the “\_\_NetworkProfile\_netprofile” group already exists in the nodehm table, then “netprofile” is used as the network profile name.

**hardwareprofile=hardware-profile**

Sets the new hardware profile name used by the discovered nodes in the **Profile Discovery** method. If a “\_\_HardwareProfile\_hwprofile” group exists, then “hwprofile” is the hardware profile name. A hardware profile defines hardware management related information for imported nodes, including: IPMI, HMC, CEC, CMM.

**hostnameformat=nost-name-format**

Sets the node name format for all discovered nodes in the **Profile Discovery** method. The two types of formats supported are prefix#NNNappendix and prefix#RRand#NNappendix, where wildcard #NNN and #NN are replaced by a system generated number that is based on the provisioning order. Wildcard #RR represents the rack number and stays constant.

For example, if the node name format is compute-#NN, the node name is generated as: compute-00, compute-01, ..., compute-99. If the node name format is blade#NNN-x64, the node name is generated as: blade001-x64, blade002-x64, ..., blade999-x64

For example, if the node name format is compute-#RR-#NN and the rack number is 2, the node name is generated as: compute-02-00, compute-02-01, ..., compute-02-99. If node name format is node-#NN-in-#RR and rack number is 1, the node name is generated as: node-00-in-01, node-01-in-01, ..., node-99-in-01

**groups=node-groups**

Sets the node groups that the discovered nodes should be put in for either the Sequential Discovery or Profile Discovery methods, where *node-group* is a comma-separated list of node groups.

**rack=rack-name>**

Sets the rack name where the node is located for either the Sequential Discovery or Profile Discovery methods.

**chassis=chassis-name**

Sets the chassis name that the Blade server or PureFlex blade is located in, for either the Sequential Discovery or Profile Discovery methods. This option is used for the Blade server and PureFlex system only. You cannot specify this option with the rack option.

**height=rack-server-height**

Sets the height of a rack-mounted server in U units for either the Sequential Discovery or Profile Discovery methods. If the rack option is not specified, the default value is 1.

**unit=rack-server-unit-location**

Sets the start unit value for the node in the rack, for either the Sequential Discovery or Profile Discovery methods. This option is for a rack server only. If the unit option is not specified, the default value is 1

**rank=rack-num**

Specifies the starting rank number that is used in the node name format, for the Profile Discovery method. The rank number must be a valid integer between 0 and 254. This option must be specified with node-nameformat option. For example, if your node name format is compute-#RR-#NN. The rack's number is 2 and rank is specified as 5, the node name is generated as follows: compute-02-05, compute-02-06, ..., compute-02-99.

**osimage=osimagename**

Specifies the osimage name that will be associated with the new discovered node, the os provisioning will be started automatically at the end of the discovery process.

#### **-n|--dns**

Specifies to run makedns <nodename> for any new discovered node. This is useful mainly for non-predefined configuration, before running the “nodediscoverstart -n”, the user needs to run makedns -n to initialize the named setup on the management node.

#### **-s|--skipbmcsetup**

Specifies to skip the bmcsetup during the sequential discovery process, if the bmciprange is specified with nodediscoverstart command, the BMC will be setup automatically during the discovery process, if the user does not want to run bmcsetup, could specify the “-s|--skipbmcsetup” with nodediscoverstart command to skip the bmcsetup.

#### **-V|--verbose**

Enumerates the free node names and host/bmc ips that are being specified in the ranges given. Use this option with Sequential Discovery to ensure that you are specifying the ranges you intend.

#### **-h|--help**

Display usage message.

#### **-v|--version**

Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. **Sequential Discovery:** To discover nodes with noderange and host/bmc ip range:

```
nodediscoverstart noderange=n[1-10] hostiprange='172.20.101.1-172.20.101.10' ↵
↵ bmciprange='172.20.102.1-172.20.102.10' -V
```

Output is similar to:

```
Sequential Discovery: Started:
  Number of free node names: 10
  Number of free host ips: 10
  Number of free bmc ips: 10
-----Free Nodes-----
↵ ---
NODE          HOST IP          BMC IP
n01           172.20.101.1    172.20.102.1
n02           172.20.101.2    172.20.102.2
...           ...           ...
```

2. **Profile Discovery:** To discover nodes using the default\_cn network profile and the rhels6.3\_package image profile, use the following command:

```
nodediscoverstart networkprofile=default_cn imageprofile=rhels6.3_packaged_
↪hostnameformat=compute#NNN
```

## SEE ALSO

nodediscoverstop(1)|nodediscoverstop.1,                      nodediscoverls(1)|nodediscoverls.1,                      nodediscoversta-  
tus(1)|nodediscoverstatus.1

## nodediscoverstatus.1

### NAME

**nodediscoverstatus** - gets the node discovery process status

### SYNOPSIS

**nodediscoverstatus** [-h | --help | -v | --version]

### DESCRIPTION

The **nodediscoverstatus** command detects if the sequential or profile node discovery process is currently running, i.e. **nodediscoverstart** has been run, but **nodediscoverstop** has not.

### OPTIONS

**-h|--help**

Display usage message.

**-v|--version**

Command Version.

### RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

To determine if there are some nodes discovered and the discovered nodes' status, enter the following command:

```
nodediscoverstatus
```

## SEE ALSO

nodediscoverstart(1)|nodediscoverstart.1, nodediscoverls(1)|nodediscoverls.1, nodediscoverstatus(1)|nodediscoverstop.1

## nodediscoverstop.1

## NAME

**nodediscoverstop** - stops the node discovery process.

## SYNOPSIS

**nodediscoverstop** [-h | --help | -v | --version]

## DESCRIPTION

The **nodediscoverstop** command stops the sequential or profile node discovery process. Once this command has been run, newly discovered nodes will not be assigned node names and attributes automatically via the sequential or profile discovery process.

## OPTIONS

**-h|--help**

Display usage message.

**-v|--version**

Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

```
nodediscoverstop
```

## SEE ALSO

nodediscoverstart(1)|nodediscoverstart.1,                      nodediscoverls(1)|nodediscoverls.1,                      nodediscoverstatus(1)|nodediscoverstatus.1

## nodegrpch.1

## NAME

**nodegrpch** - Changes attributes at the group level in the xCAT cluster database.

## SYNOPSIS

**nodegrpch** *group1,group2,... table.column=value [...]*

**nodegrpch** {-v | --version}

**nodegrpch** [-? | -h | --help]

## DESCRIPTION

The **nodegrpch** command is similar to the **nodech** command, but ensures that the parameters are declared at the group level rather than the node specific level, and clears conflicting node specific overrides of the specified groups. Using *table.column=value* will do a verbatim assignment. If “=” is used instead of “=”, the specified value will be prepended to the attribute’s comma separated list, if it is not already there. If “^=” is used, the specified value will be removed from the attribute’s comma separated list, if it is there. You can also use “^=” and “=” in the same command to essentially replace one item in the list with another. (See the Examples section.)

With these operators in mind, the unambiguous assignment operator is ‘=@’. If you need, for example, to set the *nodehm.comments* to =foo, you would have to do *nodegrpch group1 nodehm.comments=@=foo*.

See the **xcatdb** man page for an overview of each table.

The **nodegrpch** command also supports some short cut names as aliases to common attributes. See the **nodels** man page for details.

## OPTIONS

**-v|--version**

Command Version.

**-?|-h|--help**

Display usage message.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To declare all members of ipmi group to have nodehm.mgt be ipmi

```
nodegrpch ipmi nodehm.mgt=ipmi
```

## FILES

/opt/xcat/bin/nodegrpch

## SEE ALSO

nodech(1)|nodech.1, models(1)|models.1, nodeadd(8)|nodeadd.8, noderange(3)|noderange.3

## nodeimport.1

## NAME

**nodeimport** - Create profiled nodes by importing hostinfo file.

## SYNOPSIS

**nodeimport** [-h | --help | -v | --version]

**nodeimport** *file= hostinfo-filename networkprofile= network-profile imageprofile= image-profile hostnameformat= node-name-format [hardwareprofile= hardware-profile] [groups= node-groups]*

## DESCRIPTION

The **nodeimport** command creates nodes by importing a hostinfo file which is following stanza format. In this hostinfo file, we can define node's hostname, ip, mac, switch name, switch port and host location information like rack, chassis, start unit, server height...etc

After nodes imported, the configuration files related with these nodes will be updated automatically. For example: /etc/hosts, dns configuration, dhcp configuration. And the kits node plugins will also be triggered automatically to update kit related configuration/services.

## OPTIONS

### **-h|--help**

Display usage message.

### **-v|--version**

Command Version.

### **file=** *nodeinfo-filename*

Specifies the node information file, where <nodeinfo-filename> is the full path and file name of the node information file.

### **imageprofile=** *image-profile*

Sets the new image profile name used by the node, where <image-profile> is the new image profile. An image profile defines the provisioning method, OS information, kit information, and provisioning parameters for a node. If the “\_\_ImageProfile\_imgprofile” group already exists in the nodehm table, then “imgprofile” is used as the image profile name.

### **networkprofile=** *network-profile*

Sets the new network profile name used by the node, where <network-profile> is the new network profile. A network profile defines the network, NIC, and routes for a node. If the “\_\_NetworkProfile\_netprofile” group already exists in the nodehm table, then “netprofile” is used as the network profile name.

### **hardwareprofile=** *hardware-profile*

Sets the new hardware profile name used by the node, where <hardware-profile> is the new hardware management profile used by the node. If a “\_\_HardwareProfile\_hwprofile” group exists, then “hwprofile” is the hardware profile name. A hardware profile defines hardware management related information for imported nodes, including: IPMI, HMC, CEC, CMM.

### **hostnameformat=** *host-name-format*

Sets the node name format for all nodes discovered, where <node-name-format> is a supported format. The two types of formats supported are prefix#NNNappendix and prefix#RRand#NNappendix, where wildcard #NNN and #NN are replaced by a system generated number that is based on the provisioning order. Wildcard #RR represents the rack number and stays constant.

For example, if the node name format is compute-#NN, the node name is generated as: compute-00, compute-01, ... , compute-99. If the node name format is blade#NNN-x64, the node name is generated as: blade001-x64, blade002-x64, ... , blade999-x64

For example, if the node name format is compute-#RR-#NN and the rack number is 2, the node name is generated as: compute-02-00, compute-02-01, ... , compute-02-99. If node name format is node-#NN-in-#RR and rack number is 1, the node name is generated as: node-00-in-01, node-01-in-01, ... , node-99-in-01

### **groups=** *node-groups*

Sets the node groups that the imported node belongs to, where <node-group> is a comma-separated list of node groups.



## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred while validating parameters.
- 2 An error has occurred while parsing hostinfo file.

## EXAMPLES

To import nodes using a profile, follow the following steps:

1. Find all node groups and profiles, run the following command “`tabdump nodegroups`”. For detailed profile information run “`lsdef -t group <groupname>`”. Example of detailed profile information:

```
# tabdump nodegroup
#groupname,grouptype,members,membergroups,wherevals,comments,disable
"compute","static",,,,,,
"__HardwareProfile_default_ipmi","static","static",,,,,,
"__NetworkProfile_default_mn","static","static",,,,,,
"__NetworkProfile_default_cn","static",,,,,,
"__ImageProfile_rhels6.2-x86_64-install-compute","static","static",,,,,,

# lsdef -t group __NetworkProfile_default_cn
Object name: __NetworkProfile_default_cn
  grouptype=static
  installnic=eth0
  members=compute-000,compute-001
  netboot=xnba
  nichostnamesuffixes=eth0:-eth0
  nicnetworks=eth0:provision
  nictypes=eth0:Ethernet
  primarynic=eth0
```

2. Prepare a node information file.

Example of a node information file, a blade **and** a rack server **defined**:

```
# hostinfo begin
# This entry defines a blade.
__hostname__:
  mac=b8:ac:6f:37:59:24
  ip=192.168.1.20
  chassis=chassis01

# This entry defines a rack server.
__hostname__:
  mac=b8:ac:6f:37:59:25
  ip=192.168.1.20
  rack=rack01
  height=1
  unit=2

# hostinfo end.
```

(continues on next page)

(continued from previous page)

Another example of a node information file, a PureFlex X/P node **defined**:

```
# hostinfo begin
# To define a PureFlex P/X node, chassis and slot id must be specified.
# The chassis must be a PureFlex chassis.
__hostname__:
    mac=b8:ac:6f:37:59:25
    chassis=cmm01
    slotid=1
# hostinfo end.
```

Example of a node information file, a switch auto discovery node **defined**:

```
# hostinfo begin
# This entry defines a blade.
__hostname__:
    switches=eth0!switch1!1,eth0!switch2!1!eth1
```

Example of a node information file that specifies a CEC-based rack-mounted Power node, that uses direct FSP management:

```
# Node information file begins
# This entry defines a Power rack-mount node.
__hostname__:
    mac=b8:ac:6f:37:59:28
    cec=mycec

__hostname__:
    mac=b8:ac:6f:37:59:28
    cec=mycec
    lparid=2
# Node information file ends.
```

Example of a node information file that specifies a PowerKVM Guest node that uses KVM management:

```
# Node information file begins
# This entry defines a PowerKVM Guest node.
# Make sure the node 'vm01' is already created on Hypervisor
vm01:
    mac=b8:ef:3f:28:31:15
    vmhost=pkvm1
# Node information file ends.
```

The node information file includes the following items:

**\_\_hostname\_\_**: This is a mandatory item.

Description: The name of the node, where **\_\_hostname\_\_** is automatically generated by the node name format. You can also input a fixed node name, for example “compute-node”.

**mac=<mac-address>** This is a mandatory item.

Description: Specify the MAC address for the NIC used by the provisioning node, where <mac-address> is the NIC's MAC address.

**switches=<nic-name!switch-name!switch-port>** This is a mandatory item, when define switch, switchport and node nic name relationship.

Description: Specify nic name, switch name and switch port to define node and switch relationship. We can define multi nic-switch-port relations here, looks like: switches=eth0!switch1!1,eth1!switch1,2

**slotid=<slot-id>** This is a mandatory item while define a PureFlex node.

Description: The node position in the PureFlex Chassis.

**cec=<cec-name>** This is a mandatory option for defining Power rack-mounted nodes.

Description: Specifies the name of a Power rack-mount central electronic complex (CEC).

**lparid=<lpar-id>** This is a optional option for defining Power rack-mounted nodes.

Description: Specifies the LPAR ID of a Power rack-mounted node, where <lpar-id> is the ID number. The default value is 1 if it is not defined.

**ip=<ip-address>** This is an optional item.

Description: Specify the IP address used for provisioning a node, where <ip-address> is in the form xxx.xxx.xxx.xxx. If this item is not included, the IP address used to provision the node is generated automatically according to the Network Profile used by the node.

**nicips=<nics-ip>** This is an optional item.

Description: Lists the IP address for each network interface configuration (NIC) used by the node, excluding the provisioning network, where <nics-ip> is in the form <nic1>!<nic-ip1>,<nic2>!<nic-ip2>,... For example, if you have 2 network interfaces configured, the nicips attribute should list both network interfaces: nicips=eth1!10.10.10.11,bmc!192.168.10.3. If the nicips attribute is not specified, the IP addresses are generated automatically according to the network profile.

**rack=<rack-name>** This is an optional item.

Description: node location info. Specify the rack name which this node will be placed into. If not specify this item, there will be no node location info set for this node. this item must be specified together with height + unit.

**chassis=<chassis-name>** This is an optional item.

Description: node location info, for blade(or PureFlex) only. Specify the chassis name which this blade will be placed into. this item can not be specified together with rack.

**height=<chassis-height>** This is an optional item.

Description: node location info, for rack server only. Specify the server height number, in U. this item must be specified together with rack and unit.

**unit=<rack-server-unit-location>** This is an optional item.

Description: node location info, for rack server only. Specify the node's start unit number in rack, in U. this item must be specified together with rack and height.

**vmhost=<PowerKVM Hypervisor Host Name>** This is a mandatory option for defining PowerKVM Guest nodes.

Description: Specifies the vmhost of a Power KVM Guest node, where <vmhost> is the host name of PowerKVM Hypervisor.

3. Import the nodes, by using the following commands. Note: If we want to import PureFlex X/P nodes, hardware profile must be set to a PureFlex hardware type.

```
nodeimport file=/root/hostinfo.txt networkprofile=default_cn imageprofile=rhels6.3_
↪ packaged hostnameformat=compute-#NNN
```

4. After importing the nodes, the nodes are created and all configuration files used by the nodes are updated, including: /etc/hosts, DNS, DHCP.
5. Reboot the nodes. After the nodes are booted they are provisioned automatically.

## SEE ALSO

nodepurge(1)|nodepurge.1, nodechprofile(1)|nodechprofile.1, noderefresh(1)|noderefresh.1

## nodels.1

### NAME

**nodels** - lists the nodes, and their attributes, from the xCAT database.

### SYNOPSIS

**nodels** [*noderange*] [-b | --blame] [-H | --with-fieldname] [-S] [*table.column* | *shortname*] [...]

**nodels** [*noderange*] [-H | --with-fieldname] [*table*]

**nodels** [-? | -h | --help | -v | --version]

### DESCRIPTION

The **nodels** command lists the nodes specified in the node range. If no *noderange* is provided, then all nodes are listed.

Additional attributes of the nodes will also be displayed if the table names and attribute names are specified after the *noderange* in the form: *table.column* . A few shortcut names can also be used as aliases to common attributes:

#### groups

nodelist.groups

#### tags

nodelist.groups

#### mgt

nodehm.mgt

nodels can also select based on table value criteria. The following operators are available:

==

Select nodes where the *table.column* value is exactly a certain value.

!=

Select nodes where the *table.column* value is not a given specific value.

=~

Select nodes where the *table.column* value matches a given regular expression.

!~

Select nodes where the *table.column* value does not match a given regular expression.

The **nodels** command with a specific node and one or more *table.attribute* parameters is a good substitute for *grep*'ing through the tab files, as was typically done in xCAT 1.x. This is because *nodels* will translate any regular expression rows in the tables into their meaning for the specified node. The *tab\** commands will not do this, instead they will just display the regular expression row verbatim.

## OPTIONS

### **-v|--version**

Command Version.

### **-H|--with-fieldname**

Force display of table name and column name context for each result

### **-b|--blame**

For values inherited from groups, display which groups provided the inheritance

### **-S**

List all the hidden nodes (FSP/BPA nodes) with other ones.

### **-?|-h|--help**

Display usage message.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To list all defined nodes, enter:

```
node ls
```

Output is similar to:

```
node1
node2
node3
```

2. To list all defined attributes in a table for a node or noderange, enter:

```
node ls rra001a noderes
```

Output is similar to:

```
rra001a: noderes.primarynic: eth0
rra001a: noderes.xcatmaster: rra000
rra001a: noderes.installnic: eth0
rra001a: noderes.netboot: pxe
rra001a: noderes.servicenode: rra000
rra001a: noderes.node: rra001a
```

3. To list nodes in node group ppc, enter:

```
node ls ppc
```

Output is similar to:

```
ppcnode1  
ppcnode2  
ppcnode3
```

4. To list the groups each node is part of:

```
node1s all groups
```

Output is similar to:

```
node1: groups: all  
node2: groups: all,storage  
node3: groups: all,blade
```

5. To list the groups each node is part of:

```
node1s all nodehm.power
```

Output is similar to:

```
node1: nodehm.power: blade  
node2: nodehm.power: ipmi  
node3: nodehm.power: ipmi
```

6. To list the out-of-band mgt method for blade1:

```
node1s blade1 nodehm.mgt
```

Output is similar to:

```
blade1: blade
```

7. Listing blades managed through an AMM named 'amm1'

```
node1s all mp.mpa==amm1
```

Output is similar to:

```
blade1  
blade10  
blade11  
blade12  
blade13  
blade2  
blade3  
blade4  
blade5  
blade6  
blade7  
blade8  
blade9
```

8. Listing the switch.switch value for nodes in the second rack:

```
nodecls all nodepos.rack==2 switch.switch
```

Output is similar to:

```
n41: switch.switch: switch2
n42: switch.switch: switch2
n43: switch.switch: switch2
n44: switch.switch: switch2
n45: switch.switch: switch2
n46: switch.switch: switch2
n47: switch.switch: switch2
n55: switch.switch: switch2
n56: switch.switch: switch2
n57: switch.switch: switch2
n58: switch.switch: switch2
n59: switch.switch: switch2
n60: switch.switch: switch2
```

9. Listing the blade slot number for anything managed through a device with a name beginning with amm:

```
nodecls all mp.mpa=~/^amm.*/ mp.id
```

Output looks like:

```
blade1: mp.id: 1
blade10: mp.id: 10
blade11: mp.id: 11
blade12: mp.id: 12
blade13: mp.id: 13
blade2: mp.id: 2
blade3: mp.id: 3
blade4: mp.id: 4
blade5: mp.id: 5
blade6: mp.id: 6
blade7: mp.id: 7
blade8: mp.id: 8
blade9: mp.id: 9
```

10. To list the hidden nodes that can't be seen with other flags. The hidden nodes are FSP/BPAs.

```
lsdef -S
```

## FILES

/opt/xcat/bin/nodels

## SEE ALSO

noderange(3)|noderange.3, tabdump(8)|tabdump.8, lsdef(1)|lsdef.1

## nodepurge.1

## NAME

**nodepurge** - Removes nodes.

## SYNOPSIS

**nodepurge** [-h| --help | -v | --version]

**nodepurge** *noderange*

## DESCRIPTION

The **nodepurge** automatically removes all nodes from the database and any related configurations used by the node.

After the nodes are removed, the configuration files related to these nodes are automatically updated, including the following files: */etc/hosts*, DNS, DHCP. Any kits that are used by the nodes are triggered to automatically update kit configuration and services. Any related configuration files from */install/autoinst* are also removed.

## OPTIONS

**-h|--help** Display usage message.

**-v|--version** Command Version

*noderange* The nodes to be removed.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.



## EXAMPLES

To remove nodes compute-000 and compute-001:

```
nodepurge compute-000,compute-001
```

## SEE ALSO

nodeimport(1)|nodeimport.1,  
noderange(3)|noderange.3

nodechprofile(1)|nodechprofile.1,

noderefresh(1)|noderefresh.1,

## noderefresh.1

## NAME

**noderefresh** - Update nodes configurations by running associated kit plugins.

## SYNOPSIS

**noderefresh** [-h|--help | -v | --version]

**noderefresh** *noderange*

## DESCRIPTION

The **noderefresh** command will update nodes settings, it will call all associated kit plug-in configurations and also services

## OPTIONS

**-h|--help**

Display usage message.

**-v|--version**

Command Version.

*noderange*

The nodes to be updated.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

```
noderefresh compute-000,compute-001
```

## SEE ALSO

nodeimport(1)|nodeimport.1, nodechprofile(1)|nodechprofile.1, nodepurge(1)|nodepurge.1, noderange(3)|noderange.3

## noderm.1

## NAME

**noderm** -Removes the nodes in the noderange from all database table.

## SYNOPSIS

**noderm [-h] --help]**

**noderm noderange**

## DESCRIPTION

The noderm command removes the nodes in the input node range.

## OPTIONS

**-h|--help** Display usage message.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To remove the nodes in noderange node1-node4, enter:

```
noderm node1-node4
```

## FILES

/opt/xcat/bin/noderm

## SEE ALSO

nodes(1)|nodes.1, nodeadd(8)|nodeadd.8, noderange(3)|noderange.3

## nodestat.1

### Name

**nodestat** - display the running status of each node in a noderange

### Synopsis

**nodestat** [*noderange*] [-m | --usemon] [-p | --powerstat] [-f | --usefping] [-u | --updatedb]

**nodestat** [-h | --help | -v | --version]

### Description

**nodestat** displays and optionally updates the database the running status of a single or range of nodes or groups. See noderange(3)|noderange.3.

#### By default, it works as following:

1. gets the sshd,pbs\_mom,xend port status;
2. if none of them are open, it gets the fping status;
3. for pingable nodes that are in the middle of deployment, it gets the deployment status;
4. for non-pingable nodes, it shows 'noping'.

When -m is specified and there are settings in the monsetting table, it displays the status of the applications specified in the monsetting table. When -p is specified it shows the power status for the nodes that are not pingable. When -u is specified it saves the status info into the xCAT database. Node's pingable status and deployment status is saved in the nodelist.status column. Node's application status is saved in the nodelist.appstatus column.

To specify settings in the **monsetting** table, use 'xcatmon' as the name, 'apps' as the key and the value will be a list of comma separated list of application names. For each application, you can specify the port number that can be queried on the nodes to get the running status. Or you can specify a command that can be called to get the node status from. The command can be a command that can be run locally at the management node or the service node for hierarchical cluster, or a command that can be run remotely on the nodes.

The following is an example of the settings in the **monsetting** table:

```
name key value
xcatmon apps ssh,ll,gpfs,someapp
xcatmon gpfs cmd=/tmp/mycmd,group=compute,group=service
xcarmon ll port=9616,group=compute
xcatmon someapp dcmd=/tmp/somecmd
```

Keywords to use:

```
apps -- a list of comma separated application names whose status will be queried. For
↳ how to get the status of each app, look for app name in the key field in a different
↳ row.
port -- the application daemon port number, if not specified, use internal list, then /
↳ etc/services.
group -- the name of a node group that needs to get the application status from. If not
↳ specified, assume all the nodes in the nodelist table. To specify more than one groups,
↳ use group=a,group=b format.
cmd -- the command that will be run locally on mn or sn.
lcmd -- the command that will be run the mn only.
dcmd -- the command that will be run distributed on the nodes using xdsh <nodes> ....
```

For commands specified by ‘cmd’ and ‘lcmd’, the input of is a list of comma separated node names, the output must be in the following format:

```
node1:string1
node2:string2
...
```

For the command specified by ‘dcmd’, no input is needed, the output can be a string.

## Options

### **-f | --usefping**

Uses fping instead of nmap even if nmap is available. If you seem to be having a problem with false negatives, fping can be more forgiving, but slower.

### **-m | --usemon**

Uses the settings from the **monsetting** table to determine a list of applications that need to get status for.

### **-p | --powerstat**

Gets the power status for the nodes that are ‘noping’.

### **-u | --updatedb**

Updates the status and appstatus columns of the nodelist table with the returned running status from the given nodes.

### **-v | --version**

Print version.

### **-h | --help**

Print help.

## Examples

1. `nodestat compute`

Output is similar to:

```
node1  sshd
node2  sshd
node3  ping
node4  pbs
node5  noping
```

2. `nodestat compute -p`

Output is similar to:

```
node1  sshd
node2  sshd
node3  ping
node4  pbs
node5  noping(Shutting down)
```

3. `nodestat compute -u`

Output is similar to:

```
node1  sshd
node2  sshd
node3  ping
node4  netboot
node5  noping
```

4. `nodestat compute -m`

Output is similar to:

```
node1  ping,sshd,ll,gpfs=ok
node2  ping,sshd,ll,gpfs=not ok,someapp=something is wrong
node3  netboot
node4  noping
```

## See Also

`noderange(3)|noderange.3`, `nodels(1)|nodels.1`, `nodeset(8)|nodeset.8`

## packimage.1

### NAME

**packimage** - Packs the stateless image from the chroot file system.

### SYNOPSIS

**packimage** [-h] --help]

**packimage** [-v] --version]

**packimage** [-m | --method cpio|tar|squashfs] [-c | --compress gzip|pigz|xz] [--nosyncfiles] *imagename*

### DESCRIPTION

Packs the stateless image from the chroot file system into a file to be sent to the node for a diskless boot.

Note: For an osimage that is deployed on a cluster, running **packimage** will overwrite the existing rootimage file and be unavailable to the compute nodes while **packimage** is running.

### PARAMETERS

*imagename* specifies the name of a OS image definition to be used. The specification for the image is stored in the *osimage* table and *linuximage* table.

### OPTIONS

**-h**

Display usage message.

**-v**

Command Version.

**-m | --method**

Archive Method (cpio, tar, squashfs (requires overlayFS or aufs modules during provisioning), default is cpio)

**-c | --compress**

Compress Method (pigz, gzip, xz, default is pigz/gzip)

**--nosyncfiles**

Bypass of syncfiles requested, will not sync files to root image directory

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To pack the osimage 'rhels7.1-x86\_64-netboot-compute':

```
packimage rhels7.1-x86_64-netboot-compute
```

2. To pack the osimage 'rhels7.1-x86\_64-netboot-compute' with "tar" to archive and "pigz" to compress:

```
packimage -m tar -c pigz rhels7.1-x86_64-netboot-compute
```

3. To pack the osimage 'rhels7.1-x86\_64-netboot-compute' without syncing files:

```
packimage --nosyncfiles rhels7.1-x86_64-netboot-compute
```

## FILES

/opt/xcat/sbin/packimage

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

genimage(1)|genimage.1

## pasu.1

## NAME

**pasu** - run the ASU to many nodes in parallel

## SYNOPSIS

**pasu** [-V] [-d] [-l *user*] [-p *passwd*] [-f *fanout*] [-i *hostname-suffix*] *noderange command*

**pasu** [-V] [-d] [-l *user*] [-p *passwd*] [-f *fanout*] [-i *hostname-suffix*] **-b** *batchfile noderange*

**pasu** [-h | --help]

## DESCRIPTION

The **pasu** command runs the ASU command in out-of-band mode in parallel to multiple nodes. Out-of-band mode means that ASU connects from the xCAT management node to the IMM (BMC) of each node to set or query the ASU settings. To see all of the ASU settings available on the node, use the “show all” command. To query or set multiple values, use the **-b** (batch) option. To group similar output from multiple nodes, use **xcoll**.

Before running **pasu**, you must install the ASU RPM. At the time of this writing, the latest version can be downloaded from <https://support.lenovo.com/us/en/solutions/ht115050-advanced-settings-utility-asu>. Older versions can be found on the IBM Fix Central site. You also must configure the IMM properly according to xCAT documentation. Run “**rpower noderange stat**” to confirm that the IMM is configured properly.

## OPTIONS

**-l|--loginname** *username*

The username to use to connect to the IMM. If not specified, the row in the xCAT *passwd* table with key “ipmi” will be used to get the username.

**-p|--passwd** *passwd*

The password to use to connect to the IMM. If not specified, the row in the xCAT *passwd* table with key “ipmi” will be used to get the password.

**-f|--fanout**

How many processes to run in parallel simultaneously. The default is 64. You can also set the XCATPSH-FANOUT environment variable.

**-b|--batch** *-batchfile*

A simple text file that contains multiple ASU commands, each on its own line.

**-d|--donotfilter**

By default, **pasu** filters out (i.e. does not display) the standard initial output from ASU:

```
IBM Advanced Settings Utility version 9.30.79N
Licensed Materials - Property of IBM
(C) Copyright IBM Corp. 2007-2012 All Rights Reserved
Connected to IMM at IP address node2-imm
```

If you want this output to be displayed, use this flag.

**-i|--interface** *hostname-suffix*

The hostname suffix to be appended to the node names.

**-V|--verbose**

Display verbose messages.

**-h|--help**

Display usage message.



## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To display the Com1ActiveAfterBoot setting on 2 nodes:

```
pasu node1,node2 show DevicesandIOPorts.Com1ActiveAfterBoot
```

Output is similar to:

```
node1: DevicesandIOPorts.Com1ActiveAfterBoot=Enable
node2: DevicesandIOPorts.Com1ActiveAfterBoot=Enable
```

2. To display the Com1ActiveAfterBoot setting on all compute nodes:

```
pasu compute show DevicesandIOPorts.Com1ActiveAfterBoot | xcoll
```

Output is similar to:

```
=====
compute
=====
DevicesandIOPorts.Com1ActiveAfterBoot=Enable
```

3. To set several settings on all compute nodes, create a batch file called (for example) asu-settings with contents:

```
set DevicesandIOPorts.Com1ActiveAfterBoot Enable
set DevicesandIOPorts.SerialPortSharing Enable
set DevicesandIOPorts.SerialPortAccessMode Dedicated
set DevicesandIOPorts.RemoteConsole Enable
```

Then run:

```
pasu -b asu-settings compute | xcoll
```

Output is similar to:

```
=====
compute
=====
Batch mode start.
[set DevicesandIOPorts.Com1ActiveAfterBoot Enable]
DevicesandIOPorts.Com1ActiveAfterBoot=Enable

[set DevicesandIOPorts.SerialPortSharing Enable]
DevicesandIOPorts.SerialPortSharing=Enable

[set DevicesandIOPorts.SerialPortAccessMode Dedicated]
DevicesandIOPorts.SerialPortAccessMode=Dedicated
```

(continues on next page)

(continued from previous page)

```
[set DevicesandIOPorts.RemoteConsole Enable]
DevicesandIOPorts.RemoteConsole=Enable
```

```
Beginning intermediate batch update.
Waiting for command completion status.
Command completed successfully.
Completed intermediate batch update.
Batch mode competed successfully.
```

4. To confirm that all the settings were made on all compute nodes, create a batch file called (for example) asu-show with contents:

```
show DevicesandIOPorts.Com1ActiveAfterBoot
show DevicesandIOPorts.SerialPortSharing
show DevicesandIOPorts.SerialPortAccessMode
show DevicesandIOPorts.RemoteConsole
```

Then run:

```
pasu -b asu-show compute | xcoll
```

Output is similar to:

```
=====
compute
=====
Batch mode start.
[show DevicesandIOPorts.Com1ActiveAfterBoot]
DevicesandIOPorts.Com1ActiveAfterBoot=Enable

[show DevicesandIOPorts.SerialPortSharing]
DevicesandIOPorts.SerialPortSharing=Enable

[show DevicesandIOPorts.SerialPortAccessMode]
DevicesandIOPorts.SerialPortAccessMode=Dedicated

[show DevicesandIOPorts.RemoteConsole]
DevicesandIOPorts.RemoteConsole=Enable

Batch mode competed successfully.
```

## FILES

/opt/xcat/bin/pasu

## SEE ALSO

noderange(3)|noderange.3, rpower(1)|rpower.1, xcoll(1)|xcoll.1

### pcons.1

## SYNOPSIS

**pcons** *noderange command*

**pcons**[-h | --help]

**pcons**[-v | --version]

## DESCRIPTION

Runs a command to the noderange using the console.

## EXAMPLES

```
pcons 1,3 stat
pcons all,-129-256 stat
```

## SEE ALSO

psh(1)|psh.1

### pgsqlsetup.1

## NAME

**pgsqlsetup** - Sets up the PostgreSQL database for xCAT to use.

## SYNOPSIS

**pgsqlsetup** {-h | --help}

**pgsqlsetup** {-v | --version}

**pgsqlsetup** {-i | --init} [-N | --nostart] [--listen | -l *address*] [--access | -a *address*] [-P | --PCM] [-o | --odbc] [-V | --verbose]

**pgsqlsetup** {-o | --setupODBC} [-V | --verbose]

## DESCRIPTION

**pgsqlsetup** - Sets up the PostgreSQL database for xCAT to use. The **pgsqlsetup** script is run on the Management Node as root after the PostgreSQL has been installed. The **xcatd** daemon will be stopped during migration. No xCAT commands should be run during the init process, because we will be migrating the xCAT database to PostgreSQL and restarting the **xcatd** daemon as well as the PostgreSQL daemon. One password must be supplied for the setup, a password for the xcatadm unix id and the same password for the xcatadm database id. The password will be prompted for interactively or you can set the XCATPGPW environment variable to the password in order to avoid the prompt.

## OPTIONS

**-h|--help**

Displays the usage message.

**-v|--version**

Displays the release version of the code.

**-V|--verbose**

Displays verbose messages.

**-i|--init**

The **--init** option is used to setup an installed PostgreSQL database so that xCAT can use it. This involves creating the xcat database, the xcat admin id, allowing access to the xcatdb database by the Management Node. It customizes the postgresql.conf configuration file, adds the management server to the pg\_hba.conf and starts the PostgreSQL server. It also backs up the current xCAT database and restores it into the newly setup xcatdb PostgreSQL database. It creates the /etc/xcat/cfgloc file to point the **xcatd** daemon to the PostgreSQL database and restarts the **xcatd** daemon using the database. On AIX, it additionally setup the xcatadm unix id and the postgres id and group. For AIX, you should be using the PostgreSQL rpms available from the xCAT website. For Linux, you should use the PostgreSQL rpms shipped with the OS. You can chose the **-o** option, to run after the init. To add additional nodes to access the PostgreSQL server, setup on the Management Node, use the **-a** option.

For more documentation see:<<https://xcat-docs.readthedocs.io/en/stable/advanced/hierarchy/databases/index.html#postgresql>>

**-N|--nostart**

This option with the **-i** flag will create the database, but will not backup and restore xCAT tables into the database. It will create the cfgloc file such that the next start of **xcatd** will try and contact the database. This can be used to setup the xCAT PostgreSQL database during or before install.

**-l|--listen** *address*

This option is used to specify additional IP addresses on which the PostgreSQL database will listen. Without it, only localhost (on Linux) and the management node's main IP (on Linux and AIX) will be configured. This option can be specified multiple times.

#### **-a|--access *address***

This option is used to specify additional IP addresses from which the additional nodes will connect to the PostgreSQL database, for example, service nodes IP addresses or MN HA primary/standby nodes physical IP addresses. Without it, only the management node will be configured for database access. This option can be specified multiple times.

#### **-P|--PCM**

This option sets up PostgreSQL database to be used with xCAT running with PCM.

#### **-o|--odbc**

This option sets up the ODBC `/etc/./odbcinst.ini`, `/etc/./odbc.ini` and the `.odbc.ini` file in roots home directory will be created and initialized to run off the `xcatdb` PostgreSQL database.

## ENVIRONMENT VARIABLES

### **XCATPGPW**

The password to be used to setup the xCAT admin id for the database.

## EXAMPLES

1. To setup PostgreSQL for xCAT to run on the PostgreSQL `xcatdb` database :

```
pgsqlsetup -i
```

2. To setup the ODBC for PostgreSQL `xcatdb` database access :

```
pgsqlsetup -o
```

### **piflash.1**

## SYNOPSIS

**piflash** *noderange* -**\*\*\*\***-*package filename*

## DESCRIPTION

**piflash** Remotely applies firmware updates to servers.

## pping.1

### SYNOPSIS

```
pping [-i | --interface interfaces] [-f | --use_fping] noderange
```

```
pping [-h | --help]
```

```
pping {-v | --version}
```

### DESCRIPTION

**pping** is a utility used to ping a list of nodes in parallel. **pping** will return an unsorted list of nodes with a ping or noping status. **pping** front-ends **nmap** or **fping** if available.

This command does not support the xcatd client/server communication. It must be run on the management node.

### OPTIONS

**-i | --interface *interfaces***

A comma separated list of network interface names that should be pinged instead of the interface represented by the nodename/hostname. The following name resolution convention is assumed: an interface is reachable by the hostname <nodename>-<interface>. For example, the ib2 interface on node3 has a hostname of node3-ib2.

If more than one interface is specified, each interface will be combined with the nodenames as described above and will be pinged in turn.

**-f | --use\_fping**

Use **fping** instead of **nmap**

**-h | --help**

Show usage information.

**-v | --version**

Display the installed version of xCAT.

### EXAMPLES

```
1. pping all
```

Output is similar to:

```
node1: ping
node2: ping
node3: noping
```

```
2. pping all -i ib0,ib1
```

Output is similar to:

```
node1-ib0: ping
node2-ib0: ping
node3-ib0: noping
node1-ib1: ping
node2-ib1: ping
node3-ib1: noping
```

## SEE ALSO

psh(1)|psh.1, noderange(3)|noderange.3

## ppping.1

## SYNOPSIS

```
ppping [-i | --interface interfaces] [-d | --debug] [-V | --verbose] [-q | --quiet] [-s | --serial] noderange
ppping [-h | --help]
ppping {-v | --version}
```

## DESCRIPTION

**ppping** is a utility used to test the connectivity between nodes in the noderange using ping. By default, **ppping** will return an unsorted list of the node pairs that are not able to ping each other, or a message that all nodes are pingable. More or less output can be controlled by the -V and -q options. **ppping** front-ends **ppping** and **xdsh**.

## OPTIONS

**-s**

Ping serially instead of in parallel.

**-i | --interface** *interfaces*

A comma separated list of network interface names that should be pinged instead of the interface represented by the nodename/hostname. The following name resolution convention is assumed: an interface is reachable by the hostname <nodename>-<interface>. For example, the ib2 interface on node3 has a hostname of node3-ib2.

If more than one interface is specified, each interface will be combined with the nodenames as described above and will be pinged in turn.

**-V | --verbose**

Display verbose output. The result of every ping attempt from every node will be displayed. Without this option, just a summary of the successful pings are displayed, along with all of the unsuccessful pings.

**-q | --quiet**

Display minimum output: just the unsuccessful pings. This option has the effect that if all pings are successful, nothing is displayed. But it also has the performance benefit that each node does not have to send successful ping info back to the management node.

**-d | --debug**

Print debug information.

**-h | --help**

Show usage information.

**-v | --version**

Display the installed version of xCAT.

## EXAMPLES

1. `ppping all -q`

Output is similar to:

```
blade7: node2: noping
blade8: node2: noping
blade9: node2: noping
devmaster: node2: noping
node2: noping
```

2. `ppping node1,node2 -i ib0,ib1,ib2,ib3`

Output is similar to:

```
node1: pinged all nodes successfully on interface ib0
node1: pinged all nodes successfully on interface ib1
node1: pinged all nodes successfully on interface ib2
node1: pinged all nodes successfully on interface ib3
node2: pinged all nodes successfully on interface ib0
node2: pinged all nodes successfully on interface ib1
node2: pinged all nodes successfully on interface ib2
node2: pinged all nodes successfully on interface ib3
```

## SEE ALSO

`psh(1)|psh.1`, `pping(1)|pping.1`

## prsync.1

### Name

prsync - parallel rsync



## Synopsis

**prsync** *filename* [*filename ...*] *noderange:destinationdirectory*

**prsync** [-o *rsyncopts*] [-f *fanout*] [*filename filename ...*] [*directory directory ...*] *noderange:destinationdirectory*

**prsync** {-h | --help | -v | --version}

## Description

**prsync** is a front-end to rsync for a single or range of nodes and/or groups in parallel.

Note: this command does not support the xcatd client/server communication and therefore must be run on the management node. It does not support hierarchy, use **xdcp -F** to run rsync from the management node to the compute node via a service node

**prsync** is NOT multicast, but is parallel unicasts.

## Options

**-o** *rsyncopts*

rsync options. See **rsync(1)**.

**-f** *fanout*

Specifies a fanout value for the maximum number of concurrently executing remote shell processes.

*filename*

A space delimited list of files to rsync.

*directory*

A space delimited list of directories to rsync.

*noderange:destination*

A noderange(3)|noderange.3 and destination directory. The : is required.

**-h | --help**

Print help.

**-v | --version**

Print version.

## XCATPSHFANOUT

Specifies the fanout value. This variable is overridden by the **-f** flag. Default is 64.

## Examples

1. `cd /install; prsync -o "crz" post stage:/install`
2. `prsync passwd group rack01:/etc`

## See Also

noderange(3)|noderange.3, pscp(1)|pscp.1, pping(1)|pping.1, psh(1)|psh.1

## pscp.1

### Name

**pscp** - parallel remote copy

### Synopsis

**pscp** [-i *suffix*] [*scp options* ...] [-f *fanout*] *filename* [*filename* ...] *noderange:destinationdirectory*

**pscp** {-h | --help | -v | --version}

### Description

**pscp** is a utility used to copy a single or multiple set of files and/or directories to a single or range of nodes and/or groups in parallel.

**pscp** is a front-end to the remote copy **scp**.

Note: this command does not support the xcatd client/server communication and therefore must be run on the management node. It does not support hierarchy, use **xdcp** to run remote copy command from the management node to the compute node via a service node.

**pscp** is NOT multicast, but is parallel unicasts.

### Options

#### **-f** *fanout*

Specifies a fanout value for the maximum number of concurrently executing remote shell processes.

#### **-i** *suffix*

Interfaces to be used.

#### *scp options*

See **scp(1)**

#### *filename*

A space delimited list of files to copy. If **-r** is passed as an **scp** option, directories may be specified as well.

*noderange:destination*

A noderange(3)|noderange.3 and destination directory. The : is required.

**-h | --help**

Print help.

**-v | --version**

Print version.

**XCATPSHFANOUT**

Specifies the fanout value. This variable is overridden by the **-f** flag. Default is 64.

## Examples

```
1. pscp -r /usr/local node1,node3:/usr/local
```

```
2. pscp passwd group rack01:/etc
```

## See Also

noderange(3)|noderange.3, pping(1)|pping.1, prsync(1)|prsync.1, psh(1)|psh.1

## psh.1

### Name

psh - parallel remote shell

### Synopsis

**psh** [-i *interface*] [-f *fanout*] [-l *user*] *noderange command*

**psh** {-h | --help | -v | --version}

### Description

**psh** is a utility used to run a command across a list of nodes in parallel.

**ssh** must be set up to allow no prompting for **psh** to work.

Note:

This command does not run through xcatd like most xCAT commands do. This means you must either run it on the management node, or have a network connection between your machine and the nodes. It does not support hierarchy, use xdsh to run remote command from the management node to the compute node via a service node.

**psh** arguments need to precede noderange, otherwise, you will get unexpected errors.

## Options

### **-i** *interface*

The NIC on the node that psh should communicate with. For example, if *interface* is **eth1**, then psh will concatenate **-eth1** to the end of every node name before ssh'ing to it. This assumes those host names have been set up to resolve to the IP address of each of the eth1 NICs.

### **-f** *fanout*

Specifies a fanout value for the maximum number of concurrently executing remote shell processes.

### **-l** *user*

Log into the nodes as the specified username. The default is to use the same username as you are running the psh command as.

### **-n|--nonodecheck**

Do not send the noderange to xcatd to expand it into a list of nodes. Instead, use the noderange exactly as it is specified. In this case, the noderange must be a simple list of comma-separated hostnames of the nodes. This allows you to run **psh** even when xcatd is not running.

### *noderange*

See noderange(3)|noderange.3.

### *command*

Command to be run in parallel. If no command is given then **psh** enters interactive mode. In interactive mode a ">" prompt is displayed. Any command entered is executed in parallel to the nodes in the noderange. Use "exit" or "Ctrl-D" to end the interactive session.

### **-h | --help**

Print help.

## Environment Variables

### **XCATPSHFANOUT**

Specifies the fanout value. This variable is overridden by the **-f** flag. Default is 64.

## Examples

1. Run uptime on 3 nodes:

```
psh node4-node6 uptime
```

Output is similar to:

```
node4: Sun Aug  5 17:42:06 MDT 2001
node5: Sun Aug  5 17:42:06 MDT 2001
node6: Sun Aug  5 17:42:06 MDT 2001
```

2. Run a command on some BladeCenter management modules:

```
psh amm1-amm5 'info -T mm[1]'
```

3. Remove the tmp files on the nodes in the 1st frame:

```
psh rack01 'rm -f /tmp/*'
```

Notice the use of `` to forward shell expansion. This is not necessary in interactive mode.

## See Also

noderange(3)|noderange.3, pscp(1)|pscp.1, pping(1)|pping.1, prsync(1)|prsync.1

## pushinitrd.1

### NAME

**pushinitrd** - queries your SoftLayer account and gets attributes for each server.

### SYNOPSIS

**pushinitrd** [-v | --verbose] [-w *waittime*] [*noderange*]

**pushinitrd** [-? | -h | --help]

### DESCRIPTION

The **pushinitrd** command copies the initrd, kernel, params, and static IP info to nodes, so they can be net installed even across vlans (w/o setting up pxe/dhcp broadcast relay). This assumes a working OS is on the nodes. Before running this command, you must run nodeset for these nodes. All of the nodes given to one invocation of **pushinitrd** must be using the same osimage.

Before using this command, it will be most convenient if you exchange the ssh keys using:

```
xdsh <noderange> -K
```

### OPTIONS

**-w** *waittime*

The number of seconds the initrd should wait before trying to communicate over the network. The default is 75. This translates into the netwait kernel parameter and is usually needed in a SoftLayer environment because it can take a while for a NIC to be active after changing state.

**-?|-h|--help**

Display usage message.

**-v|--version**

Command Version.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. Configure nodes for net installing in a SoftLayer environment:

```
pushinitrd <noderange>
```

## FILES

/opt/xcat/bin/pushinitrd

## SEE ALSO

getslnodes(1)|getslnodes.1

## rbeacon.1

## SYNOPSIS

**rbeacon** [-h | --help | -v | --version | -V | --verbose]

### BMC (using IPMI):

**rbeacon** *noderange* { **on** | **blink** | **off** | **stat** }

### OpenPOWER BMC (using IPMI):

**rbeacon** *noderange* { **on** | **blink** | **off** | **stat** }

### OpenPOWER OpenBMC:

**rbeacon** *noderange* { **on** | **off** | **stat** }

## DESCRIPTION

**rbeacon** Turns beacon (a light on the front and/or rear of the physical server) on/off/blink or gives status of a node or noderange.

## EXAMPLES

```
rbeacon 1,3 off
rbeacon 14-56,70-203 on
rbeacon 1,3,14-56,70-203 blink
rbeacon all,-129-256 stat
```

## SEE ALSO

noderange(3)|noderange.3, rpower(1)|rpower.1

## rbootseq.1

## SYNOPSIS

**rbootseq** [-h | --help | -v | --version]

### Blade specific:

**rbootseq** *noderange* {hd0 | hd1 | hd2 | hd3 | net | iscsi | iscsicrit | cdrom | usbflash | floppy | none | list | stat},...

### HP Blade specific:

**rbootseq** *noderange* {hd | net1 | net2 | net3 | net4 | cdrom | usbflash | floppy | none},...

### PPC (using Direct FSP Management) specific:

**rbootseq** *noderange* [hfi|net]

## DESCRIPTION

For Blade specific:

**rbootseq** sets the boot sequence (the order in which boot devices should be tried) for the specified blades. Up to four different medium/devices can be listed, separated by commas. The boot sequence will remain in effect for these blades until set differently.

For PPC (using Direct FSP Management) specific:

**rbootseq** sets the ethernet (net) or hfi device as the first boot device for the specified PPC LPARs. The **rbootseq** command requires that the ethernet or hfi mac address is stored in the mac table, and that the network information is correct in the networks table.

## **OPTIONS**

### **hd0 | harddisk0 | hd | harddisk**

The first hard disk.

### **hd1 | harddisk1**

The second hard disk.

### **hd2 | harddisk2**

The third hard disk.

### **hd3 | harddisk3**

The fourth hard disk.

### **n | net | network**

Boot over the ethernet network, using a PXE or BOOTP broadcast.

### **n | net | network | net1 | nic1 (HP Blade Only)**

Boot over the first ethernet network, using a PXE or BOOTP broadcast.

### **net2 | nic2 (HP Blade Only)**

Boot over the second ethernet network, using a PXE or BOOTP broadcast.

### **net3 | nic3 (HP Blade Only)**

Boot over the third ethernet network, using a PXE or BOOTP broadcast.

### **net3 | nic3 (HP Blade Only)**

Boot over the fourth ethernet network, using a PXE or BOOTP broadcast.

### **hfi**

Boot p775 nodes over the HFI network, using BOOTP broadcast.

### **iscsi**

Boot to an iSCSI disk over the network.

### **iscsicrit**

??

### **cd | cdrom**

The CD or DVD drive.

### **usbflash | usb | flash**

A USB flash drive.

### **floppy**

The floppy drive.

### **none**

If it gets to this part of the sequence, do not boot. Can not be specified 1st, or before any real boot devices.

### **list | stat**

Display the current boot sequence.



## EXAMPLES

1. Set blades 14-56 and 70-203 to try to boot first from the CD drive, then the floppy drive, then the network, and finally from the 1st hard disk:

```
rbootseq blade[14-56],blade[70-203] c,f,n,hd0
```

## SEE ALSO

rsetboot(1)|rsetboot.1

## rcons.1

### Name

**rcons** - remotely accesses the serial console of a node

### Synopsis

**rcons** *singlenode* [*conserver-host*] [-f] [-s]

**rcons** [-h | --help | -v | --version]

### Description

**rcons** provides access to a single remote node serial console, using the out-of-band infrastructure for the node (e.g. BMC, Management Module, HMC, KVM, etc.). It uses the consverger open source package to provide one read-write and multiple read-only instances of the console, plus console logging.

If *conserver-host* is specified, the consverger daemon on that host will be contacted, instead of on the local host.

To exit the console session, enter: 'ctrl-e c .' (3 characters: ctrl-e, 'c' and '.').

### Options

#### -f

If another console for this node is already open in read-write mode, force that console into read-only (spy) mode, and open this console in read-write mode. If -f is not specified, this console will be put in spy mode if another console is already open in read-write mode. The -f flag can not be used with the -s flag.

#### -s

Open the console in read-only (spy) mode, in this mode all the escape sequences work, but all other keyboard input is discarded. The -s flag can not be used with the -f flag.

#### -h | --help

Print help.

#### -v | --version

Print version.

## Files

**nodehm** table - xCAT node hardware management table. See `nodehm(5)`|`nodehm.5` for further details. This is used to determine the console access method.

## Examples

```
rcons node5
```

## See Also

`wcons(1)`|`wcons.1`

## regnotif.1

## NAME

**regnotif** - Registers a Perl module or a command that will get called when changes occur in the desired xCAT database tables.

## SYNOPSIS

**regnotif** [-h] --help]

**regnotif** [-v] --version]

**regnotif** *filename tablename[,tablename]...* [-o | --operation *actions*]

## DESCRIPTION

This command is used to register a Perl module or a command to the xCAT notification table. Once registered, the module or the command will get called when changes occur in the xCAT database tables indicated by *tablename*. The changes can be row addition, deletion and update which are specified by *actions*.

## PARAMETERS

*filename* is the path name of the Perl module or command to be registered. *tablename* is the name of the table that the user is interested in.

## OPTIONS

**-h | --help** Display usage message.

**-v | --version** Command Version.

**-V | --verbose** Verbose output.

**-o | --operation** specifies the database table actions that the user is interested in. It is a comma separated list. 'a' for row addition, 'd' for row deletion and 'u' for row update.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To register a Perl module that gets invoked when rows get added or deleted. in the nodelist and the nodehm tables, enter:

```
regnotif /opt/xcat/lib/perl/xCAT_monitoring/mycode.pm nodelist,nodehm -o a,d
```

2. To register a command that gets invoked when rows get updated in the switch table, enter:

```
regnotif /usr/bin/mycmd switch -o u
```

## FILES

/opt/xcat/bin/regnotif

## SEE ALSO

unregnotif(1)|unregnotif.1

## reenergy.1

## NAME

**reenergy** - remote energy management tool

## SYNOPSIS

**renergy** [-h | --help]

**renergy** [-v | --version]

**Power 6 server specific :**

**renergy** *noderange* [-V] {all | [savingstatus] [cappingstatus] [cappingmaxmin] [cappingvalue] [cappingsoftmin] [averageAC] [averageDC] [ambienttemp] [exhausttemp] [CPUspeed] [syssbpower] [sysIPLtime]}

**renergy** *noderange* [-V] {savingstatus={on | off} | cappingstatus={on | off} | cappingwatt=watt | cappingperc=percentage}

**Power 7 server specific :**

**renergy** *noderange* [-V] {all | [savingstatus] [dsavingstatus] [cappingstatus] [cappingmaxmin] [cappingvalue] [cappingsoftmin] [averageAC] [averageDC] [ambienttemp] [exhausttemp] [CPUspeed] [syssbpower] [sysIPLtime] [fsavingstatus] [ffoMin] [ffoVmin] [ffoTurbo] [ffoNorm] [ffovalue]}

**renergy** *noderange* [-V] {savingstatus={on | off} | dsavingstatus={on-norm | on-maxp | off} | fsavingstatus={on | off} | ffovalue=MHZ | cappingstatus={on | off} | cappingwatt=watt | cappingperc=percentage}

**Power 8 server specific :**

**renergy** *noderange* [-V] {all | [savingstatus] [dsavingstatus] [averageAC] [averageAChistory] [averageDC] [averageDChistory] [ambienttemp] [ambienttemphistory] [exhausttemp] [exhausttemphistory] [fanspeed] [fanspeedhistory] [CPUspeed] [CPUspeedhistory] [syssbpower] [sysIPLtime] [fsavingstatus] [ffoMin] [ffoVmin] [ffoTurbo] [ffoNorm] [ffovalue]}

**renergy** *noderange* [-V] {savingstatus={on | off} | dsavingstatus={on-norm | on-maxp | off} | fsavingstatus={on | off} | ffovalue=MHZ }

*NOTE:* The setting operation for **Power 8** server is only supported for the server which is running in PowerVM mode. Do NOT run the setting for the server which is running in OPAL mode.

**BladeCenter specific :**

**For Management Modules:**

**renergy** *noderange* [-V] {all | pd1all | pd2all | [pd1status] [pd2status] [pd1policy] [pd2policy] [pd1powermodule1] [pd1powermodule2] [pd2powermodule1] [pd2powermodule2] [pd1avaiaiblepower] [pd2avaiaiblepower] [pd1reservedpower] [pd2reservedpower] [pd1remainpower] [pd2remainpower] [pd1inusedpower] [pd2inusedpower] [availableDC] [averageAC] [thermaloutput] [ambienttemp] [mmtemp]}

**For a blade server nodes:**

**renergy** *noderange* [-V] {all | [averageDC] [capability] [cappingvalue] [CPUspeed] [maxCPUspeed] [savingstatus] [dsavingstatus]}

**renergy** *noderange* [-V] {savingstatus={on | off} | dsavingstatus={on-norm | on-maxp | off}}

**Flex specific :**

**For Flex Management Modules:**

**renergy** *noderange* [-V] {all | [powerstatus] [powerpolicy] [powermodule] [avaiaiblepower] [reservedpower] [remainpower] [inusedpower] [availableDC] [averageAC] [thermaloutput] [ambienttemp] [mmtemp]}

**For Flex node (power and x86):**

**renergy** *noderange* [-V] {all | [averageDC] [capability] [cappingvalue] [cappingmaxmin] [cappingmax] [cappingmin] [cappingGmin] [CPUspeed] [maxCPUspeed] [savingstatus] [dsavingstatus]}

```
renergy noderange [-V] {cappingstatus={on | off} | cappingwatt=watt | cappingperc=percentage | savingstatus={on | off} | dsavingstatus={on-norm | on-maxp | off}}
```

**iDataPlex specific :**

```
renergy noderange [-V] [{cappingmaxmin | cappingmax | cappingmin}] [cappingstatus] [cappingvalue] [relhistogram]
```

```
renergy noderange [-V] {cappingstatus={on | enable | off | disable} | {cappingwatt|cappingvalue}=watt}
```

**OpenPOWER server specific :**

```
renergy noderange {powerusage | temperature}
```

## DESCRIPTION

This **renergy** command can be used to manage the energy consumption of IBM servers which support IBM EnergyScale technology. Through this command, user can query and set the power saving and power capping status, and also can query the average consumed energy, the ambient and exhaust temperature, the processor frequency for a server.

**renergy** command supports IBM POWER6, POWER7 and POWER8 rack-mounted servers, BladeCenter management modules, blade servers, and iDataPlex servers. For *Power6* and *Power7* rack-mounted servers, the following specific hardware types are supported: *8203-E4A*, *8204-E8A*, *9125-F2A*, *8233-E8B*, *8236-E8C*. For *Power8* server, there's no hardware type restriction.

The parameter *noderange* needs to be specified for the **renergy** command to get the target servers. The *noderange* should be a list of CEC node names, blade management module node names or blade server node names. Lpar name is not acceptable here.

**renergy** command can accept multiple of energy attributes to query or one of energy attribute to set. If only the attribute name is specified, without the '=', **renergy** gets and displays the current value. Otherwise, if specifying the attribute with '=' like 'savingstatus=on', **renergy** will set the attribute savingstatus to value 'on'.

The attributes listed in the **SYNOPSIS** section are which ones can be handled by **renergy** command. But for each specific type of server, there are some attributes that are not supported. If user specifies an attribute which is not supported by a specific server, the return value of this attribute will be 'na'.

**Note:** the options *powerusage* and *temperature* are only supported for **OpenPOWER** servers.

The supported attributes for each specific system p hardware type is listed as follows:

### 8203-E4A, 8204-E8A

Supported attributes:

**Query:** savingstatus,cappingstatus,cappingmin,cappingmax, cappingvalue,cappingsoftmin,averageAC,averageDC,ambienttemp, exhausttemp,CPUspeed,sysbbpower,sysIPLtime

**Set:** savingstatus,cappingstatus,cappingwatt,cappingperc

### 9125-F2A

Supported attributes:

**Query:** savingstatus,averageAC,ambienttemp,exhausttemp, CPUspeed

**Set:** savingstatus

### 8233-E8B, 8236-E8C

Supported attributes:

**Query:** savingstatus,dsavingstatus,cappingstatus,cappingmin, cappingmax,cappingvalue,cappingsoftmin,averageAC,averageDC, ambienttemp,exhausttemp,CPUspeed,sysbbpower,sysIPLtime

**Set:** savingstatus,dsavingstatus,cappingstatus,cappingwatt, cappingperc

**9125-F2C, 9119-FHB**

Supported attributes:

**Query:** savingstatus,dsavingstatus,cappingstatus,cappingmin, cappingmax,cappingvalue,cappingsoftmin,averageAC,averageDC, ambienttemp,exhausttemp,CPUspeed,sysbbpower,sysIPLtime, fsavingstatus,ffoMin,ffoVmin,ffoTurbo,ffoNorm,ffovalue

**Set:** savingstatus,dsavingstatus,cappingstatus,cappingwatt, cappingperc,fsavingstatus,ffovalue

**Non of Above**

For the machine type which is not in the above list, the following attributes can be tried but not guaranteed:

**Query:** savingstatus,dsavingstatus,cappingstatus,cappingmin, cappingmax,,cappingvalue,cappingsoftmin,averageAC,averageDC, ambienttemp,exhausttemp,CPUspeed,sysbbpower,sysIPLtime

**Set:** savingstatus,dsavingstatus,cappingstatus,cappingwatt, cappingperc

Note: For system P CEC nodes, each query operation for attribute CPUspeed, averageAC or averageDC needs about 30 seconds to complete. The query for others attributes will get response immediately.

## PREREQUISITES

For the *Power6* and *Power7* nodes, the **renergy** command depends on the Energy Management Plugin **xCAT-pEnergy** to communicate with server. **xCAT-pEnergy** can be downloaded from the IBM web site: <http://www.ibm.com/support/fixcentral/>. (Other Software -> EM)

NOTE: *Power8* nodes don't need this specific energy management package.

For iDataPlex nodes, the **renergy** command depends on the Energy Management Plugin **xCAT-xEnergy** to communicate with server. This plugin must be requested from IBM.

(The support for BladeCenter energy management is built into base xCAT, so no additional plugins are needed for BladeCenter.)

## OPTIONS

**-h | --help**

Display the usage message.

**-v | --version**

Display the version information.

**-V**

Verbose output.

**all**

Query all energy attributes which supported by the specific type of hardware.

For *Power8* machines, will not display the attributes for historical records.

**pd1all**

Query all energy attributes of the power domain 1 for blade management module node.

**pd2all**

Query all energy attributes of the power domain 2 for blade management module node.

### **ambienttemp**

Query the current ambient temperature. (Unit is centigrade)

### **ambienttemphistory**

Query the historical records which were generated in last one hour for **ambienttemp**.

### **availableDC**

Query the total DC power available for the entire blade center chassis.

### **averageAC**

Query the average power consumed (Input). (Unit is watt)

Note: For 9125-F2A,9125-F2C server, the value of attribute averageAC is the aggregate for all of the servers in a rack.

Note: For Blade Center, the value of attribute averageAC is the total AC power being consumed by all modules in the chassis. It also includes power consumed by the Chassis Cooling Devices for BCH chassis.

### **averageAChistory**

Query the historical records which were generated in last one hour for **averageAC**.

### **averageDC**

Query the average power consumed (Output). (Unit is watt)

### **averageDChistory**

Query the historical records which were generated in last one hour for **averageDC**.

### **capability**

Query the Power Capabilities of the blade server.

staticPowerManagement: the module with the static worst case power values.

fixedPowermanagement: the module with the static power values but ability to throttle.

dynamicPowerManagement: the module with power meter capability, measurement enabled, but capping disabled.

dynamicPowerMeasurement1: the module with power meter capability, measurement enabled, phase 1 only

dynamicPowerMeasurement2: the module with power meter capability, measurement enabled, phase 2 or higher

dynamicPowerMeasurementWithPowerCapping: the module with power meter capability, both measurement and capping enabled, phase 2 or higher

### **cappingGmin**

Query the Guaranteed Minimum power capping value in watts.

### **cappingmax**

Query the Maximum of power capping value in watts.

### **cappingmaxmin**

Query the Maximum and Minimum of power capping value in watts.

### **cappingmin**

Query the Minimum of power capping value in watts.

**cappingperc=percentage**

Set the power capping value base on the percentage of the max-min of capping value which getting from *cappingmaxmin* attribute. The valid value must be from 0 to 100.

**cappingsoftmin**

Query the minimum value that can be assigned to power capping without guaranteed enforceability. (Unit is watt)

**cappingstatus**

Query the power capping status. The result should be 'on' or 'off'.

**cappingstatus={on | off}**

Set the power capping status. The value must be 'on' or 'off'. This is the switch to turn on or turn off the power capping function.

**cappingvalue**

Query the current power capping value. (Unit is watt)

**cappingwatt=watt**

Set the power capping value base on the watt unit.

If the 'watt' > maximum of *cappingmaxmin* or 'watt' < *cappingsoftmin*, the setting operation will be failed.

If the 'watt' > *cappingsoftmin* and 'watt' < minimum of *cappingmaxmin*, the value can NOT be guaranteed.

**CPUspeed**

Query the effective processor frequency. (Unit is MHz)

**CPUspeedhistory**

Query the historical records which were generated in last one hour for **CPUspeed**

**dsavingstatus**

Query the dynamic power saving status. The result should be 'on-norm', 'on-maxp' or 'off'.

If turning on the dynamic power saving, the processor frequency and voltage will be dropped dynamically based on the core utilization. It supports two modes for turn on state:

*on-norm* - means normal, the processor frequency cannot exceed the nominal value;

*on-maxp* - means maximum performance, the processor frequency can exceed the nominal value.

**dsavingstatus={on-norm | on-maxp | off}**

Set the dynamic power saving. The value must be 'on-norm', 'on-maxp' or 'off'.

The *dsavingstatus* setting operation needs about 2 minutes to take effect. (The used time depends on the hardware type)

The **dsavingstatus** only can be turned on when the **savingstatus** is in turn off status.

**exhausttemp**

Query the current exhaust temperature. (Unit is centigrade)

**exhausttemphistory**

Query the historical records which were generated in last one hour for **exhausttemp**

**fanspeed**



Query the fan speed for all the fans which installed in this node. (Unit is RPM - Rotations Per Minute))

If there are multiple fans for a node, multiple lines will be output. And a fan name in bracket will be appended after **fanspped** attribute name.

#### **fanspeedhistory**

Query the historical records which were generated in last one hour for **fanspeed**.

#### **ffoMin**

Query the minimum cpu frequency which can be set for FFO. (Fixed Frequency Override)

#### **ffoNorm**

Query the maximum cpu frequency which can be set for FFO.

#### **ffoTurbo**

Query the advertised maximum cpu frequency (selling point).

#### **ffoVmin**

Query the minimum cpu frequency which can be set for dropping down the voltage to save power. That means when you drop the cpu frequency from the ffoVmin to ffoVmin, the voltage won't change, then there's no obvious power to be saved.

#### **ffovalue**

Query the current value of FFO.

#### **ffovalue=MHZ**

Set the current value of FFO. The valid value of ffovalue should be between the ffoMin and ffoNorm.

Note1: Due to the limitation of firmware, the frequency in the range 3501 MHz - 3807 MHz can NOT be set to ffovalue. This range may be changed in future.

Note2: The setting will take effect only when the fsavingstatus is in 'on' status. But you need to set the ffovalue to a valid value before enabling the fsavingstatus. (It's a limitation of the initial firmware and will be fixed in future.)

The ffovalue setting operation needs about 1 minute to take effect.

#### **fsavingstatus**

Query the status of FFO. The result should be 'on' or 'off'. 'on' - enable; 'off' - disable.

#### **fsavingstatus={on | off}**

Set the status of FFO. The value must be 'on' or 'off'.

'on' - enable. It will take effect only when the **ffovalue** has been set to a valid value.

'off' -disable. It will take effect immediately.

Note: See the Note2 of ffovalue=MHZ.

#### **maxCPUspeed**

Query the maximum processor frequency. (Unit is MHz)

#### **mmtemp**

Query the current temperature of management module. (Unit is centigrade)

#### **pd1status | powerstatus**

Query the status of power domain 1 for blade management module node.

Note: for the attribute without the leading 'pd1' which means there's only one power domain in the chassis.

**pd1policy | powerpolicy**

Query the power management policy of power domain 1.

**pd1powermodule1 | powermodule**

Query the First Power Module capacity in power domain 1.

**pd1powermodule2 | powermodule**

Query the Second Power Module capacity in power domain 1.

**pd1avaipower | availablepower**

Query the total available power in power domain 1.

**pd1reservedpower | reservedpower**

Query the power that has been reserved for power domain 1.

**pd1remainpower | remainpower**

Query the remaining power available in power domain 1.

**pd1inusedpower | inusedpower**

Query the total power being used in power domain 1.

**pd2status**

Query the status of power domain 2 for blade management module node.

**pd2policy**

Query the power management policy of power domain 2.

**pd2powermodule1**

Query the First Power Module capacity in power domain 2.

**pd2powermodule2**

Query the Second Power Module capacity in power domain 2.

**pd2avaipower**

Query the total available power in power domain 2.

**pd2reservedpower**

Query the power that has been reserved for power domain 2.

**pd2remainpower**

Query the remaining power available in power domain 2.

**pd2inusedpower**

Query the total power being used in power domain 2.

**relhistogram**

Query histogram data for wattage information

**savingstatus**

Query the static power saving status. The result should be 'on' or 'off'. 'on' - enable; 'off' - disable.

**savingstatus**={ on | off }

Set the static power saving. The value must be 'on' or 'off'.

If turning on the static power saving, the processor frequency and voltage will be dropped to a fixed value to save energy.

The savingstatus setting operation needs about 2 minutes to take effect. (The used time depends on the hardware type)

The **savingstatus** only can be turned on when the **dsavingstatus** is in turn off status.

**sysIPLtime**

Query the time used from FSP standby to OS standby. (Unit is Second)

**sysbpower**

Query the system power consumed prior to power on. (Unit is Watt)

**thermaloutput**

Query the thermal output (load) in BTUs per hour for the blade center chassis.

**powerusage**

Query System Power Statistics with DCMI (Data Center Manageability Interface).

**temperature**

Query the temperature from DCMI (Data Center Manageability Interface) Temperature sensor. Currently, only CPU temperature and baseboard temperature sensor available for OpenPOWER servers.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. Query all attributes which CEC1,CEC2 supported.

```
renergy CEC1,CEC2 all
```

The output of the query operation:

```
CEC1: savingstatus: off
CEC1: dsavingstatus: off
CEC1: cappingstatus: off
CEC1: cappingmin: 1953 W
CEC1: cappingmax: 2358 W
CEC1: cappingvalue: 2000 W
CEC1: cappingsoftmin: 304 W
CEC1: averageAC: na
CEC1: averageDC: na
CEC1: ambienttemp: na
CEC1: exhausttemp: na
CEC1: CPUspeed: na
```

(continues on next page)

(continued from previous page)

```
CEC1: syssbpower: 40 W
CEC1: sysIPLtime: 900 S
CEC2: savingstatus: off
CEC2: cappingstatus: off
CEC2: cappingmin: 955 W
CEC2: cappingmax: 1093 W
CEC2: cappingvalue: 1000 W
CEC2: cappingsoftmin: 226 W
CEC2: averageAC: 627 W
CEC2: averageDC: 531 W
CEC2: ambienttemp: 25 C
CEC2: exhausttemp: 40 C
CEC2: CPUspeed: 4695 MHz
```

2. Query the **fanspeed** attribute for Power8 CEC.

```
renergy CEC1 fanspeed
```

The output of the query operation:

```
CEC1: fanspeed (Fan U78CB.001.WZS00MA-A1 00002101): 5947 RPM
CEC1: fanspeed (Fan U78CB.001.WZS00MA-A2 00002103): 6081 RPM
CEC1: fanspeed (Fan U78CB.001.WZS00MA-A3 00002105): 6108 RPM
CEC1: fanspeed (Fan U78CB.001.WZS00MA-A4 00002107): 6000 RPM
CEC1: fanspeed (Fan U78CB.001.WZS00MA-A5 00002109): 6013 RPM
CEC1: fanspeed (Fan U78CB.001.WZS00MA-A6 0000210B): 6013 RPM
CEC1: fanspeed (Fan U78CB.001.WZS00MA-E1 0000210C): 4992 RPM
CEC1: fanspeed (Fan U78CB.001.WZS00MA-E2 0000210D): 5016 RPM
```

3. Query the historical records for the **CPUspeed** attribute. (Power8 CEC)

```
renergy CEC1 CPUspeedhistory
```

The output of the query operation:

```
CEC1: CPUspeedhistory: 2027 MHZ: 20141226042900
CEC1: CPUspeedhistory: 2027 MHZ: 20141226042930
CEC1: CPUspeedhistory: 2244 MHZ: 20141226043000
CEC1: CPUspeedhistory: 2393 MHZ: 20141226043030
CEC1: CPUspeedhistory: 2393 MHZ: 20141226043100
CEC1: CPUspeedhistory: 2393 MHZ: 20141226043130
CEC1: CPUspeedhistory: 2393 MHZ: 20141226043200
CEC1: CPUspeedhistory: 2393 MHZ: 20141226043230
CEC1: CPUspeedhistory: 2393 MHZ: 20141226043300
CEC1: CPUspeedhistory: 2393 MHZ: 20141226043330
...
```

- 4 Query all the attributes for management module node MM1. (For chassis)

```
renergy MM1 all
```

The output of the query operation:

```
mm1: availableDC: 5880W
mm1: frontpaneltmp: 18.00 Centigrade
mm1: inusedAC: 2848W
mm1: mmtmp: 28.00 Centigrade
mm1: pd1avaiblepower: 2940W
mm1: pd1inusedpower: 848W
mm1: pd1policy: redundantWithoutPerformanceImpact
mm1: pd1powermodule1: Bay 1: 2940W
mm1: pd1powermodule2: Bay 2: 2940W
mm1: pd1remainpower: 1269W
mm1: pd1reservedpower: 1671W
mm1: pd1status: 1 - Power domain status is good.
mm1: pd2avaiblepower: 2940W
mm1: pd2inusedpower: 1490W
mm1: pd2policy: redundantWithoutPerformanceImpact
mm1: pd2powermodule1: Bay 3: 2940W
mm1: pd2powermodule2: Bay 4: 2940W
mm1: pd2remainpower: 51W
mm1: pd2reservedpower: 2889W
mm1: pd2status: 2 - Warning: Power redundancy does not exist in this power_
↪domain.
mm1: thermaloutput: 9717.376000 BTU/hour
```

5. Query all the attributes for blade server node blade1.

```
renergy blade1 all
```

The output of the query operation:

```
blade1: CPUSpeed: 4204MHZ
blade1: averageDC: 227W
blade1: capability: dynamicPowerMeasurement2
blade1: cappingvalue: 315W
blade1: dsavingstatus: off
blade1: maxCPUSpeed: 4204MHZ
blade1: savingstatus: off
```

6. Query the attributes savingstatus, cappingstatus and CPUSpeed for server CEC1.

```
renergy CEC1 savingstatus cappingstatus CPUSpeed
```

The output of the query operation:

```
CEC1: savingstatus: off
CEC1: cappingstatus: on
CEC1: CPUSpeed: 3621 MHz
```

7. Turn on the power saving function of CEC1.

```
renergy CEC1 savingstatus=on
```

The output of the setting operation:

```
CEC1: Set savingstatus succeeded.  
CEC1: This setting may need some minutes to take effect.
```

8. Set the power capping value base on the percentage of the max-min capping value. Here, set it to 50%.

```
renergy CEC1 cappingperc=50
```

If the maximum capping value of the CEC1 is 850w, and the minimum capping value of the CEC1 is 782w, the Power Capping value will be set as  $((850-782)*50\% + 782) = 816w$ .

The output of the setting operation:

```
CEC1: Set cappingperc succeeded.  
CEC1: cappingvalue: 816
```

9. Query powerusage and temperature for OpenPOWER servers.

```
renergy ops01 powerusage temperature
```

The output will be like this:

```
ops01: Current Power           : 591W  
ops01: Minimum Power over sampling duration : 558W  
ops01: Maximum Power over sampling duration : 607W  
ops01: Average Power over sampling duration : 572W  
ops01: Time Stamp              : 11/18/2015 - 1:4:1  
ops01: Statistics reporting time period    : 10000 milliseconds  
ops01: Power Measurement        : Active  
ops01: CPU Temperature Instance 0         : +39 Centigrade  
ops01: Baseboard temperature Instance 0    : +28 Centigrade
```

## REFERENCES

1. For more information on 'Power System Energy Management':  
<http://www-03.ibm.com/systems/power/software/energy/index.html>
2. EnergyScale white paper for Power6:  
<http://www-03.ibm.com/systems/power/hardware/whitepapers/energyscale.html>
3. EnergyScale white paper for Power7:  
<http://www-03.ibm.com/systems/power/hardware/whitepapers/energyscale7.html>

## FILES

/opt/xcat/bin/renergy

## replaycons.1

### NAME

**replaycons** - replay the console output for a node

### SYNOPSIS

**replaycons** *node log file* [*bps*] [*tail\_amount*]

**replaycons** [-h | --help | -v | --version]

### DESCRIPTION

The **replaycons** command reads the console log stored by conserver for this node, and displays it in a way that simulates the original output of the console. Using the *bps* value, it will throttle the speed of the output play back. (The logs are stored in /var/log/consoles.)

**replaycons** must be run locally on the system on which the console log is stored. This is normally that management node, but in a hierarchical cluster will usually be the service node.

### OPTIONS

*bps*

The display rate to use to play back the console output. Default is 19200.

*tail\_amount*

The place in the console log file to start play back, specified as the number of lines from the end.

**-v|--version**

Command Version.

**-h|--help**

Display usage message.

### RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To replay the console for node1 at the default rate, starting 2000 lines from the end:

```
replaycons node1.log 19200 2000
```

## FILES

/opt/xcat/bin/replaycons

## SEE ALSO

rcons(1)|rcons.1

## restartxcatd.1

## NAME

**restartxcatd** - Restart the xCAT daemon (xcatd).

## SYNOPSIS

**restartxcatd** [[-h | --help] | [-v | --version] | [-r | --reload]] [-V | --verbose]

## DESCRIPTION

The **restartxcatd** command restarts the xCAT daemon (xcatd).

### Linux Specific:

It will perform the xcatd *fast restart*. The xcatd *fast restart* is a specific restart which has two advantages compares to the *stop* and then *start*.

1. The interval of xcatd out of service is very short.
2. The in processing request which initiated by old xcatd will not be stopped by force. The old xcatd will hand over the sockets to new xcatd, but old xcatd will still be waiting for the in processing request to finish before the exit.

It does the same thing as **service xcatd restart** on NON-systemd enabled Operating System like rh6.x and sles11.x. But for the systemd enabled Operating System like rh7 and sles12, the **service xcatd restart** will just do the *stop* and *start* instead of xcatd *fast restart*.

It's recommended to use **restartxcatd** command to restart xcatd on systemd enabled system like rh7 and sles12 instead of **service xcatd restart** or **systemctl restart xcatd**.

### AIX Specific:

It runs **stopsrc -s xcatd** to stop xcatd first if xcatd is active, then runs **startsrc -s xcatd** to start xcatd.

If the xcatd subsystem was not created, **restartxcatd** will create it automatically.



## OPTIONS

- h|--help** Display usage message.
- v|--version** Command Version.
- r|--reload** On a Service Node, services will not be restarted.
- V|--verbose** Display the verbose messages.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To restart the xCAT daemon, enter:

```
restartxcatd
```

## FILES

/opt/xcat/sbin/restartxcatd

## restorexCATdb.1

## NAME

**restorexCATdb** - restores the xCAT db tables .

## SYNOPSIS

```
restorexCATdb [-a] [-V] [{-p | --path} path]
restorexCATdb [-b] [-V] [{-t | --timestamp} timestamp] [{-p | --path} path]
restorexCATdb [-h | --help] [-v | --version]
```

## DESCRIPTION

If not using binary restore(-b), the restorexCATdb command restores the xCAT database tables from the \*.csv files in directory given by the -p flag. The site table skiptables attribute can be set to a list of tables not to restore. It will not restore isnm\_perf\* tables. See man dumpxCATdb.

If using the binary restore option for DB2 or postgresQL, the entire database is restored from the binary backup made with dumpxCATdb. The database will be restored using the database Utilities. For DB2, the timestamp of the correct DB2 backup file (-t) must be provided. All applications accessing the DB2 database must be stopped before you can use the binary restore options. See the xCAT DB2 document for more information. For postgresQL, you do not have to stop the applications accessing the database and the complete path to the backup file, must be supplied on the -p flag.

## OPTIONS

**-h|--help** Display usage message.

**-v|--version** Command Version.

**-V|--verbose** Verbose.

**-a** All, without this flag the eventlog and auditlog will be skipped. These tables are skipped by default because restoring will generate new indexes

**-b** Restore from the binary image.

**-p|--path** Path to the directory containing the database restore files. If restoring from the binary image (-b) and using PostgreSQL, then this is the complete path to the restore file that was created with dumpxCATdb -b.

**-t|--timestamp** Use with the -b flag to designate the timestamp of the binary image to use to restore for DB2.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To restore the xCAT database from the /dbbackup/db directory, enter:

```
restorexCATdb -p /dbbackup/db
```

2. To restore the xCAT database including auditlog and eventlog from the /dbbackup/db directory, enter:

```
restorexCATdb -a -p /dbbackup/db
```

3. To restore the xCAT DB2 database from the binary image with timestamp 20111130130239 enter:

```
restorexCATdb -b -t 20111130130239 -p /dbbackup/db
```

4. To restore the xCAT PostgreSQL database from the binary image file pgbackup.20553 created by dumpxCATdb enter:

```
restorexCATdb -b -p /dbbackup/db/pgbackup.20553
```

## FILES

/opt/xcat/sbin/restorexCATdb

## SEE ALSO

dumpxCATdb(1)|dumpxCATdb.1

## reventlog.1

### Name

**reventlog** - retrieve or clear remote hardware event logs

### Synopsis

**reventlog** *noderange* [*number-of-entries* [-s]]|**all** [-s] | **clear**

**reventlog** [-h | --help | -v | --version]

### OpenPOWER OpenBMC specific :

**reventlog** *noderange* [**resolved**=*{id-list|LED}*]

### Description

**reventlog** can display any number of remote hardware event log entries or clear them for a range of nodes. Hardware event logs are stored on each servers service processor.

### Options

*number-of-entries*

Retrieve the specified number of entries from the nodes' service processors.

**all**

Retrieve all entries.

**-s**

To sort the entries from latest (always the last entry in event DB) to oldest (always the first entry in event DB). If **number-of-entries** specified, the latest **number-of-entries** events will be output in the order of latest to oldest.

**clear**

Clear event logs.

**resolved**=*{id-list|LED}*

Mark event log entries as resolved. Use comma separated list of entry ids to specify individual entries. Use **LED** to mark as resolved all event log entries that contribute to LED fault.

**-h | --help**

Print help.

**-v | --version**

Print version.

## Examples

1. List last 5 event log entries from node4 and node5

```
reventlog node4,node5 5
```

Output is similar to:

```
node4: SERVPROC I 09/06/00 15:23:33 Remote Login Successful User ID =  
↳USERID[00]  
node4: SERVPROC I 09/06/00 15:23:32 System spn1 started a RS485 connection.  
↳with us[00]  
node4: SERVPROC I 09/06/00 15:22:35 RS485 connection to system spn1 has  
↳ended[00]  
node4: SERVPROC I 09/06/00 15:22:32 Remote Login Successful User ID =  
↳USERID[00]  
node4: SERVPROC I 09/06/00 15:22:31 System spn1 started a RS485 connection.  
↳with us[00]  
node5: SERVPROC I 09/06/00 15:22:32 Remote Login Successful User ID =  
↳USERID[00]  
node5: SERVPROC I 09/06/00 15:22:31 System spn1 started a RS485 connection.  
↳with us[00]  
node5: SERVPROC I 09/06/00 15:21:34 RS485 connection to system spn1 has  
↳ended[00]  
node5: SERVPROC I 09/06/00 15:21:30 Remote Login Successful User ID =  
↳USERID[00]  
node5: SERVPROC I 09/06/00 15:21:29 System spn1 started a RS485 connection.  
↳with us[00]
```

2. Clear all event log entries from node4 and node5

```
reventlog node4,node5 clear
```

Output is similar to:

```
node4: clear  
node5: clear
```

3. Mark as resolved all event log entries from node4 that contribute to LED fault

```
reventlog node4 resolved=LED
```

Output is similar to:

```
Attempting to resolve the following log entries: LED...  
node4: Resolved 51.  
node4: Resolved 52.  
node4: Resolved 58.
```

## SEE ALSO

rpower(1)|rpower.1, monstart(1)|monstart.1

## rflash.1

### Name

**rflash** - Performs Licensed Internal Code (LIC) update or firmware update on supported xCAT managed nodes.

### Synopsis

**rflash** [-h | --help | -v | --version | -V | --verbose]

### PPC (with HMC) specific:

**rflash** *noderange* -p *directory* [--activate {concurrent | disruptive}]

**rflash** *noderange* [--commit | --recover]

### PPC (without HMC, using Direct FSP Management) specific:

**rflash** *noderange* -p *directory* [--activate {disruptive | deferred}] [-d *data\_directory*]

**rflash** *noderange* [--commit | --recover]

### NeXtScale FPC specific:

**rflash** *noderange* *http\_directory*

### OpenPOWER BMC specific (using IPMI):

**rflash** *noderange* [*hpm\_file\_path* | -d *data\_directory*] [-c | --check] [--retry=*count*]

**rflash** *noderange* --recover *bmc\_file\_path*

### OpenPOWER OpenBMC specific :

**rflash** *noderange* {[-c | --check] | [-l | --list]}

**rflash** *noderange* *tar\_file\_path* {[-c | --check] | [-a | --activate] | [-u | --upload]}

**rflash** *noderange* *tar\_file\_directory* [-d]

**rflash** *noderange* *image\_id* {[-a | --activate] | [--delete]}

## Description

The **rflash** command initiates Firmware updates on supported xCAT nodes. Licensed Internal Code (also known as microcode) updates are performed on supported HMC-attached POWER5 and POWER6 pSeries nodes, and POWER7 systems using Direct FSP management.

The command scans the specified directory structure for Firmware update package files applicable to the given nodes and components. And then it will **automatically** select the **latest** version for the upgrade. The firmware update files include the Microcode update package and associated XML file. They can be downloaded from the IBM Web site: <http://www-933.ibm.com/support/fixcentral/>.

The POWER5 and POWER6 systems contain several components that use Licensed Internal Code. The **rflash** command supports two of these components: the managed system (also known as the Central Electronics Complex, or CEC) and the power subsystem (also known as the Bulk Power Assembly (BPA) or Bulk Power Controller (BPC)). Some POWER5 managed systems can be attached to a power subsystem. These power subsystems can support multiple managed systems. When the **rflash** command is invoked, xCAT will determine the managed system or power subsystem associated with that CEC and perform the update.

The *noderange* can be an CEC or CEC list, a Lpar or Lpar list and a Frame or Frame list. But CEC (or Lpar) and Frame **can't** be used at the same time. When the *noderange* is an CEC or CEC list, **rflash** will upgrade the firmware of the CEC or CECs in the cec list. If *noderange* is a Lpar or Lpar list, **rflash** will update Licensed Internal Code (LIC) on HMC-attached POWER5 and POWER6 pSeries nodes, and POWER7 systems using Direct FSP management. If *noderange* is a Frame or Frame list, **rflash** will update Licensed Internal Code (LIC) of the power subsystem on HMC-attached POWER5 and POWER6 pSeries nodes. The *noderange* can also be the specified node groups. You can specify a comma or space-separated list of node group ranges. See the *noderange* man page for detailed usage information.

The command will update firmware for NeXtScale FPC when given an FPC node and the http information needed to access the firmware.

### PPC (with HMC) specific:

The **rflash** command uses the **xdsh** command to connect to the HMC controlling the given managed system and perform the updates. Before running **rflash**, use **rspconfig** to check if the related HMC ssh is enabled. To enable a HMC ssh connection, use **rspconfig** command.

**Warning!** This command may take considerable time to complete, depending on the number of systems being updated and the workload on the target HMC. In particular, power subsystem updates may take an hour or more if there are many attached managed systems.

Depending on the Licensed Internal Code update that is installed, the affected HMC-attached POWER5 and POWER6 systems may need to be recycled. The **--activate** flag determines how the affected systems activate the new code. The concurrent option activates code updates that do not require a system recycle (known as a “concurrent update”). If this option is given with an update that requires a system recycle (known as a “disruptive update”), a message will be returned, and no activation will be performed. The disruptive option will cause any affected systems that are powered on to be powered down before installing and activating the update. Once the update is complete, the command will attempt to power on any affected systems that it powered down. Those systems that were powered down when the command was issued will remain powered down when the update is complete.

The flash chip of a POWER5 and POWER6 managed system or power subsystem stores firmware in two locations, referred to as the temporary side and the permanent side. By default, most POWER5 and POWER6 systems boot from the temporary side of the flash. When the **rflash** command updates code, the current contents of the temporary side are written to the permanent side, and the new code is written to the temporary side. The new code is then activated. Therefore, the two sides of the flash will contain different levels of code when the update has completed.

The **--commit** flag is used to write the contents of the temporary side of the flash to the permanent side. This flag should be used after updating code and verifying correct system operation. The **--recover** flag is used to write the

permanent side of the flash chip back to the temporary side. This flag should be used to recover from a corrupt flash operation, so that the previously running code can be restored.

**NOTE:** When the **--commit** or **--recover** two flags is used, the noderange **cannot** be BPA. It only **can** be CEC or LPAR, and will take effect for **both** managed systems and power subsystems.

xCAT recommends that you shutdown your Operating System images and power off your managed systems before applying disruptive updates to managed systems or power subsystems.

Any previously activated code on the affected systems will be automatically accepted into permanent flash by this procedure.

**IMPORTANT!** If the power subsystem is recycled, all of its attached managed systems will be recycled.

If it outputs **“Timeout waiting for prompt”** during the upgrade, set the **“ppctimeout”** larger in the **site** table. After the upgrade, remember to change it back. If run the **“rflash”** command on an AIX management node, need to make sure the value of **“useSSHonAIX”** is **“yes”** in the site table.

### PPC (using Direct FSP Management) specific:

In currently Direct FSP/BPA Management, our **rflash** doesn't support **concurrent** value of **--activate** flag, and supports **disruptive** and **deferred**. The **disruptive** option will cause any affected systems that are powered on to be powered down before installing and activating the update. So we require that the systems should be powered off before do the firmware update.

The **deferred** option will load the new firmware into the T (temp) side, but will not activate it like the disruptive firmware. The customer will continue to run the Frames and CECs working with the P (perm) side and can wait for a maintenance window where they can activate and boot the Frame/CECs with new firmware levels. Refer to the doc to get more details: [XCAT\\_Power\\_775\\_Hardware\\_Management](#)

In Direct FSP/BPA Management, there is **-d data\_directory** option. The default value is /tmp. When doing firmware update, **rflash** will put some related data from rpm packages in <data\_directory> directory, so the execution of **rflash** will require available disk space in <data\_directory> for the command to properly execute:

For one GFW rpm package and one power code rpm package, if the GFW rpm package size is gfw\_rpmsize, and the Power code rpm package size is power\_rpmsize, it requires that the available disk space should be more than:  $1.5 * \text{gfw\_rpmsize} + 1.5 * \text{power\_rpmsize}$

For Power 775, the **rflash** command takes effect on the primary and secondary FSPs or BPAs almost in parallel.

For more details about the Firmware Update using Direct FSP/BPA Management, refer to: [XCAT\\_Power\\_775\\_Hardware\\_Management#Updating\\_the\\_BPA\\_and\\_FSP\\_firmware\\_using\\_xCAT\\_DFM](#)

### NeXtScale FPC specific:

The command will update firmware for NeXtScale FPC when given an FPC node and the http information needed to access the firmware. The http information required includes both the MN IP address as well as the directory containing the firmware. It is recommended that the firmware be downloaded and placed in the /install directory structure as the xCAT MN /install directory is configured with the correct permissions for http. Refer to the doc to get more details: [XCAT\\_NeXtScale\\_Clusters](#)

### **OpenPOWER specific (using IPMI):**

The command will update firmware for OpenPOWER BMC when given an OpenPOWER node with *mgt=ipmi* and either the hpm formatted file path or path to a data directory.

**Note:** When using **rflash** in hierarchical environment, the hpm file or data directory must be accessible from Service Nodes.

### **OpenPOWER specific (using OpenBMC):**

The command will update firmware for OpenPOWER BMC when given an OpenPOWER node with *mgt=openbmc* and either an update .tar file or an uploaded image id.

#### **-l|--list:**

The list option will list out available firmware on the BMC. It provides an interface to display the ID of the various firmware levels.

The (\*) symbol indicates the active running firmware on the server.

The (+) symbol indicates the firmware that is pending and a reboot is required to set it to be the active running firmware level.

#### **-u|--upload:**

The upload option expects a .tar file as the input and will upload the file to the BMC. Use the list option to view the result.

#### **-a|--activate:**

The activate option expects either a .tar file or an ID as the input. If a .tar file is provided, it will upload and activate the firmware in a single step

To apply the firmware level, a reboot is required to BMC and HOST.

**Note:** When using **rflash** in hierarchical environment, the .tar file must be accessible from Service Nodes.

#### **-d:**

This option streamlines the update, activate, reboot BMC and reboot HOST procedure. It expects a directory containing both BMC and Host .tar files. When BMC and Host tar files are provided, the command will upload and activate firmware. After BMC becomes activate, it will reboot BMC. If BMC state is Ready, the command will reboot the HOST. If BMC state is NotReady, the command will exit.

#### **--delete:**

This delete option will delete update image from BMC. It expects an ID as the input.

## **Options**

#### **-h|--help**

Writes the command's usage statement to standard output.

#### **-c|--check**

Check the firmware version of BMC and an update file.

#### **-p *directory***

Specifies the directory where the packages are located.



**-d** *data\_directory*

PPC (without HMC, using Direct FSP Management) specific:

Specifies the directory where the raw data from rpm packages for each CEC/Frame are located. The default directory is /tmp. The option is only used in Direct FSP/BPA Management.

OpenPOWER BMC specific (using IPMI):

Used for IBM Power S822LC for Big Data systems only. Specifies the directory where the **pUpdate** utility and at least one of BMC or Host update files are located. The utility and update files can be downloaded from FixCentral.

**--activate {concurrent | disruptive}**

Must be specified to activate the new Licensed Internal Code. The “disruptive” option will cause the target systems to be recycled. Without this flag, LIC updates will be installed only, not activated.

**--commit**

Used to commit the flash image in the temporary side of the chip to the permanent side for both managed systems and power subsystems.

**--recover**

PPC (with HMC) and PPC (without HMC, using Direct FSP Management) specific:

Used to recover the flash image in the permanent side of the chip to the temporary side for both managed systems and power subsystems.

OpenPOWER BMC specific (using IPMI):

Used for IBM Power S822LC for Big Data systems only. Used to recover the BMC with a BMC image downloaded from FixCentral. This option will only work if BMC is in “Brick protection” state.

**--retry=*count***

Specify number of times to retry the update if failure is detected. Default value is 2. Value of 0 can be used to indicate no retries.

**-a|--activate**

Activate update image. Image id or update file must be specified.

**-l|--list**

List currently uploaded update images. “(\*)” indicates currently active image.

**-u|--upload**

Upload update image. Specified file must be in .tar format.

**--delete**

Delete update image from BMC

**-v|--version**

Displays the command’s version.

**-V|--verbose**

Verbose output.

## Exit Status

- 0 The command completed successfully.
- 1 An error has occurred.

## Examples

1. To update only the power subsystem attached to a single HMC-attached pSeries CEC(cec\_name), and recycle the power subsystem and all attached managed systems when the update is complete, and the Microcode update package and associated XML file are in /tmp/fw, enter:

```
rflash cec_name -p /tmp/fw --activate disruptive
```

2. To update only the power subsystem attached to a single HMC-attached pSeries node, and recycle the power subsystem and all attached managed systems when the update is complete, and the Microcode update package and associated XML file are in /tmp/fw, enter:

```
rflash bpa_name -p /tmp/fw --activate disruptive
```

3. To commit a firmware update to permanent flash for both managed system and the related power subsystems, enter:

```
rflash cec_name --commit
```

4. To update the firmware on a NeXtScale FPC specify the FPC node name and the HTTP location of the file including the xCAT MN IP address and the directory on the xCAT MN containing the firmware as follows:

```
rflash fpc01 http://10.1.147.169/install/firmware/fhet17a/ibm_fw_fpc_fhet17a-2.
→02_anyos_noarch.rom
```

5. To update the firmware on OpenPOWER machine specify the node name and the file path of the HPM firmware file as follows:

```
rflash fs3 /firmware/8335_810.1543.20151021b_update.hpm
```

Print verbose message to rflash log file (/var/log/xcata/rflash/fs3.log) when updating firmware:

```
rflash fs3 /firmware/8335_810.1543.20151021b_update.hpm -V
```

6. To update the firmware on IBM Power S822LC for Big Data machine specify the node name and the file path of the data directory containing pUpdate utility, both BMC and Host update files:

```
rflash briggs01 -d /root/supermicro/OP825
```

7. To update the firmware on the OpenBMC machine, specify the firmware update file to upload and activate:

```
rflash p9euh02 -a /tmp/witherspoon.pnor.squashfs.tar
```

## Location

`/opt/xcat/bin/rflash`

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

`rinv(1)`|`rinv.1`, `rspconfig(1)`|`rspconfig.1`

### **rinv.1**

## Name

**rinv** - Remote hardware inventory

## Synopsis

**rinv** [**-h** | **--help** | **-v** | **--version**]

### BMC/MPA specific:

**rinv** *noderange* [**model** | **serial** | **asset** | **vpd** | **deviceid** | **guid** | **firm** | **dimm** | **mprom** | **all**]

### OpenPOWER (IPMI) server specific:

**rinv** *noderange* [**model** | **serial** | **deviceid** | **uuid** | **guid** | **vpd** | **mprom** | **firm** | **all**]

### OpenPOWER (OpenBMC) server specific:

**rinv** *noderange* [**model**][**serial**][**firm**][**cpu**][**dimm**][**all**] [**-V** | **--verbose**]

### PPC (with HMC) specific:

**rinv** *noderange* {**bus** | **config** | **serial** | **model** | **firm** | **all**}

### PPC (using Direct FSP Management) specific:

**rinv** *noderange* {**firm**}

**rinv** *noderange* {**deconfig** [-x]}

### Blade specific:

**rinv** *noderange* {**mtm** | **serial** | **mac** | **bios** | **diag** | **mprom** | **mparom** | **firm** | **all**}

### VMware specific:

**rinv** *noderange* [-t]

### pdu specific:

**rinv** *noderange*

### zVM specific:

**rinv** *noderange* [**config** | **all**]

**rinv** *noderange* [--diskpoolspace]

**rinv** *noderange* [--diskpool *pool space*]

**rinv** *noderange* [--fcpdevices *state details*]

**rinv** *noderange* [--diskpoolnames]

**rinv** *noderange* [--networknames]

**rinv** *noderange* [--network *name*]

**rinv** *noderange* [--ssi]

**rinv** *noderange* [--smapilevel]

**rinv** *noderange* [--wwpns *fc\_channel*]

**rinv** *noderange* [--zfcppool *pool space*]

**rinv** *noderange* [--zfcppoolnames]

## Description

**rinv** retrieves hardware configuration information from the on-board Service Processor for a single or range of nodes and groups.

Calling **rinv** for VMware will display the UUID/GUID, number of CPUs, amount of memory, the MAC address and a list of Hard disks. The output for each Hard disk includes the label, size and backing file location.

## Options

### **bus**

List all buses for each I/O slot.

### **config**

Retrieves number of processors, speed, total memory, and DIMM locations.

### **model**

Retrieves model number.

### **serial**

Retrieves serial number.

### **firm**

Retrieves firmware versions.

### **deconfig**

Retrieves deconfigured resources. Deconfigured resources are hw components (cpus, memory, etc.) that have failed so the firmware has automatically turned those components off. This option is only capable of listing some of the deconfigured resources and should not be the only method used to check the hardware status.

### **-x**

To output the raw information of deconfigured resources for CEC.

### **asset**

Retrieves asset tag. Usually it's the MAC address of eth0.

### **vpd**

Same as specifying model, serial, deviceid, and mprom.

### **diag**

Diagnostics information of firmware.

### **mprom**

Retrieves mprom firmware level.

### **dimmm**

Retrieves dual in-line memory module information.

### **deviceid**

Retrieves device identification. Usually device, manufacturing and product IDs.

### **uuid**

Retrieves the universally unique identifier.

### **guid**

Retrieves the global unique identifier .

### **all**

All of the above.

### **-h | --help**

Print help.

**-v | --version**

Print version.

**-V | --verbose**

Prints verbose output, if available.

**-t**

Set the values in the vm table to what vCenter has for the indicated nodes.

**zVM specific :**

**--diskpoolspace**

Calculates the total size of every known storage pool.

**--diskpool *pool space***

Lists the storage devices (ECKD and FBA) contained in a disk pool. Space can be: all, free, or used.

**--fcpdevices *state details***

Lists the FCP device channels that are active, free, or offline. State can be: active, free, or offline.

**--diskpoolnames**

Lists the known disk pool names.

**--networknames**

Lists the known network names.

**--network *name***

Shows the configuration of a given network device.

**--ssi**

Obtain the SSI and system status.

**--smapilevel**

Obtain the SMAPI level installed on the z/VM system.

**--wwpns *fc channel***

Query a given FCP device channel on a z/VM system and return a list of WWPNS.

**--zfcpool *pool space***

List the SCSI/FCP devices contained in a zFCP pool. Space can be: free or used.

**--zfcpoolnames**

List the known zFCP pool names.

## Examples

1. To retrieve all information available from blade node4, enter:

```
rinv node5 all
```

Output is similar to:

```
node5: Machine Type/Model 865431Z
node5: Serial Number 23C5030
node5: Asset Tag 00:06:29:1F:01:1A
node5: PCI Information
node5:  Bus  VendID  DevID   RevID  Description              Slot Pass/Fail
node5:  0    1166    0009    06    Host Bridge              0    PASS
node5:  0    1166    0009    06    Host Bridge              0    PASS
node5:  0    5333    8A22    04    VGA Compatible Controller0    PASS
node5:  0    8086    1229    08    Ethernet Controller      0    PASS
node5:  0    8086    1229    08    Ethernet Controller      0    PASS
node5:  0    1166    0200    50    ISA Bridge               0    PASS
node5:  0    1166    0211    00    IDE Controller           0    PASS
node5:  0    1166    0220    04    Universal Serial Bus     0    PASS
node5:  1    9005    008F    02    SCSI Bus Controller      0    PASS
node5:  1    14C1    8043    03    Unknown Device Type      2    PASS
node5: Machine Configuration Info
node5: Number of Processors:
node5: Processor Speed: 866 MHz
node5: Total Memory: 512 MB
node5: Memory DIMM locations: Slot(s) 3 4
```

2. To output the raw information of deconfigured resources for CEC cec01, enter:

```
rinv cec01 deconfig -x
```

Output is similar to:

```
cec01:
<SYSTEM>
<System_type>IH</System_type>
<NODE>
<Location_code>U78A9.001.0123456-P1</Location_code>
<RID>800</RID>
</NODE>
</SYSTEM>
```

3. To retrieve 'config' information from the HMC-managed LPAR node3, enter:

```
rinv node3 config
```

Output is similar to:

```
node5: Machine Configuration Info
node5: Number of Processors: 1
node5: Total Memory (MB): 1024
```

4. To retrieve information about a VMware node vm1, enter:

```
rinv vm1
```

Output is similar to:

```
vm1: UUID/GUID: 42198f65-d579-fb26-8de7-3ae49e1790a7
vm1: CPUs: 1
vm1: Memory: 1536 MB
vm1: Network adapter 1: 36:1b:c2:6e:04:02
vm1: Hard disk 1 (d0): 9000 MB @ [nfs_192.168.68.21_vol_rc1storage_vmware] vm1_
↳ 3/vm1.vmdk
vm1: Hard disk 2 (d4): 64000 MB @ [nfs_192.168.68.21_vol_rc1storage_vmware]_
↳ vm1_3/vm1_5.vmdk
```

#### zVM specific :

5. To list the defined network names available for a given node:

```
rinv pokdev61 --getnetworknames
```

Output is similar to:

```
pokdev61: LAN:QDIO SYSTEM GLAN1
pokdev61: LAN:HIPERS SYSTEM GLAN2
pokdev61: LAN:QDIO SYSTEM GLAN3
pokdev61: VSWITCH SYSTEM VLANTST1
pokdev61: VSWITCH SYSTEM VLANTST2
pokdev61: VSWITCH SYSTEM VSW1
pokdev61: VSWITCH SYSTEM VSW2
pokdev61: VSWITCH SYSTEM VSW3
```

6. To list the configuration for a given network:

```
rinv pokdev61 --getnetwork GLAN1
```

Output is similar to:

```
pokdev61: LAN SYSTEM GLAN1          Type: QDIO    Connected: 1    Maxconn:_
↳ INFINITE
pokdev61:  PERSISTENT UNRESTRICTED IP                      Accounting: OFF
pokdev61:  IPTimeout: 5                      MAC Protection: Unspecified
pokdev61:  Isolation Status: OFF
```

7. To list the disk pool names available:

```
rinv pokdev61 --diskpoolnames
```

Output is similar to:

```
pokdev61: POOL1
pokdev61: POOL2
pokdev61: POOL3
```

8. List the configuration for a given disk pool:



```
rinv pokdev61 --diskpool POOL1 free
```

Output is similar to:

```
pokdev61: #VolID DevType StartAddr Size
pokdev61: EMC2C4 3390-09 0001 10016
pokdev61: EMC2C5 3390-09 0001 10016
```

9. List the known zFCP pool names.

```
rinv pokdev61 --zfcppoolnames
```

Output is similar to:

```
pokdev61: zfcpl
pokdev61: zfcpl
pokdev61: zfcpl
```

10. List the SCSI/FCP devices contained in a given zFCP pool:

```
rinv pokdev61 --zfcppool zfcpl
```

Output is similar to:

```
pokdev61: #status,wwpn,lun,size,range,owner,channel,tag
pokdev61: used,500512345678c411,4014412100000000,2g,3B40-3B7F,ihost13,3b77,
pokdev61: used,500512345678c411,4014412200000000,8192M,3B40-3B7F,ihost13,3b77,
↳ replace_root_device
pokdev61: free,500512345678c411,4014412300000000,8g,3B40-3B7F,,,
pokdev61: free,5005123456789411,4014412400000000,2g,3B40-3B7F,,,
pokdev61: free,5005123456789411;5005123456789411,4014412600000000,2G,3B40-3B7F,
↳ , ,
```

## SEE ALSO

rpower(1)|rpower.1

## rmdef.1

## NAME

**rmdef** - Use this command to remove xCAT data object definitions.

## SYNOPSIS

**rmdef** [-h | --help] [-t *object-types*]

**rmdef** [-V | --verbose] [-a | --all] [-t *object-types*] [-o *object-names*] [-f | --force] [-C | --cleanup] [*noderange*]

## DESCRIPTION

This command is used to remove xCAT object definitions that are stored in the xCAT database.

## OPTIONS

### **-a|--all**

Clear the whole xCAT database. A backup of the xCAT definitions should be saved before using this option as the xCAT daemons will no longer work once cleared.

To restore:

1. **export XCATBYPASS=1** and run the **restorexCATdb** command.
- or
2. Run **xcatconfig -d** which initializes the database the same as when xCAT was installed.

### **-f|--force**

Use this with the **--all** option as an extra indicator that ALL definitions are to be removed.

### **-h|--help**

Display a usage message.

### *noderange*

A set of comma delimited node names and/or group names. See the “noderange” man page for details on supported formats.

### **-o** *object-names*

A set of comma delimited object names.

### **-t** *object-types*

A set of comma delimited object types.

### **-C|--cleanup**

Perform additional cleanup by running **nodeset offline**, **makeconservrcf -d** and **makegocons --cleanup** on the objects specified in the *noderange*.

### **-V|--verbose**

Verbose mode.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To remove a range of node definitions.

```
rmdef -t node node1-node4
```

2. To remove all node definitions for the nodes contained in the group bpcnodes.

```
rmdef -t node -o bpcnodes
```

3. To remove the group called bpcnodes.

```
rmdef -t group -o bpcnodes
```

(This will also update the values of the “groups” attribute of the member nodes.)

## FILES

\$XCATROOT/bin/rmdef

(The XCATROOT environment variable is set when xCAT is installed. The default value is “/opt/xcat”.)

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

mkdef(1)|mkdef.1, lsdef(1)|lsdef.1, chdef(1)|chdef.1, xcatstanzafile(5)|xcatstanzafile.5

## rmdisknode.1

## NAME

**rmdisknode** - Use this xCAT command to remove AIX/NIM diskless machine definitions.

## SYNOPSIS

**rmnsklsnode** [-h | --help ]

**rmnsklsnode** [-V|--verbose] [-f|--force] [-r|--remdef] [-i *image\_name*] [-p|--primarySN] [-b|--backupSN] *noderange*

## DESCRIPTION

Use this command to remove all NIM client machine definitions that were created for the specified xCAT nodes.

The xCAT node definitions will not be removed. Use the xCAT **rmdef** command to remove xCAT node definitions.

If you are using xCAT service nodes the **rmnsklsnode** command will automatically determine the correct server(s) for the node and remove the NIM definitions on that server(s).

If the node you are trying to remove is currently running the **rmnsklsnode** command will not remove the definitions. You can use the “-f” option to shut down the node and remove the definition.

### Removing alternate NIM client definitions

If you used the “-n” option when you created the NIM client definitions with the **mknsklsnode** command then the NIM client machine names would be a combination of the xCAT node name and the osimage name used to initialize the NIM machine. To remove these definitions, you must provide the name of the osimage that was used using the “-i” option.

In most cases you would most likely want to remove the old client definitions without disturbing the nodes that you just booted with the new alternate client definition. The **rmnsklsnode -r** option can be used to remove the old alternate client definitions without stopping the running node.

However, if you have NIM dump resources assign to your nodes be aware that when the old NIM alternate client definitions are removed it will leave the nodes unable to produce a system dump. This is a current limitation in the NIM support for alternate client definitions. For this reason, it is recommended that you wait to do this cleanup until right before you do your next upgrade.

## OPTIONS

### -f|--force

Use the force option to stop and remove running nodes. This handles the situation where a NIM machine definition indicates that a node is still running even though it is not.

### -b|--backupSN

When using backup service nodes only update the backup. The default is to update both the primary and backup service nodes.

### -h|--help

Display usage message.

### -i *image\_name*

The name of an xCAT image definition.

### *noderange*

A set of comma delimited node names and/or group names. See the “noderange” man page for details on additional supported formats.

### -p|--primarySN

When using backup service nodes only update the primary. The default is to update both the primary and backup service nodes.

### **-r|--remdef**

Use this option to reset, deallocate, and remove NIM client definitions. This option will not attempt to shut down running nodes. This option should be used when remove alternate NIM client definitions that were created using **mkdsklsnode -n**.

### **-V |--verbose**

Verbose mode.

## **RETURN VALUE**

0 The command completed successfully.

1 An error has occurred.

## **EXAMPLES**

- 1) Remove the NIM client definition for the xCAT node named “node01”. Give verbose output.

```
rmdsklsnode -V node01
```

- 2) Remove the NIM client definitions for all the xCAT nodes in the group “aixnodes”. Attempt to shut down the nodes if they are running.

```
rmdsklsnode -f aixnodes
```

- 3) Remove the NIM client machine definition for xCAT node “node02” that was created with the **mkdsklsnode -n** option and the image “AIXdskls”. (i.e. NIM client machine name “node02\_AIXdskls”).

```
rmdsklsnode -i AIXdskls node02
```

This assume that node02 is not currently running.

- 4) Remove the old alternate client definition “node27\_olddskls”.

```
rmdsklsnode -r -i olddskls node27
```

Assuming the node was booted using an new alternate NIM client definition then this will leave the node running.

## **FILES**

/opt/xcat/bin/rmdsklsnode

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

`mkdsklsnode(1)`|`mkdsklsnode.1`

## `rmflexnode.1`

## NAME

**rmflexnode** - Delete a flexible node.

## SYNOPSIS

**rmflexnode** [-h | --help]

**rmflexnode** [-v | --version]

**rmflexnode** *noderange*

## DESCRIPTION

Delete a flexible node which created by the **mkflexnode** command.

The **rmflexnode** command will delete the **Partition** which the slots in *id* attribute assigned to.

The action of deleting flexible node will impact the hardware status. Before deleting it, the blades in the slot range should be in **power off** state.

After the deleting, use the **lsflexnode** to check the status of the node.

The *noderange* only can be a blade node.

## OPTIONS

**-h | --help**

Display the usage message.

**-v | --version**

Display the version information.

## EXAMPLES

1 Delete a flexible node base on the xCAT node blade1.

The blade1 should belong to a complex, the *id* attribute should be set correctly and all the slots should be in **power off** state.

```
rmflexnode blade1
```

## FILES

/opt/xcat/bin/rmflexnode

## SEE ALSO

lsflexnode(1)|lsflexnode.1, mkflexnode(1)|mkflexnode.1

## rmhwconn.1

## NAME

**rmhwconn** - Use this command to remove connections from CEC and Frame nodes to HMC nodes.

## SYNOPSIS

**rmhwconn** [-h] --help]

**rmhwconn** [-v] --version]

### PPC (with HMC) specific:

**rmhwconn** [-V] --verbose] *noderange*

### PPC (without HMC, using FSPAPI) specific:

**rmhwconn** *noderange* -T *tooltype*

### PPC (use HMC as SFP) specific:

**rmhwconn** -s

## DESCRIPTION

For PPC (with HMC) specific:

This command is used to disconnect CEC and Frame nodes from HMC nodes, according to the connection information defined in ppc table in xCAT DB.

Note: If a CEC belongs to a frame with a BPA installed, this CEC cannot be disconnected individually. Instead, the whole frame should be disconnected.

For PPC (without HMC, using FSPAPI) specific:

It's used to disconnection CEC and Frame nodes from hardware server.

For PPC (use HMC as SFP) specific:

It is used to disconnect Frame nodes from HMC nodes.

## OPTIONS

### **-h|--help**

Display usage message.

### **-V|--verbose**

Verbose output.

### **-T**

The tooltype is used to communicate to the CEC/Frame. The value could be **lpar** or **fnm**. The tooltype value **lpar** is for xCAT and **fnm** is for CNM.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To disconnect all CEC nodes in node group cec from their HMC nodes:

```
rmhwconn cec
```

2. To remove the connection for Frame node frame1:

```
rmhwconn frame1
```

3. To disconnect all CEC nodes in node group cec from their related hardware serveri, using lpar tooltype:

```
rmhwconn cec -T lpar
```



## FILES

\$XCATROOT/bin/rmhwcconn

(The XCATROOT environment variable is set when xCAT is installed. The default value is “/opt/xcat”.)

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

lshwconn(1)|lshwconn.1, mkhwconn(1)|mkhwconn.1

## rmhypervisor.1

## NAME

**rmhypervisor** - Remove the virtualization hosts.

## SYNOPSIS

**RHEV specific :**

**rmhypervisor** *noderange* [-f]

## DESCRIPTION

The **rmhypervisor** command can be used to remove the virtualization host.

## OPTIONS

**-f**

If **-f** is specified, the host will be deactivated to maintenance before the removing.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To remove the host 'host1', enter:

```
rmhypervisor host1
```

## FILES

/opt/xcat/bin/rmhypervisor

## rmigrate.1

### Name

**rmigrate** - Execute migration of a guest VM between hosts/hypervisors

### Synopsis

**rmigrate** *noderange target\_host*

### For zVM:

**rmigrate** *noderange* [**destination**=*target\_host*] [**action**=*action*] [**force**=*force*] [**immediate**=*yes\_no*]  
[**max\_total**=*total*] [**max\_quiesce**=*quiesce*]

### Description

**rmigrate** requests that a guest VM to be moved from the current hypervisor to another. It will request a live migration if possible. The vmstorage directory should be shared between the source and destination hypervisors.

Note: Make sure SELINUX is disabled and firewall is turned off on source and destination hypervisors.

### For zVM:

**rmigrate** migrates a VM from one z/VM member to another in an SSI cluster (only in z/VM 6.2).

## OPTIONS

### zVM specific:

**destination=** The name of the destination z/VM system to which the specified virtual machine will be relocated.

**action=** It can be: (MOVE) initiate a VMRELOCATE MOVE of the VM, (TEST) determine if VM is eligible to be relocated, or (CANCEL) stop the relocation of VM.

**force=** It can be: (ARCHITECTURE) attempt relocation even though hardware architecture facilities or CP features are not available on destination system, (DOMAIN) attempt relocation even though VM would be moved outside of its

domain, or (STORAGE) relocation should proceed even if CP determines that there are insufficient storage resources on destination system.

**immediate=** It can be: (YES) VMRELOCATE command will do one early pass through virtual machine storage and then go directly to the quiesce stage, or (NO) specifies immediate processing.

**max\_total=** The maximum wait time for relocation to complete.

**max\_quiesce=** The maximum quiesce time a VM may be stopped during a relocation attempt.

## Files

**vm** table - Table governing VM parameters. See `vm(5)|vm.5` for further details. This is used to determine the current host to migrate from.

## Examples

1. To migrate kvm guest “kvm1” from hypervisor “hyp01” to hypervisor “hyp02”, run:

```
rmigrate kvm1 hyp02
```

2. To migrate kvm guest “kvm1” to hypervisor “hyp02”, which is already on host “hyp02”, enter:

```
rmigrate kvm1 hyp02
```

## zVM specific:

```
rmigrate ihost123 destination=pokdev62
```

## rmimage.1

### NAME

**rmimage** - Removes the Linux stateless or statelite image from the file system.

### SYNOPSIS

**rmimage** [-h | --help]

**rmimage** [-V | --verbose] *imagename* [--xcatdef]

## DESCRIPTION

Removes the Linux stateless or statelite image from the file system. The install dir is setup by using “installdir” attribute set in the site table.

If *imagename* is specified, this command uses the information in the *imagename* to calculate the image root directory; otherwise, this command uses the operating system name, architecture and profile name to calculate the image root directory.

The osimage definition will not be removed from the xCAT tables by default, specifying the flag **--xcatdef** will remove the osimage definition, or you can use `rmdef -t osimage` to remove the osimage definition.

The statelite image files on the diskful service nodes will not be removed, remove the image files on the service nodes manually if necessary, for example, use command “`rsync -az --delete /install <sn>:/`” to remove the image files on the service nodes, where the <sn> is the hostname of the service node.

## Parameters

*imagename* specifies the name of an os image definition to be used. The specification for the image is stored in the *osimage* table and *linuximage* table.

## OPTIONS

**-h | --help** Display usage message.

**-V | --verbose** Verbose mode.

**--xcatdef** Remove the xCAT osimage definition

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To remove a RHEL 7.1 stateless image for a compute node architecture x86\_64, enter:

```
rmimage rhels7.1-x86_64-netboot-compute
```

2. To remove a rhels5.5 statelite image for a compute node architecture ppc64 and the osimage definition, enter:

```
rmimage rhels5.5-ppc64-statelite-compute --xcatdef
```

## FILES

/opt/xcat/sbin/rmimage

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

genimage(1)|genimage.1, packimage(1)|packimage.1

## rmkit.1

## NAME

**rmkit** - Remove Kits from xCAT

## SYNOPSIS

**rmkit** [-? | -h | --help] [-v | --version]

**rmkit** [-V | --verbose] [-f | --force] [-t | --test] *kitlist*

## DESCRIPTION

The **rmkit** command removes kits on the xCAT management node from kit names.

Note: The xCAT support for Kits is only available for Linux operating systems.

## OPTIONS

### -h|--help

Display usage message.

### -V|--verbose

Verbose mode.

### -v|--version

Command version.

### -f|--force

Remove this kit even there is any component in this kit is listed by osimage.kitcomponents. If this option is not specified, this kit will not be removed if any kit components listed in an osimage.kitcomponents

### -t|--test

Test if kitcomponents in this kit are used by osimage

### *kitlist*

A comma delimited list of kits that are to be removed from the xCAT cluster. Each entry can be a kitname or kit basename. For kit basename, rmkit command will remove all the kits that have that kit basename.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To remove two kits from tarball files.

```
rmkit kit-test1,kit-test2
```

Output is similar to:

```
Kit kit-test1-1.0-Linux,kit-test2-1.0-Linux was successfully removed.
```

2. To remove two kits from tarball files even the kit components in them are still being used by osimages.

```
rmkit kit-test1,kit-test2 --force
```

Output is similar to:

```
Kit kit-test1-1.0-Linux,kit-test2-1.0-Linux was successfully removed.
```

3. To list kitcomponents in this kit used by osimage

```
rmkit kit-test1,kit-test2 -t
```

Output is similar to:

```
kit-test1-kitcomp-1.0-Linux is being used by osimage osimage-test  
Following kitcomponents are in use: kit-test1-kitcomp-1.0-Linux
```

## SEE ALSO

lskit(1)|lskit.1,    addkit(1)|addkit.1,    addkitcomp(1)|addkitcomp.1,    rmkitcomp(1)|rmkitcomp.1,    chkkitcomp(1)|chkkitcomp.1

## rmkitcomp.1

### NAME

**rmkitcomp** - Remove Kit components from an xCAT osimage.

### SYNOPSIS

**rmkitcomp** [-? | -h | --help] [-v | --version]

**rmkitcomp** [-V | --verbose] [-u | --uninstall] [-f | --force] [--noscripts] -i *osimage kitcompname\_list*

### DESCRIPTION

The **rmkitcomp** command removes kit components from an xCAT osimage. All the kit component attribute values that are contained in the osimage will be removed, and the kit component meta rpm and package rpm could be uninstalled by **-u|--uninstall** option.

Note: The xCAT support for Kits is only available for Linux operating systems.

### OPTIONS

#### **-u|--uninstall**

All the kit component meta rpms and package rpms in otherpkglist will be uninstalled during genimage for stateless image and updatenode for stateful nodes.

#### **-h|--help**

Display usage message.

#### **-V|--verbose**

Verbose mode.

#### **-v|--version**

Command version.

#### **-f|--force**

Remove this kit component from osimage no matter it is a dependency of other kit components.

#### **--noscripts**

Do not remove kitcomponent's postbootscripts from osimage

#### **-i *osimage***

osimage name that include this kit component.

#### ***kitcompname\_list***

A comma-delimited list of valid full kit component names or kit component basenames that are to be removed from the osimage. If a basename is specified, all kitcomponents matching that basename will be removed from the osimage.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To remove a kit component from osimage

```
rmkitcomp -i rhels6.2-ppc64-netboot-compute comp-test1-1.0-1-rhels-6.2-ppc64
```

Output is similar to:

```
kitcomponents comp-test1-1.0-1-rhels-6.2-ppc64 were removed from osimage rhels6.2-ppc64-  
↪netboot-compute successfully
```

2. To remove a kit component even it is still used as a dependency of other kit component.

```
rmkitcomp -f -i rhels6.2-ppc64-netboot-compute comp-test1-1.0-1-rhels-6.2-ppc64
```

Output is similar to:

```
kitcomponents comp-test1-1.0-1-rhels-6.2-ppc64 were removed from osimage rhels6.2-ppc64-  
↪netboot-compute successfully
```

3. To remove a kit component from osimage and also remove the kit component meta RPM and package RPM.  
So in next genimage for stateless image and updatenode for stateful nodes, the kit component meta RPM and package RPM will be uninstalled.

```
rmkitcomp -u -i rhels6.2-ppc64-netboot-compute comp-test1-1.0-1-rhels-6.2-ppc64
```

Output is similar to:

```
kitcomponents comp-test1-1.0-1-rhels-6.2-ppc64 were removed from osimage rhels6.2-ppc64-  
↪netboot-compute successfully
```

## SEE ALSO

lskit(1)|lskit.1, addkit(1)|addkit.1, rmkit(1)|rmkit.1, addkitcomp(1)|addkitcomp.1, chkkitcomp(1)|chkkitcomp.1

## rmnimimage.1

## NAME

**rmnimimage** - Use this xCAT command to remove NIM resources specified in an xCAT osimage definition.



## SYNOPSIS

**rmnimimage** [-h|--help]

**rmnimimage** [-V|--verbose] [-f|--force] [-d|--delete] [-x|--xcatdef] [-M|--managementnode] [-s *servicenoderange*]  
*osimage\_name*

## DESCRIPTION

Use this xCAT command to remove the AIX resources specified in an xCAT osimage definition.

To list the contents of the xCAT osimage definition use the xCAT **lsdef** command (“lsdef -t osimage -l -o <osimage\_name>”). **Before running the rmnimimage command you should be absolutely certain that you really want to remove the NIM resources specified in the xCAT osimage definition!**

The default behavior of this command is to remove all the NIM resources, except the lpp\_source, on the xCAT management node in addition to the resources that were replicated on any xCAT service nodes.

This command may also be used to clean up individual xCAT service nodes and remove the xCAT osimage definitions.

The “nim -o remove” operation is used to remove the NIM resource definitions. If you wish to completely remove all the files and directories (left behind by the NIM command) you must specify the “-d” option when you run **rmnimimage**. The “-d” option will also remove the lpp\_source resource.

If you wish to remove the NIM resource from one or more xCAT service nodes without removing the resources from the management node you can use the “-s <servicenoderange>” option. In this case the NIM resources specified in the xCAT osimage definition will be removed from the service nodes ONLY. The NIM resources on the management node will not be removed.

If you wish to remove NIM resources on the management node only, you can specify the “-M” option.

If you wish to also remove the xCAT osimage definition you must specify the “-x” option.

This command will not remove NIM resources if they are currently being used in another xCAT osimage definition. To see which resources are common between osimages you can specify the “-V” option. You can override this check by specifying the “-f” option.

This command will not remove NIM resources if they are currently allocated. You must deallocate the resources before they can be removed. See the **xcats2nim** and **rmdsklsnode** commands for information on how to deallocate and remove NIM machine definitions for standalone and diskless nodes.

See the AIX NIM documentation for additional details on how to deallocate and remove unwanted NIM objects.

## OPTIONS

**-h |--help**

Display usage message.

**-d|--delete**

Delete any files or directories that were left after the “nim -o remove” command was run. This option will also remove the lpp\_source resource and all files contained in the lpp\_source directories. When this command completes all definitions and files will be completely erased so use with caution!

**-f|--force**

Override the check for shared resources when removing an xCAT osimage.

**-M|--managementnode**

Remove NIM resources from the xCAT management node only.

**-s** *servicenoderange*

Remove the NIM resources on these xCAT service nodes only. Do not remove the NIM resources from the xCAT management node.

*osimage\_name*

The name of the xCAT osimage definition.

**-V|--verbose**

Verbose mode. This option will display the underlying NIM commands that are being called.

**-x|--xcatdef**

Remove the xCAT osimage definition.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

- 1) Remove all NIM resources specified in the xCAT “61image” definition.

```
rmnimimage 61image
```

The “nim -o remove” operation will be used to remove the NIM resource definitions on the management node as well as any service nodes where the resource has been replicated. This NIM operation does not completely remove all files and directories associated with the NIM resources.

- 2) Remove all the NIM resources specified by the xCAT “61rte” osimage definition. Delete ALL files and directories associated with the NIM resources. This will also remove the lpp\_source resource.

```
rmnimimage -d 61rte
```

- 3) Remove all the NIM resources specified by the xCAT “614img” osimage definition and also remove the xCAT definition.

```
rmnimimage -x -d 614img
```

Note: When this command completes all definitions and files will be completely erased, so use with caution!

- 4) Remove the NIM resources specified in the “614dskls” osimage definition on the xcatsn1 and xcatsn2 service nodes. Delete all files or directories associated with the NIM resources.

```
rmnimimage -d -s xcatsn1,xcatsn2 614dskls
```

- 5) Remove the NIM resources specified in the “614old” osimage definition on the xCAT management node only.

```
rmnimimage -M -d 614old
```

## FILES

/opt/xcat/bin/rmnimimage

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

mknimimage(1)|mknimimage.1

## rmvlan.1

## NAME

**rmvlan** - It removes the vlan from the cluster.

## SYNOPSIS

**rmvlan** *vlanid*

**rmvlan** [-h | --help]

**rmvlan** [-v | --version]

## DESCRIPTION

The **rmvlan** command removes the given vlan ID from the cluster. It removes the vlan id from all the switches involved, deconfigures the nodes so that vlan adaptor (tag) will be removed, cleans up /etc/hosts, DNS and database tables for the given vlan.

For added security, the root guard and bpdu guard were enabled for the ports in this vlan by mkvlan and chvlan commands. However, the guards will not be disabled by this command. To disable them, you need to use the switch command line interface. Refer to the switch command line interface manual to see how to disable the root guard and bpdu guard for a port.

## Parameters

*vlanid* is a unique vlan number.

## OPTIONS

**-h|--help** Display usage message.

**-v|--version** The Command Version.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To remove vlan 3

```
rmvlan 3
```

If the nodes are KVM guest then the do the following after the vlan is removed:

```
rpower node1,node2 off rmvm node1,node2
```

## FILES

/opt/xcat/bin/rmvlan

## SEE ALSO

mkvlan(1)|mkvlan.1, chvlan(1)|chvlan.1, lsvlan(1)|lsvlan.1

## rmvm.1

## NAME

**rmvm** - Removes HMC-, DFM-, IVM-, KVM-, VMware- and zVM-managed partitions or virtual machines.

## SYNOPSIS

**rmvm** [-h] --help]

**rmvm** [-v] --version]

**rmvm** [-V] --verbose] *noderange* [-r] [--service]

**For KVM and VMware:**

```
rmvm [-p] [-f] noderange
```

**PPC (using Direct FSP Management) specific:**

```
rmvm [-p] noderange
```

**DESCRIPTION**

The **rmvm** command removes the partitions specified in *noderange*. If *noderange* is an CEC, all the partitions associated with that CEC will be removed. Note that removed partitions are automatically removed from the xCAT database. For IVM-managed systems, care must be taken to not remove the VIOS partition, or all the associated partitions will be removed as well.

For DFM-managed (short For Direct FSP Management mode) normal Power machines, only partitions can be removed. No options are needed.

**OPTIONS****-h|--help**

Display usage message.

**-v|--version**

Command Version.

**-V|--verbose**

Verbose output.

**-r**

Retain the data object definitions of the nodes.

**--service**

Remove the service partitions of the specified CECs.

**-p**

KVM: Purge the existence of the VM from persistent storage. This will erase all storage related to the VM in addition to removing it from the active virtualization configuration. Storage devices of “raw” or “block” type are not removed.

PPC: Remove the specified partition on normal Power machine.

**-f**

Force remove the VM, even if the VM appears to be online. This will bring down a live VM if requested.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To remove the HMC-managed partition lpar3, enter:

```
rmvm lpar3
```

Output is similar to:

```
lpar3: Success
```

2. To remove all the HMC-managed partitions associated with CEC cec01, enter:

```
rmvm cec01
```

Output is similar to:

```
lpar1: Success  
lpar2: Success  
lpar3: Success
```

3. To remove the HMC-managed service partitions of the specified CEC cec01 and cec02, enter:

```
rmvm cec01,cec02 --service
```

Output is similar to:

```
cec01: Success  
cec02: Success
```

4. To remove the HMC-managed partition lpar1, but retain its definition, enter:

```
rmvm lpar1 -r
```

Output is similar to:

```
lpar1: Success
```

5. To remove a zVM virtual machine:

```
rmvm gpok4
```

Output is similar to:

```
gpok4: Deleting virtual server LNX4... Done
```

6. To remove a DFM-managed partition on normal Power machine:

```
rmvm lpar1
```

Output is similar to:

```
lpar1: Done
```

## FILES

/opt/xcat/bin/rmvm

## SEE ALSO

mkvm(1)|mkvm.1, lsvm(1)|lsvm.1, chvm(1)|chvm.1

## rmzone.1

## NAME

**rmzone** - Removes a zone from the cluster.

## SYNOPSIS

**rmzone** *zonename* [-g] [-f]

**rmzone** [-h | -v]

## DESCRIPTION

The **rmzone** command is designed to remove a previously defined zone from the cluster. It will remove the zone entry in the zone table. It will remove the zone from the zonename attributes on the nodes that were assigned to the zone. Optionally, it will remove the zonename group from the nodes that were assigned to the zone. It will also remove the root ssh keys that were created for that zone on the Management Node. The rmzone command is only supported on Linux ( No AIX support). The nodes are not automatically updated with new root ssh keys by rmzone. You must run updatenode -k or xdsh -K to the nodes to update the root ssh keys. The nodes new ssh key will be assigned from the defaultzone in the zone table, or if no entries in the zone table, the keys will come from /root/.ssh. Note: if any zones in the zone table, there must be one and only one defaultzone. Otherwise, errors will occur.

## OPTIONS

**-h | --help**

Displays usage information.

**-v | --version**

Displays command version and build date.

**-f | --force**

Used to remove a zone that is defined as current default zone. This should only be done if you are removing all zones, or you will adding a new zone or changing an existing zone to be the default zone.

**-g | --assigngroup**

Remove the assigned group named **zonename** from all nodes assigned to the zone being removed.

**-V | --verbose**

Verbose mode.

## Examples

1. To remove zone1 from the zone table and the zonename attribute on all it's assigned nodes , enter:

```
rmzone zone1
```

2. To remove zone2 from the zone table, the zone2 zonename attribute, and the zone2 group assigned to all nodes that were in zone2, enter:

```
rmzone zone2 -g
```

3. To remove zone3 from the zone table, all the node zone attributes and override the fact it is the defaultzone, enter:

```
rmzone zone3 -g -f
```

## Files

**/opt/xcat/bin/rmzone/**

Location of the rmzone command.

## SEE ALSO

mkzone(1)|mkzone.1, chzone(1)|chzone.1, xdsh(1)|xdsh.1, updatenode(1)|updatenode.1

## rnetboot.1

## NAME

**rnetboot** - Cause the range of nodes to boot to network.

## SYNOPSIS

**rnetboot** [-V | --verbose] [-s *boot\_device\_order*] [-F] [-f] *noderange* [-m *table.column==expectedstatus*] [-m *table.column=~expectedstatus*] [-t *timeout*] [-r *retrycount*]

**rnetboot** [-h | --help] [-v | --version]



**zVM specific:****rnetboot** *noderange* [**ipl=** *address*]**DESCRIPTION**

The **rnetboot** command will do what is necessary to make each type of node in the given *noderange* boot from the network. This is usually used to boot the nodes stateless or to network install system p nodes.

**OPTIONS****-s**

Set the boot device order. Accepted boot devices are **hd** and **net**.

**-F**

Force reboot the system no matter what state the node is. By default, **rnetboot** will not reboot the node if node is in 'boot' state.

**-f**

Force immediate shutdown of the partition.

**-m**

Use one or multiple **-m** flags to specify the node attributes and the expected status for the node installation monitoring and automatic retry mechanism. The operators **==**, **!=**, **=~** and **!~** are valid. This flag must be used with **-t** flag.

Note: if the "val" fields includes spaces or any other characters that will be parsed by shell, the "attr<operator>val" needs to be quoted. If the operator is "!", the "attr<operator>val" needs to be quoted using single quote.

**-r**

Specify the number of retries that the monitoring process will perform before declaring the failure. The default value is 3. Setting the retrycount to 0 means only monitoring the os installation progress and will not re-initiate the installation if the node status has not been changed to the expected value after timeout. This flag must be used with **-m** flag.

**-t**

Specify the timeout, in minutes, to wait for the expectedstatus specified by **-m** flag. This is a required flag if the **-m** flag is specified.

**-V|--verbose**

Verbose output.

**-h|--help**

Display usage message.

**-v|--version**

Command Version.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

```
rnetboot 1,3
rnetboot 14-56,70-203
rnetboot 1,3,14-56,70-203
rnetboot all,-129-256
rnetboot all -s hd,net
rnetboot all ipl=00c
```

## SEE ALSO

nodeset(8)|nodeset.8

## rollupdate.1

### NAME

**rollupdate** - performs cluster rolling update

### SYNOPSIS

**cat stanza-file | rollupdate** [-V | --verbose] [-t | --test]

**rollupdate** [-? | -h | --help | -v | --version]

### DESCRIPTION

The **rollupdate** command creates and submits scheduler reservation jobs that will notify xCAT to shutdown a group of nodes, run optional out-of-band commands from the xCAT management node, and reboot the nodes. Currently, only LoadLeveler is supported as a job scheduler with **rollupdate**.

Input to the **rollupdate** command is passed in as stanza data through STDIN. Information such as the sets of nodes that will be updated, the name of the job scheduler, a template for generating job command files, and other control data are required. See /opt/xcat/share/xcat/rollupdate/rollupdate.input.sample and /opt/xcat/share/xcat/rollupdate/rollupdate\_all.input.sample for stanza keywords, usage, and examples.

The **rollupdate** command will use the input data to determine each set of nodes that will be managed together as an update group. For each update group, a job scheduler command file is created and a reservation request is submitted. When the group of nodes becomes available and the scheduler activates the reservation, the xcatd daemon on the

management node will be notified to begin the update process for all the nodes in the update group. If specified, prescripts will be run, an operating system shutdown command will be sent to each node, out-of-band operations can be run on the management node, and the nodes are powered back on.

The **rollupdate** command assumes that, if the update is to include rebooting stateless or statelite nodes to a new operating system image, the image has been created and tested, and that all relevant xCAT commands have been run for the nodes such that the new image will be loaded when xCAT reboots the nodes.

## OPTIONS

### **-v|--version**

Command Version.

### **-V|--verbose**

Display additional progress and error messages. Output is also logged in /var/log/xcat/rollupdate.log.

### **-t|--test**

Run the rollupdate command in test mode only to verify the output files that are created. No scheduler reservation requests will be submitted.

### **-?|-h|--help**

Display usage message.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To run a cluster rolling update based on the information you have created in the file /u/admin/rolling\_updates/update\_all.stanza enter:

```
cat /u/admin/rolling_updates/update_all.stanza | rollupdate
```

## FILES

/opt/xcat/bin/rollupdate

/opt/xcat/share/xcat/rollupdate/rollupdate.input.sample

/opt/xcat/share/xcat/rollupdate/ll.tmpl

/opt/xcat/share/xcat/rollupdate/rollupdate\_all.input.sample

/opt/xcat/share/xcat/rollupdate/llall.tmpl

/var/log/xcat/rollupdate.log

## SEE ALSO

`rpower.1`

## NAME

**rpower** - remote power control of nodes

## SYNOPSIS

**rpower** *noderange* [--nodeps] [on | onstandby | off | suspend | stat | state | reset | boot] [-m *table.column==expectedstatus*] [-m *table.column=~expectedstatus*] [-t *timeout*] [-r *retrycount*]

**rpower** [-h | --help | -v | --version]

### BMC (using IPMI):

**rpower** *noderange* [on | off | softoff | reset | boot | cycle | stat | state | status | wake | suspend [-w *timeout*] [-o] [-r]]

**rpower** *noderange* [pduon | pduoff | pdustat | pdureset]

### OpenPOWER BMC (using IPMI):

**rpower** *noderange* [on | off | reset | boot | stat | state | status]

**rpower** *noderange* [pduon | pduoff | pdustat | pdureset]

### OpenPOWER OpenBMC:

**rpower** *noderange* [off | on | softoff | reset | boot | bmcreboot | bmcstate | stat | state | status]

### PPC (with IVM or HMC) specific:

**rpower** *noderange* [--nodeps] {of}

### CEC (with HMC) specific:

**rpower** *noderange* [on | off | reset | boot | onstandby]

#### LPAR (with HMC) specific:

**rpower** *noderange* [**on** | **off** | **stat** | **state** | **reset** | **boot** | **of** | **sms** | **softoff**]

#### CEC (using Direct FSP Management) specific:

**rpower** *noderange* [**onstandby** | **stat** | **state**] [-T **tooltype**]

**rpower** *noderange* [**on** | **off** | **resetsp**]

#### Frame (using Direct FSP Management) specific:

**rpower** *noderange* [**rackstandby** | **exit\_rackstandby** | **stat** | **state** | **resetsp**]

#### LPAR (using Direct FSP Management) specific:

**rpower** *noderange* [**on** | **off** | **stat** | **state** | **reset** | **boot** | **of** | **sms**]

#### Blade (using Direct FSP Management) specific:

**rpower** *noderange* [**on** | **onstandby** | **off** | **stat** | **state** | **sms**]

#### Blade specific:

**rpower** *noderange* [**cycle** | **softoff**]

#### Lenovo High-Density Server specific:

**rpower** *noderange* [**on** | **off** | **reset** | **boot** | **reseat**]

#### zVM specific:

**rpower** *noderange* [**on** | **off** | **reset** | **stat** | **softoff**]

#### pdu specific:

**rpower** *noderange* [**stat** | **off** | **on** | **reset**]

## DESCRIPTION

**rpower** controls the power for a single or range of nodes, via the out-of-band path.

## OPTIONS

### on

Turn power on.

### onstandby

Turn power on to standby state

### -T

The value could be **lpar** or **fnm**. The tooltype value **lpar** is for xCAT and **fnm** is for CNM. The default value is “**lpar**”. For cold start in the large cluster, it will save a lot of time if the admins use “**rpower noderange onstandby -T fnm**” to power on all the CECs from the management node through the **fnm** connections.

### rackstandby

Places the rack in the rack standby state. It requires that all CECs and DE be powered off before it will run.

### exit\_rackstandby

Exit Rack standby will be the default state that a rack goes into when power is initially applied to the rack. It simply moves the BPA from Rack standby to both bpa's in standby state.

### resetsp

Reboot the service processor. If there are primary and secondary FSPs/BPAs of one cec/frame, it will reboot them almost at the same time.

### softoff

Attempt to request clean shutdown of OS (may not detect failures in completing command)

### off

Turn power off.

### suspend

Suspend the target nodes execution.

The **suspend** action could be run together with **-w -o -r**.

Refer to the following steps to enable the **suspend** function:

1. Add the ‘acpid’ and ‘suspend’(the suspend package is not needed on RHEL) package to the .pkglist of your osimage so that the required package could be installed correctly to your target system.
2. Add two configuration files for the base function:

```
/etc/pm/config.d/suspend
S2RAM_OPTS="--force --vbe_save --vbe_post --vbe_mode"

/etc/acpi/events/suspend_event
event=button/sleep.*
action=/usr/sbin/pm-suspend
```

3. Add the hook files for your specific applications which need specific action before or after the suspend action.

Refer to the 'pm-utils' package for how to create the specific hook files.

## wake

Wake up the target nodes which is in **suspend** state.

Don't try to run **wake** against the 'on' state node, it would cause the node gets to 'off' state.

For some of xCAT hardware such as NeXtScale, it may need to enable S3 before using **wake**. The following steps can be used to enable S3. Reference pasu(1)|pasu.1 for "pasu" usage.

```
[root@xcatmn home]# echo "set Power.S3Enable Enable" > power-setting
[root@xcatmn home]# pasu -b power-setting node01
node01: Batch mode start.
node01: [set Power.S3Enable Enable]
node01: Power.S3Enable=Enable
node01:
node01: Beginning intermediate batch update.
node01: Waiting for command completion status.
node01: Command completed successfully.
node01: Completed intermediate batch update.
node01: Batch mode completed successfully.

[root@xcatmn home]# pasu node01 show all/grep -i s3
node01: IMM.Community_HostIPAddress3.1=
node01: IMM.Community_HostIPAddress3.2=
node01: IMM.Community_HostIPAddress3.3=
node01: IMM.DNS_IP_Address3=0.0.0.0
node01: IMM.IPv6DNS_IP_Address3=:
node01: Power.S3Enable=Enable
```

## stat | state

Print the current power state/status.

## reset

Send a hard reset.

## boot

If off, then power on. If on, then hard reset. This option is recommended over **cycle**.

## cycle

Power off, then on.

## reseat

For Lenovo high-density servers, simulates unplugging and replugging the node into the chassis.

## of

Boot the node to open firmware console mode.

## sms

Boot the node to open firmware SMS menu mode.

**-m** *table.column==expectedstatus* **-m** *table.column=~expectedstatus*

Use one or multiple **-m** flags to specify the node attributes and the expected status for the node installation monitoring and automatic retry mechanism. The operators **==**, **!=**, **=~** and **!~** are valid. This flag must be used with **-t** flag.

Note: if the “val” fields includes spaces or any other characters that will be parsed by shell, the “attr<operator>val” needs to be quoted. If the operator is “!~”, the “attr<operator>val” needs to be quoted using single quote.

**--nodeps**

Do not use dependency table (default is to use dependency table). Valid only with **on|off|boot|reset|cycle** for blade power method and **on|off|reset|softoff** for hmc/fsp power method.

**-r** *retrycount*

specify the number of retries that the monitoring process will perform before declaring the failure. The default value is 3. Setting the retrycount to 0 means only monitoring the os installation progress and will not re-initiate the installation if the node status has not been changed to the expected value after timeout. This flag must be used with **-m** flag.

**-t** *timeout*

Specify the timeout, in minutes, to wait for the expectedstatus specified by **-m** flag. This is a required flag if the **-m** flag is specified.

Power off, then on.

**-w** *timeout*

To set the *timeout* for the **suspend** action to wait for the success.

**-o**

To specify that the target node will be power down if **suspend** action failed.

**-r**

To specify that the target node will be reset if **suspend** action failed.

**pause**

To pause all processes in the instance.

**unpause**

To unpause all processes in the instance.

**bmcreboot**

To reboot BMC.

**bmcstate**

To get state of the BMC.

**state**

To get state of the instance.

**-h | --help**

Prints out a brief usage message.

**-v | --version**

Display the version number.



## EXAMPLES

1. To display power status of nodes4 and note5

```
rpower node4,node5 stat
```

Output is similar to:

```
node4: on
node5: off
```

2. To power on node5

```
rpower node5 on
```

Output is similar to:

```
node5: on
```

## SEE ALSO

noderange(3)|noderange.3, rcons(1)|rcons.1, rinov(1)|rinov.1, rvitals(1)|rvitals.1, rscan(1)|rscan.1

## rscan.1

## NAME

**rscan** - Collects node information from one or more hardware control points.

## SYNOPSIS

**rscan** [-h|--help]

**rscan** [-v|--version]

**rscan** [-V|--verbose] *noderange* [-u][*-w*][*-x*][*-z*]

## DESCRIPTION

The rscan command lists hardware information for each node managed by the hardware control points specified in noderange.

For the management module of blade, if the blade server is a Flex system P node, the fsp belongs to the blade server also will be scanned.

For the KVM host, all the KVM guests on the specified KVM host will be scanned.

Note: The first line of the output always contains information about the hardware control point. When using the rscan command to generate output for HMC or IVM hardware control points, it provides the FSPs and BPAs as part of the output. The only exception is the rscan -u flag which provides updates made hardware control point in the xCAT database.

## OPTIONS

**-h|--help** Display usage message.

**-v|--version** Command Version.

**-V|--verbose** Verbose output.

**-u** Updates and then prints out node definitions in the xCAT database for CEC/BPA. It updates the existing nodes that contain the same mtms and serial number for nodes managed by the specified hardware control point. This primarily works with CEC/FSP and frame/BPA nodes when the node name is not the same as the managed system name on hardware control point (HMC). This flag will update the BPA/FSP node name definitions to be listed as the managed system name in the xCAT database.

For the Flex system manager, both the blade server and fsp object of xCAT will be updated if the mpa and slot id are matched to the object which has been defined in the xCAT database.

For KVM host, the information of the KVM guests which have been defined in xCAT database will be updated.

Note: only the matched object will be updated.

**-n** For KVM host, the information of the KVM guests, which are not defined in xCAT database yet, will be written into xCAT database.

**-w** Writes output to xCAT database.

For KVM host, updates the information of the KVM guests which have been defined in xCAT database with the same node name and KVM host, creates the definition of the KVM guests which do not exist in xCAT database, and notifies user about the conflicting KVM guests that the name exist in xCAT database but the kvm host is different.

**-x** XML format.

**-z** Stanza formatted output.

Note: For KVM host, -z is not a valid option for rscan.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To list all nodes managed by HMC hmc01 in tabular format, enter:

```
rscan hmc01
```

Output is similar to:

type	name	id	type-model	serial-number	address
hmc	hmc01		7310-C05	10F426A	hmc01
fsp	Server-9117-MMA-SN10F6F3D		9117-MMA	10F6F3D	3.3.3.197
lpar	lpar3	4	9117-MMA	10F6F3D	
lpar	lpar2	3	9117-MMA	10F6F3D	
lpar	lpar1	2	9117-MMA	10F6F3D	
lpar	p6vios	1	9117-MMA	10F6F3D	

2. To list all nodes managed by IVM ivm02 in XML format and write the output to the xCAT database, enter:

```
rscan ivm02 -x -w
```

Output is similar to:

```
<Node>
  <cons></cons>
  <profile></profile>
  <parent></parent>
  <serial>10B7D1G</serial>
  <model>9133-55A</model>
  <node>Server-9133-55A-10B7D1G</node>
  <mgt>ivm</mgt>
  <nodetype>fsp</nodetype>
  <hcp>ivm02</hcp>
  <groups>fsp,all</groups>
  <id>10</id>
</Node>

<Node>
  <cons>ivm</cons>
  <profile>lpar01</profile>
  <parent>Server-9133-55A-10B7D1G</parent>
  <serial></serial>
  <model></model>
  <node>lpar01</node>
  <mgt>ivm</mgt>
  <nodetype>lpar,osi</nodetype>
  <hcp>ivm02</hcp>
  <groups>lpar,all</groups>
  <id>1</id>
<Node>

</Node>
  <cons>ivm</cons>
  <profile>lpar02</profile>
  <parent>Server-9133-55A-10B7D1G</parent>
  <serial></serial>
  <model></model>
  <node>lpar02</node>
  <mgt>ivm</mgt>
  <nodetype>lpar,osi</nodetype>
  <hcp>ivm02</hcp>
  <groups>lpar,all</groups>
  <id>2</id>
</Node>
```

3. To list all nodes managed by HMC hmc02 in stanza format and write the output to the xCAT database, enter:

```
rscan hmc02 -z -w
```

Output is similar to:

Server-9458-100992001Y\_B:

```
objtype=node
nodetype=bpa
id=2
model=9458-100
serial=992001Y
hcp=hmc02
profile=
parent=
groups=bpa,all
mgt=hmc
cons=
```

Server-9119-590-SN02C5F9E:

```
objtype=node
type=fsp
id=10
model=9119-590
serial=02C5F9E
hcp=hmc02
profile=
parent=Server-9458-100992001Y_B
groups=fsp,all
mgt=hmc
cons=
```

lpar01:

```
objtype=node
nodetype=lpar,osi
id=1
model=
serial=
hcp=hmc02
profile=lpar01
parent=Server-9119-590-SN02C5F9E
groups=lpar,all
mgt=hmc
cons=hmc
```

lpar02:

```
objtype=node
nodetype=lpar,osi
id=2
model=
serial=
hcp=hmc02
profile=lpar02
parent=Server-9119-590-SN02C5F9E
groups=lpar,all
mgt=hmc
cons=hmc
```

4. To update definitions of nodes, which is managed by hmc03, enter:

```
rscan hmc03 -u
```

Output is similar to:

*#Updated following nodes:*

type	name	id	type-model	serial-number	address
fsp	Server-9125-F2A-SN0262672-B	3	9125-F2A	0262672	192.168.200.243

5. To collect the node information from one or more hardware control points on zVM AND populate the database with details collected by rscan:

```
rscan gpok2 -w
```

Output is similar to:

```
gpok2:
  objtype=node
  arch=s390x
  os=sles10sp3
  hcp=gpok3.endicott.ibm.com
  userid=LINUX2
  nodetype=vm
  parent=POKDEV61
  groups=all
  mgt=zvm
```

6. To scan the Flex system cluster:

```
rscan cmm01
```

Output is similar to:

type	name	id	type-model	serial-number	mpa	address
cmm	AMM680520153	0	789392X	100048A	cmm01	cmm01
blade	SN#YL10JH184067	1	789542X	10F752A	cmm01	12.0.0.9
xblade	SN#YL10JH184068	2	789542X	10F652A	cmm01	12.0.0.10
blade	SN#YL10JH184079	3	789542X	10F697A	cmm01	12.0.0.11

7. To update the Flex system cluster:

```
rscan cmm01 -u
```

Output is similar to:

cmm	[AMM680520153]	Matched To =>	[cmm01]
blade	[SN#YL10JH184067]	Matched To =>	[cmm01node01]
blade	[SN#YL10JH184079]	Matched To =>	[cmm01node03]

8. To scan the KVM host “hyp01”, write the KVM guest information into xCAT database:

```
rscan hyp01 -w
```

9. To update definitions of kvm guest, which is managed by hypervisor hyp01, enter:

```
rscan hyp01 -u
```

Output is similar to:

type	name	hypervisor	id	cpu	memory	nic	disk
kvm	kvm2	hyp01	12	2	1024	virbr0	/install/vms/kvm2.hda.
↪ qcow2							
kvm	kvm1	hyp01	10	1	1024	virbr0	/install/vms/kvm1.hda.
↪ qcow2							

## FILES

/opt/xcat/bin/rscan

## SEE ALSO

lsslp(1)|lsslp.1

## rsetboot.1

## SYNOPSIS

**rsetboot** *nodename* [**hd** | **net** | **cd** | **default** | **stat**] [-u] [-p]

**rsetboot** [-h | --help | -v | --version | -V | --verbose]

## DESCRIPTION

**rsetboot** sets the boot media and boot mode that should be used on the next boot of the specified nodes. After the nodes are booted with the specified device and boot mode (e.g. via `rpower(1)`|`rpower.1`), the nodes will return to using the default boot device specified in the BIOS.

## OPTIONS

### hd

Boot from the hard disk.

### net

Boot over the network, using a PXE or BOOTP broadcast.

### cd

Boot from the CD or DVD drive.

### def | default

Boot using the default set in BIOS.

### stat

Display the current boot setting.

## -u

To specify the next boot mode to be “UEFI Mode”. (Not supported for OpenBMC)

## -p

To make the specified boot device and boot mode settings persistent.

## EXAMPLES

1. Set nodes 1 and 3 to boot from the network on the next boot:

```
rsetboot node1,node3 net
```

2. Display the next-boot value for nodes 14-56 and 70-203:

```
rsetboot node[14-56],node[70-203] stat
```

Or:

```
rsetboot node[14-56],node[70-203]
```

3. Restore the next-boot value for these nodes back to their default set in the BIOS:

```
rsetboot node1,node3,node[14-56],node[70-203] default
```

## SEE ALSO

rbootseq(1)|rbootseq.1

## rspconfig.1

## NAME

**rspconfig** - Configures nodes' service processors

## SYNOPSIS

```
rspconfig [-h | --help | -v | --version | -V | --verbose]
```

## BMC/MPA specific:

```
rspconfig noderange {alert | snmpdest | community}
```

```
rspconfig noderange alert={on | enable | off | disable}
```

```
rspconfig noderange snmpdest=snmpmanager-IP
```

```
rspconfig noderange community={public | string}
```

### BMC specific:

**rspconfig noderange** {ip | netmask | gateway | backupgateway | garp | vlan}

**rspconfig noderange** garp=*time*

### OpenBMC specific:

**rspconfig noderange** {ipsrc | ip | netmask | gateway | vlan}

**rspconfig noderange** admin\_passwd={currentpasswd,newpasswd}

**rspconfig noderange** autoreboot

**rspconfig noderange** autoreboot={0|1}

**rspconfig noderange** bootmode

**rspconfig noderange** bootmode={safe|regular|setup}

**rspconfig noderange** dump [-l | --list] [-g | --generate] [-c | --clear {id | all}] [-d | --download {id | all}]

**rspconfig noderange** gard -c|--clear

**rspconfig noderange** ip=dhcp

**rspconfig noderange** hostname

**rspconfig noderange** hostname={\* | name}

**rspconfig noderange** ntpservers

**rspconfig noderange** ntpservers={ntpservers}

**rspconfig noderange** powerrestorepolicy

**rspconfig noderange** powerrestorepolicy={always\_on|restore|always\_off}

**rspconfig noderange** powersupplyredundancy

**rspconfig noderange** powersupplyredundancy={disabled|enabled}

**rspconfig noderange** sshcfg

**rspconfig noderange** thermalmode

**rspconfig noderange** thermalmode={default|custom|heavy\_io|max\_base\_fan\_floor}

**rspconfig noderange** timesyncmethod

**rspconfig noderange** timesyncmethod={manual|ntp}

### MPA specific:

**rspconfig noderange** {sshcfg | snmpcfg | pd1 | pd2 | network | swnet | ntp | textid | frame}

**rspconfig noderange** USERID={newpasswd} updateBMC={y | n}

**rspconfig noderange** sshcfg={enable | disable}

**rspconfig noderange** snmpcfg={enable | disable}

**rspconfig noderange** solcfg={enable | disable}

**rspconfig noderange** pd1={nonred | redwoperf | redwperf}



```
rspconfig noderange pd2={nonred | redwoperf | redwperf}
rspconfig noderange network={ [ip],[host],[gateway],[netmask]]* }
rspconfig noderange initnetwork={ [ip],[host],[gateway],[netmask]]* }
rspconfig noderange textid={ * | textid }
rspconfig singlenode frame={frame_number}
rspconfig noderange frame={ * }
rspconfig noderange swnet={ [ip],[gateway],[netmask] }
rspconfig noderange ntp={ [ntpenable],[ntpserver],[frequency],[v3] }
```

### FSP/CEC specific:

```
rspconfig noderange { autopower | iocap | decfg | memdecfg | procdecfg | time | date | spdump | sysdump | network }
rspconfig noderange autopower={ enable | disable }
rspconfig noderange iocap={ enable | disable }
rspconfig noderange time=hh:mm:ss
rspconfig noderange date=mm:dd:yyyy
rspconfig noderange decfg={ enable|disable:polycname,... }
rspconfig noderange procdecfg={ configure|deconfigure:processingunit:id,... }
rspconfig noderange memdecfg={ configure|deconfigure:processingunit:unit|bank:id,... > }
rspconfig noderange network={ nic,* }
rspconfig noderange network={ nic,[IP],[hostname],[gateway],[netmask] }
rspconfig noderange network={ nic,0.0.0.0 }
rspconfig noderange HMC_passwd={ currentpasswd,newpasswd }
rspconfig noderange admin_passwd={ currentpasswd,newpasswd }
rspconfig noderange general_passwd={ currentpasswd,newpasswd }
rspconfig noderange *_passwd={ currentpasswd,newpasswd }
rspconfig noderange { hostname }
rspconfig noderange hostname={ * | name }
rspconfig noderange --resetnet
```

### Flex system Specific:

```
rspconfig noderange sshcfg={ enable | disable }
rspconfig noderange snmpcfg={ enable | disable }
rspconfig noderange network={ [ip],[host],[gateway],[netmask] | * }
rspconfig noderange solcfg={ enable | disable }
rspconfig noderange textid={ * | textid }
rspconfig noderange cec_off_policy={ poweroff | stayon }
```

### BPA/Frame Specific:

```
rspconfig noderange {network}
rspconfig noderange network={nic,*}
rspconfig noderange network={nic,[IP],[hostname],[gateway],[netmask]}
rspconfig noderange network={nic,0.0.0.0}
rspconfig noderange HMC_passwd={currentpasswd,newpasswd}
rspconfig noderange admin_passwd={currentpasswd,newpasswd}
rspconfig noderange general_passwd={currentpasswd,newpasswd}
rspconfig noderange *_passwd={currentpasswd,newpasswd}
rspconfig noderange {hostname}
rspconfig noderange hostname={* | name}
rspconfig noderange --resetnet
```

### FSP/CEC (using Direct FSP Management) Specific:

```
rspconfig noderange HMC_passwd={currentpasswd,newpasswd}
rspconfig noderange admin_passwd={currentpasswd,newpasswd}
rspconfig noderange general_passwd={currentpasswd,newpasswd}
rspconfig noderange *_passwd={currentpasswd,newpasswd}
rspconfig noderange {sysname}
rspconfig noderange sysname={* | name}
rspconfig noderange {pending_power_on_side}
rspconfig noderange pending_power_on_side={temp | perm}
rspconfig noderange {cec_off_policy}
rspconfig noderange cec_off_policy={poweroff | stayon}
rspconfig noderange {BSR}
rspconfig noderange {huge_page}
rspconfig noderange huge_page={NUM}
rspconfig noderange {setup_failover}
rspconfig noderange setup_failover={enable | disable}
rspconfig noderange {force_failover}
rspconfig noderange --resetnet
```

### BPA/Frame (using Direct FSP Management) Specific:

```
rspconfig noderange HMC_passwd={currentpasswd,newpasswd}
rspconfig noderange admin_passwd={currentpasswd,newpasswd}
rspconfig noderange general_passwd={currentpasswd,newpasswd}
rspconfig noderange *_passwd={currentpasswd,newpasswd}
rspconfig noderange {frame}
rspconfig noderange frame={* | frame_number}
rspconfig noderange {sysname}
rspconfig noderange sysname={* | name}
rspconfig noderange {pending_power_on_side}
rspconfig noderange pending_power_on_side={temp | perm}
rspconfig noderange --resetnet
```

### HMC Specific:

```
rspconfig noderange {sshcfg}
rspconfig noderange sshcfg={enable | disable}
rspconfig noderange --resetnet
```

## DESCRIPTION

**rspconfig** configures various settings in the nodes' service processors.

For options **autopower** | **iocap** | **decfg** | **memdecfg** | **prodecfg** | **time** | **date** | **spdump** | **sysdump** | **network**, user need to use `chdef -t site enableASMI=yes` to enable ASMI first.

## OPTIONS

**alert**={on | enable | off | disable}

Turn on or off SNMP alerts.

**autopower**={enable | disable}

Select the policy for auto power restart. If enabled, the system will boot automatically once power is restored after a power disturbance.

**backupgateway**

Get the BMC backup gateway ip address.

**community**={public | string}

Get or set the SNMP community value. The default is **public**.

**date**=mm:dd:yyy

Enter the current date.

**decfg**={**enable** | **disable**:*policyname*,... }

Enables or disables deconfiguration policies.

**frame**={*framenumber* | \*}

Set or get frame number. If no *framenumber* and \* specified, *framenumber* for the nodes will be displayed and updated in the xCAT database. If *framenumber* is specified, it only supports single node and the *framenumber* will be set for that frame. If \* is specified, it supports *noderange* and all the frame numbers for the *noderange* will be read from xCAT database and set to frames. Setting the frame number is a disruptive command which requires all CECs to be powered off prior to issuing the command.

**cec\_off\_policy**={**poweroff** | **stayon**}

Set or get cec off policy after *lpars* are powered off. If no **cec\_off\_policy** value specified, the **cec\_off\_policy** for the nodes will be displayed. The **cec\_off\_policy** has two values: **poweroff** and **stayon**. **poweroff** means Power off when last partition powers off. **stayon** means Stay running after last partition powers off. If **cec\_off\_policy** value is specified, the cec off policy will be set for that cec.

**HMC\_passwd**={*currentpasswd,newpasswd*}

Change the password of the userid **HMC** for CEC/Frame. If the CEC/Frame is the factory default, the *currentpasswd* should NOT be specified; otherwise, the *currentpasswd* should be specified to the current password of the userid **HMC** for the CEC/Frame.

**admin\_passwd**={*currentpasswd,newpasswd*}

Change the password of the userid **admin** for CEC/Frame from *currentpasswd* to *newpasswd*. If the CEC/Frame is the factory default, the *currentpasswd* should NOT be specified; otherwise, the *currentpasswd* should be specified to the current password of the userid **admin** for the CEC/Frame.

**general\_passwd**={*currentpasswd,newpasswd*}

Change the password of the userid **general** for CEC/Frame from *currentpasswd* to *newpasswd*. If the CEC/Frame is the factory default, the *currentpasswd* should NOT be specified; otherwise, the *currentpasswd* should be specified to the current password of the userid **general** for the CEC/Frame.

**\*\_passwd**={*currentpasswd,newpasswd*}

Change the passwords of the userids **HMC**, **admin** and **general** for CEC/Frame from *currentpasswd* to *newpasswd*. If the CEC/Frame is the factory default, the *currentpasswd* should NOT be specified; otherwise, if the current passwords of the userids **HMC**, **admin** and **general** for CEC/Frame are the same one, the *currentpasswd* should be specified to the current password, and then the password will be changed to the *newpasswd*. If the CEC/Frame is NOT the factory default, and the current passwords of the userids **HMC**, **admin** and **general** for CEC/Frame are NOT the same one, this option could NOT be used, and we should change the password one by one.

**frequency**

The NTP update frequency (in minutes).

**gard -c|--clear**

Clear *gard* file. [OpenBMC]

**garp**=*time*

Get or set Gratuitous ARP generation interval. The unit is number of 1/2 second.

**gateway**

The gateway ip address.

**hostname**

Display the CEC/BPA system names.

## BSR

Get Barrier Synchronization Register (BSR) allocation for a CEC.

## huge\_page

Query huge page information or request NUM of huge pages for CEC. If no value specified, it means query huge page information for the specified CECs, if a CEC is specified, the specified huge\_page value NUM will be used as the requested number of huge pages for the CEC, if CECs are specified, it means to request the same NUM huge pages for all the specified CECs.

## setup\_failover={enable | disable}

Enable or disable the service processor failover function of a CEC or display status of this function.

## force\_failover

Force a service processor failover from the primary service processor to the secondary service processor.

## hostname={\* | name}

Set CEC/BPA system names to the names in xCAT DB or the input name.

## iocap={enable | disable}

Select the policy for I/O Adapter Enlarged Capacity. This option controls the size of PCI memory space allocated to each PCI slot.

## hostname

Get or set hostname on the service processor.

## vlan

Get or set vlan ID. For get vlan ID, if vlan is not enabled, 'BMC VLAN disabled' will be displayed. For set vlan ID, the valid value are [1-4096].

## ipsrc

Get the IP source for OpenBMC.

## ip

The IP address.

## memdecfg={configure | deconfigure:processingunit:unit|bank:id,...}

Select whether each memory bank should be enabled or disabled. State changes take effect on the next platform boot.

## netmask

The subnet mask.

## powerrestorepolicy

Display or control BMC Power Restore Policy attribute setting. [OpenBMC]

## powersupplyredundancy

Display or control BMC Power Supply Redundancy attribute setting. [OpenBMC]

## autoreboot

Display or control BMC Auto Reboot attribute setting. [OpenBMC]

## bootmode

Display or control BMC Boot Mode attribute setting. [OpenBMC]

**dump**

Generate/Manage BMC system dumps. If no sub-option is provided, will generate, wait, and download the dump. [OpenBMC]

**-c** will clear a single specified dump, or use 'all' to clear all dumps on the BMC.

**-l** will list all the generated dumps on the BMC.

**-g** will generate a new dump on the BMC. Dump generation can take a few minutes.

**-d** will download a single dump or all generated dumps from the BMC to /var/log/xcat/dump on management or service node.

**thermalmode**

Display or set the thermal mode of the system to a setting, depending on your system, adapter, and cable type. After a factory reset of the system, the thermal mode setting is lost and must be reapplied. To choose the correct setting for your system, see [https://www.ibm.com/support/knowledgecenter/POWER9/p9ei3/p9ei3\\_thermal\\_mode.htm](https://www.ibm.com/support/knowledgecenter/POWER9/p9ei3/p9ei3_thermal_mode.htm) [OpenBMC]

**timesyncmethod**

Set the method for time synchronization on the BMC. [OpenBMC]

**network**=*{[ip],[host],[gateway],[netmask]]\*}*

For MPA: get or set the MPA network parameters. If '\*' is specified, all parameters are read from the xCAT database.

For FSP of Flex system P node: set the network parameters. If '\*' is specified, all parameters are read from the xCAT database.

**initnetwork**=*{[ip],[host],[gateway],[netmask]]\*}*

For MPA only. Connecting to the IP of MPA from the hosts.otherinterfaces to set the MPA network parameters. If '\*' is specified, all parameters are read from the xCAT database.

**network**=*{nic,{[ip],[host],[gateway],[netmask]]\*}*

Not only for FSP/BPA but also for IMM. Get or set the FSP/BPA/IMM network parameters. If '\*' is specified, all parameters are read from the xCAT database. If the value of *ip* is '0.0.0.0', this *nic* will be configured as a DHCP client. Otherwise this *nic* will be configured with a static IP.

Note that IPs of FSP/BPAs will be updated with this option, user needs to put the new IPs to /etc/hosts manually or with xCAT command makehosts. For more details, see the man page of makehosts.

**nonred**

Allows loss of redundancy.

**ntp**=*{[ntpenable],[ntpserver],[frequency],[v3]}*

Get or set the MPA Network Time Protocol (NTP) parameters.

**ntpenable**

Enable or disable NTP (enable|disable).

**ntpserver**

Get or set NTP server IP address or name.

**ntpserver**s

Get or set NTP servers name. [OpenBMC]

**pd1={nonred | redwoperf | redwperf }**

Power Domain 1 - determines how an MPA responds to a loss of redundant power.

**pd2={nonred | redwoperf | redwperf }**

Power Domain 2 - determines how an MPA responds to a loss of redundant power.

**procdcfg={configure|deconfigure;processingunit:id,... }**

Selects whether each processor should be enabled or disabled. State changes take effect on the next platform boot.

**redwoperf**

Prevents components from turning on that will cause loss of power redundancy.

**redwperf**

Power throttles components to maintain power redundancy and prevents components from turning on that will cause loss of power redundancy.

**snmpcfg={enable | disable }**

Enable or disable SNMP on MPA.

**snmpdest=snmpmanager-IP**

Get or set where the SNMP alerts should be sent to.

**solcfg={enable | disable }**

Enable or disable the sol on MPA (or CMM) and blade servers belongs to it.

**spdump**

Performs a service processor dump.

**sshcfg={enable | disable }**

Enable or disable SSH on MPA.

**sshcfg**

Copy SSH keys.

**swnet={{[ip],[gateway],[netmask]}}**

Set the Switch network parameters.

**sysdump**

Performs a system dump.

**sysname**

Query or set sysname for CEC or Frame. If no value specified, means to query sysname of the specified nodes. If '\*' specified, it means to set sysname for the specified nodes, and the sysname values would get from xCAT database. If a string is specified, it means to use the string as sysname value to set for the specified node.

**pending\_power\_on\_side={temp|perm }**

List or set pending power on side for CEC or Frame. If no pending\_power\_on\_side value specified, the pending power on side for the CECs or frames will be displayed. If specified, the pending\_power\_on\_side value will be set to CEC's FSPs or Frame's BPAs. The value 'temp' means T-side or temporary side. The value 'perm' means P-side or permanent side.

**time=hh:mm:ss**

Enter the current time in UTC (Coordinated Universal Time) format.

**textid**={\\*|textid}

Set the blade or MPA textid. When using '\*', the textid used is the node name specified on the command-line. Note that when specifying an actual textid, only a single node can be specified in the noderange.

**USERID**={newpasswd} **updateBMC**={y|n}

Change the password of the userid **USERID** for CMM in Flex system cluster. The option *updateBMC* can be used to specify whether updating the password of BMCs that connected to the specified CMM. The value is 'y' by default which means whenever updating the password of CMM, the password of BMCs will be also updated. Note that there will be several seconds needed before this command complete.

If value "\*" is specified for USERID and the object node is *Flex System X node*, the password used to access the BMC of the System X node through IPMI will be updated as the same password of the userid **USERID** of the CMM in the same cluster.

**--resetnet**

Reset the network interfaces of the specified nodes.

**v3**

Enable or disable v3 authentication (enable|disable).

**-h | --help**

Prints out a brief usage message.

**-v | --version**

Display the version number.

## EXAMPLES

1. To setup new ssh keys on the Management Module mm:

```
rspconfig mm snmpcfg=enable sshcfg=enable
```

2. To turn on SNMP alerts for node5:

```
rspconfig node5 alert=on
```

Output is similar to:

```
node5: Alerts: enabled
```

3. To display the destination setting for SNMP alerts for node4:

```
rspconfig node4 snmpdest
```

Output is similar to:

```
node4: BMC SNMP Destination 1: 9.114.47.227
```

4. To display the frame number for frame 9A00-10000001

```
rspconfig> 9A00-10000001 frame
```

Output is similar to:



```
9A00-10000001: 1
```

5. To set the frame number for frame 9A00-10000001

```
rspconfig 9A00-10000001 frame=2
```

Output is similar to:

```
9A00-10000001: SUCCESS
```

6. To set the frame numbers for frame 9A00-10000001 and 9A00-10000002

```
rspconfig 9A00-10000001,9A00-10000002 frame=*
```

Output is similar to:

```
9A00-10000001: SUCCESS
9A00-10000002: SUCCESS
```

7. To display the MPA network parameters for mm01:

```
rspconfig mm01 network
```

Output is similar to:

```
mm01: MM IP: 192.168.1.47
mm01: MM Hostname: MM001125C31F28
mm01: Gateway: 192.168.1.254
mm01: Subnet Mask: 255.255.255.224
```

8. To change the MPA network parameters with the values in the xCAT database for mm01:

```
rspconfig mm01 network=*
```

Output is similar to:

```
mm01: MM IP: 192.168.1.47
mm01: MM Hostname: mm01
mm01: Gateway: 192.168.1.254
mm01: Subnet Mask: 255.255.255.224
```

9. To change only the gateway parameter for the MPA network mm01:

```
rspconfig mm01 network=,,192.168.1.1,
```

Output is similar to:

```
mm01: Gateway: 192.168.1.1
```

10. To display the FSP network parameters for fsp01:

```
rspconfig> fsp01 network
```

Output is similar to:

```

fsp01:
  eth0:
    IP Type: Dynamic
    IP Address: 192.168.1.215
    Hostname:
    Gateway:
    Netmask: 255.255.255.0

  eth1:
    IP Type: Dynamic
    IP Address: 192.168.200.51
    Hostname: fsp01
    Gateway:
    Netmask: 255.255.255.0

```

11. To change the FSP network parameters with the values in command line for eth0 on fsp01:

```
rspconfig fsp01 network=eth0,192.168.1.200,fsp01,,255.255.255.0
```

Output is similar to:

```
fsp01: Success to set IP address,hostname,netmask
```

12. To change the FSP network parameters with the values in the xCAT database for eth0 on fsp01:

```
rspconfig fsp01 network=eth0,*
```

Output is similar to:

```
fsp01: Success to set IP address,hostname,gateway,netmask
```

13. To configure eth0 on fsp01 to get dynamic IP address from DHCP server:

```
rspconfig fsp01 network=eth0,0.0.0.0
```

Output is similar to:

```
fsp01: Success to set IP type to dynamic.
```

14. To get the current power redundancy mode for power domain 1 on mm01:

```
rspconfig mm01 pd1
```

Output is similar to:

```
mm01: Redundant without performance impact
```

15. To change the current power redundancy mode for power domain 1 on mm01 to non-redundant:

```
rspconfig mm01 pd1=nonred
```

Output is similar to:

```
mm01: nonred
```

16. To enable NTP with an NTP server address of 192.168.1.1, an update frequency of 90 minutes, and with v3 authentication enabled on mm01:

```
rspconfig mm01 ntp=enable,192.168.1.1,90,enable
```

Output is similar to:

```
mm01: NTP: disabled
mm01: NTP Server: 192.168.1.1
mm01: NTP: 90 (minutes)
mm01: NTP: enabled
```

17. To disable NTP v3 authentication only on mm01:

```
rspconfig mm01 ntp=,,,disable
```

Output is similar to:

```
mm01: NTP v3: disabled
```

18. To disable Predictive Failure and L2 Failure deconfiguration policies on mm01:

```
rspconfig mm01 decfg=disable:predictive,L3
```

Output is similar to:

```
mm01: Success
```

19. To deconfigure processors 4 and 5 of Processing Unit 0 on mm01:

```
rspconfig mm01 procedecfg=deconfigure:0:4,5
```

Output is similar to:

```
mm01: Success
```

20. To check if CEC sysname set correct on mm01:

```
rspconfig mm01 sysname
mm01: mm01
rspconfig mm01 sysname=cec01
mm01: Success
rspconfig mm01 sysname
mm01: cec01
```

21. To check and change the pending\_power\_on\_side value of cec01's fsps:

```
rspconfig cec01 pending_power_on_side
cec01: Pending Power On Side Primary: temp
```

(continues on next page)

(continued from previous page)

```
cec01: Pending Power On Side Secondary: temp
rspconfig cec01 pending_power_on_side=perm
cec01: Success
rspconfig cec01 pending_power_on_side
cec01: Pending Power On Side Primary: perm
cec01: Pending Power On Side Secondary: perm
```

22. To show the BSR allocation for cec01:

```
rspconfig cec01 BSR
```

Output is similar to:

```
cec01: Barrier Synchronization Register (BSR)
cec01: Number of BSR arrays: 256
cec01: Bytes per BSR array : 4096
cec01: Available BSR array : 0
cec01: Partition name: BSR arrays
cec01: lpar01      : 32
cec01: lpar02      : 32
cec01: lpar03      : 32
cec01: lpar04      : 32
cec01: lpar05      : 32
cec01: lpar06      : 32
cec01: lpar07      : 32
cec01: lpar08      : 32
```

23. To query the huge page information for CEC1, enter:

```
rspconfig CEC1 huge_page
```

Output is similar to:

```
CEC1: Huge Page Memory
CEC1: Available huge page memory(in pages): 0
CEC1: Configurable huge page memory(in pages): 12
CEC1: Page Size (in GB): 16
CEC1: Maximum huge page memory(in pages): 24
CEC1: Requested huge page memory(in pages): 15
CEC1: Partition name: Huge pages
CEC1: lpar1       : 3
CEC1: lpar5       : 3
CEC1: lpar9       : 3
CEC1: lpar13      : 3
CEC1: lpar17      : 0
CEC1: lpar21      : 0
CEC1: lpar25      : 0
CEC1: lpar29      : 0
```

24. To request 10 huge pages for CEC1, enter:

```
rspconfig CEC1 huge_page=10
```

Output is similar to:

```
CEC1: Success
```

25. To disable service processor failover for cec01, in order to complete this command, the user should power off cec01 first:

```
rspconfig cec01 setup_failover
cec01: Failover status: Enabled
rpower cec01 off
rspconfig cec01 setup_failover=disable
cec01: Success
rspconfig cec01 setup_failover
cec01: Failover status: Disabled
```

26. To force service processor failover for cec01:

```
lshwconn cec01
cec01: 192.168.1.1: LINE DOWN
cec01: 192.168.2.1: sp=primary,ipadd=192.168.2.1,alt_ipadd=unavailable,
↪state=LINE UP
cec01: 192.168.1.2: sp=secondary,ipadd=192.168.1.2,alt_ipadd=unavailable,
↪state=LINE UP
cec01: 192.168.2.2: LINE DOWN
rspconfig cec01 force_failover
cec01: Success.
lshwconn> cec01
cec01: 192.168.1.1: sp=secondary,ipadd=192.168.1.1,alt_ipadd=unavailable,
↪state=LINE UP
cec01: 192.168.2.1: LINE DOWN
cec01: 192.168.1.2: LINE DOWN
cec01: 192.168.2.2: sp=primary,ipadd=192.168.2.2,alt_ipadd=unavailable,
↪state=LINE UP
```

27. To deconfigure memory bank 9 and 10 of Processing Unit 0 on mm01:

```
rspconfig mm01 memdecfg=deconfigure:bank:0:9,10
```

Output is similar to:

```
mm01: Success
```

28. To reset the network interface of the specified nodes:

```
rspconfig --resetnet
```

Output is similar to:

```
Start to reset network..
Reset network failed nodes:
Reset network succeed nodes:
Server-8233-E8B-SN1000ECP-A, Server-9119-FHA-SN0275995-B, Server-9119-FHA-
↪ SN0275995-A,
Reset network finished.
```

29. To update the existing admin password on fsp:

```
rspconfig fsp admin_passwd=admin,abc123
```

Output is similar to:

```
fsp: Success
```

30. To set the initial password for user HMC on fsp:

```
rspconfig fsp HMC_passwd=,abc123
```

Output is similar to:

```
fsp: Success
```

31. To list BMC dumps available for download:

```
rspconfig p9euh02 dump -l
```

Output is similar to:

```
p9euh02: [1] Generated: 09/06/2017 14:31:49, Size: 4528
p9euh02: [2] Generated: 09/06/2017 14:31:55, Size: 4516
p9euh02: [3] Generated: 09/06/2017 14:32:01, Size: 4236
p9euh02: [4] Generated: 09/06/2017 14:32:07, Size: 4248
p9euh02: [5] Generated: 09/06/2017 14:32:11, Size: 4268
```

32. To generate and download BMC dump:

```
rspconfig p9euh02 dump
```

Output is similar to:

```
Capturing BMC Diagnostic information, this will take some time...
p9euh02: Dump requested. Target ID is 6, waiting for BMC to generate...
```

(continues on next page)

(continued from previous page)

```
p9euh02: Dump 6 generated. Downloading to /var/log/xcat/dump/20171211-0951_
↪p9euh02_dump_6.tar.xz
```

## SEE ALSO

noderange(3)|noderange.3, rpower(1)|rpower.1, rcons(1)|rcons.1, rinvt(1)|rinvt.1, rvitals(1)|rvitals.1, rscan(1)|rscan.1, rflash(1)|rflash.1

## rsreset.1

### Name

**rsreset** - resets the service processors associated with the specified nodes

### Synopsis

**rsreset** *noderange*

**rsreset** [-h | --help | -v | --version]

### Description

**rsreset** resets the service processors associated with the specified nodes. It searches the **nodehm** table and associated tables to find the service processors associated with the nodes specified. If the node is a BMC-based node, the node's BMC will be reset. If the node is a blade, the blade's on board service processor will be reset.

### Options

**-h | --help**

Print help.

**-v | --version**

Print version.

### Examples

1. Reset the service processor that controls node5:

```
rsreset node5
```

## SEE ALSO

rpower(1)|rpower.1, nodehm(5)|nodehm.5

## rvitals.1

### Name

**rvitals** - remote hardware vitals

### Synopsis

**rvitals** [-h | --help | -v | --version]

### FSP/LPAR (with HMC) specific:

**rvitals** *noderange* { temp | voltage | lcds | all }

### CEC/LPAR/Frame (using Direct FSP Management) specific:

**rvitals** *noderange* { rackenv | lcds | all } [-V | --verbose]

### MPA specific:

**rvitals** *noderange* { temp | voltage | wattage | fanspeed | power | leds | summary | all }

### Blade specific:

**rvitals** *noderange* { temp | wattage | fanspeed | leds | summary | all }

### BMC specific:

**rvitals** *noderange* { temp | voltage | wattage | fanspeed | power | leds | all }

### OpenPOWER (IPMI) specific:

**rvitals** *noderange* [temp | voltage | wattage | fanspeed | power | leds | chassis | all]



**OpenPOWER (OpenBMC) specific:**

**rvitals** *noderange* [**temp** | **voltage** | **wattage** | **fanspeed** | **power** | **leds** | **altitude** | **all**] [-V] [--verbose]

**Description**

**rvitals** Retrieves hardware vital information from the on-board Service Processor for a single or range of nodes and groups.

**Options****cputemp**

Retrieves CPU temperatures.

**disktemp**

Retrieves HD back plane temperatures.

**ambtemp**

Retrieves ambient temperatures.

**temp**

Retrieves all temperatures.

**voltage**

Retrieves power supply and VRM voltage readings.

**fanspeed**

Retrieves fan speeds.

**leds**

Retrieves LCDs status.

**rackenv**

Retrieves rack environmentals.

**leds**

Retrieves LEDs status.

**chassis**

Retrieves chassis status.

**altitude**

Retrieves altitude related attributes.

**power**

Retrieves power status.

**powertime**

Retrieves total power uptime. This value only increases, unless the Service Processor flash gets updated. This option is not valid for x86 architecture systems.

**reboot**

Retrieves total number of reboots. This value only increases, unless the Service Processor flash gets updated. This option is not valid for x86 architecture systems.

**state**

Retrieves the system state.

**all**

All of the above.

**-h | --help**

Print help.

**-v | --version**

Print version.

**Examples**

```
rvitals node5 all
```

Output is similar to:

```
node5: CPU 1 Temperature: + 29.00 C (+ 84.2 F)
node5: CPU 2 Temperature: + 19.00 C (+ 66.2 F)
node5: DASH Sensor 1 Temperature: + 32.00 C (+ 89.6 F)
node5: System Ambient Temperature Temperature: + 26.00 C (+ 78.8 F)
node5: +5V Voltage: + 5.01V
node5: +3V Voltage: + 3.29V
node5: +12V Voltage: + 11.98V
node5: +2.5V Voltage: + 2.52V
node5: VRM1 Voltage: + 1.61V
node5: VRM2 Voltage: + 1.61V
node5: Fan 1 Percent of max: 100%
node5: Fan 2 Percent of max: 100%
node5: Fan 3 Percent of max: 100%
node5: Fan 4 Percent of max: 100%
node5: Fan 5 Percent of max: 100%
node5: Fan 6 Percent of max: 100%
node5: Current Power Status On
node5: Current LCD1: SuSE Linux
node5: Power On Seconds 11855915
node5: Number of Reboots 930
node5: System State Booting OS or in unsupported OS
```

## SEE ALSO

rpower(1)|rpower.1, rinv(1)|rinv.1

## sinv.1

## NAME

**sinv** - Checks the software configuration of the nodes in the cluster.

## SYNOPSIS

**sinv** [-o *output*] -p *template path* [-t *template count*] [-s *seed node*] [-i] [-e] [-r] [-V] [--devicetype *type\_of\_device*] [-l *userID*] {-f *command file* | -c *command*}

**sinv** [-h | -v]

## DESCRIPTION

The **sinv** command is designed to check the configuration of the nodes in a cluster. The command takes as input command line flags, and one or more templates which will be compared against the output of the **xdsh** command, designated to be run by the -c or -f flag, on the nodes in the noderange.

The nodes will then be grouped according to the template they match and a report returned to the administrator in the output file designated by the -o flag, or to stdout.

**sinv** supports checking the output from the **rinv** or **xdsh** command.

The **sinv** command is an xCAT Distributed Shell Utility.

### COMMAND SPECIFICATION:

The **xdsh** or **rinv** command to execute on the remote targets is specified by the -c flag, or by the -f flag which is followed by the fully qualified path to a file containing the command.

Note: do not add | **xdshcoll** to the command on the command line or in the command file, it is automatically added by **sinv**.

The syntax for the -c parameter is as follows:

“*command*[; *command*].…”

where *command* is the command to run on the remote target. Quotation marks are required to ensure that all commands in the list are executed remotely, and that any special characters are interpreted correctly on the remote target.

The **sinv** command does not work with any interactive commands, including those that read from standard input.

### REMOTE SHELL COMMAND:

For **xdsh**, support is explicitly provided for AIX Remote Shell and OpenSSH, but any secure remote command that conforms to the IETF (Internet Engineering Task Force) Secure Remote Command Protocol can be used. See man **xdsh** for more details.

## OPTIONS

### **-o | --output** *report output file*

Optional output file. This is the location of the file that will contain the report of the nodes that match, and do not match, the input templates. If the flag is not used, the output will go to stdout.

### **-p | --tp** *template path*

This is the path to the template file. The template contains the output of **xdsh** or **rinv** command, that has been run against a “seed” node, a node that contains the configuration that you would like all nodes in your noderange to match.

The admin can create the template by running the **xdsh** or **rinv** command on the seed node, pipe to **xdshcoll** (required) and store the output in the template path. See examples.

**Note:** The admin can also edit the template to remove any lines that they do not want checked.

An alternative method is to use the **[-s seed node]** parameter, which will automatically build the template for you from the seed node named.

If a seed node is not provided, then command will automatically use the first node in the noderange as the seed node.

### **-t | --tc** *template count*

This count is the number of templates that the command will use to check for nodes matches. If the template in the template path does not match a node, the **sinv** will check additional templates up to the template count.

For each node, it will compare the node against each template to see if there is a match. If there is no match, and we are not over the template count, then a new template will be created from the node output. This will result in having all nodes that match a given template reported in their group at the end of the run in the output file. If no template count is specified, 0 is the default, and all nodes will be compared against the first template.

### **-s | --seed** *seed node*

This is the node that will be used to build the first template that is stored in template path. You can use this parameter instead of running the command yourself to build the template.

**Note:** If no seed node is supplied, the first node in the noderange is automatically selected as a seed node.

### **-i | --ignorefirst**

This flag suppresses the reporting of the nodes matching the first template. In very large systems, you may not want to show the nodes that have the correct configuration, since the list could contain thousands of nodes. This allows you to only report the nodes that do not match the required configuration.

### **-e | --exactmatch**

This requires the check of node output against template to be an exact match. If this flag is not set, **sinv** checks to see if the return from the **xdsh** or **rinv** command to the nodes contain a match for each line in the input template (except for **xdshcoll** header and comments). If not in exactmatch mode, there can be more lines in the **xdsh** or **rinv** return from the nodes.

For example, if running a “**rpm -qa | grep xCAT**” command, without exactmatch set, if the node contains more xCAT rpms that listed in the template, it would be considered a match, as long as all rpms listed in the template were on the node. With exactmatch set, the output must be identical to the template.

### **--devicetype** *type\_of\_device*

Specify a user-defined device type that references the location of relevant device configuration file. The `devicetype` value must correspond to a valid device configuration file. xCAT ships some default configuration files for Ethernet switches and IB switches under `/opt/xcat/share/xcat/devicetype` directory. If you want to overwrite any of the configuration files, copy them to `/var/opt/xcat/` directory and customize. For example, `base/IBSwitch/Qlogic/config` is the configuration file location if `devicetype` is specified as `IB-Switch::Qlogic`. xCAT will first search config file using `/var/opt/xcat/` as the base. If not found, it will search for it using `/opt/xcat/share/xcat/devicetype/` as the base.

#### **-l | --user *user\_ID***

Specifies a remote user name to use for remote command execution.

#### **-c | --command**

The **xdsh** or **rinv** command that will be run. The command should be enclosed in double quotes to insure correct shell interpretation. This parameter must only contain, the node range or the image path (Linux) or spot name for AIX. It cannot be used to set additional input flags to **xdsh** or **rinv** (for example **-s,-T,-e**). See examples below.

**Note:** do not add the `| xdshcoll` to the command, it is automatically added by **sinv**. **sinv** also automatically sets the **-v** flag for **xdsh**.

#### **-f | --file**

The file containing the **xdsh** or **rinv** command that will be run. This should be the fully qualified name of the file.

**Note:** do not add the `| xdshcoll` to the command in the file, it is automatically added by **sinv**.

#### **-r | --remove**

This flag indicates that generated templates should be removed at the at the end of the **sinv** command execution.

If the flag is specified, then all templates that are generated by the **sinv** command, will be removed. If the first template is created by the admin, it will not be removed.

If the flag is not specified, no templates will be removed. It is up to the admin to cleanup templates.

#### **-h | --help**

Displays usage information.

#### **-v | --version**

Displays xCAT release version.

#### **-V | --verbose**

Verbose mode.

## Examples

1. To setup `sinv.template` (name optional) for input to the **sinv** command, enter:

```
xdsh node1,node2 "rpm -qa | grep ssh " | xdshcoll > /tmp/sinv.template
```

Note: when setting up the template the output of **xdsh** must be piped to **xdshcoll**, **sinv** processing depends on it.

2. To setup `rinv.template` for input to the **sinv** command , enter:

```
rinv node1-node2 serial | xdshcoll > /tmp/rinv.template
```

Note: when setting up the template the output of **rinv** must be piped to **xdshcoll**, **sinv** processing depends on it.

3. To execute **sinv** using the **sinv.template** generated above on the nodegroup, *testnodes*, possibly generating up to two new templates, and removing all generated templates in the end, and writing output report to **/tmp/sinv.output**, enter:

```
sinv -c "xdsh testnodes rpm -qa | grep ssh" -p /tmp/sinv.template -t 2 -r -o /  
→tmp/sinv.output
```

Note: do not add the pipe to **xdshcoll** on the **-c** flag, it is automatically added by the **sinv**.

4. To execute **sinv** on noderange, *node1-node4*, using the seed node, *node8*, to generate the first template, using the **xdsh** command (**-c**), possibly generating up to two additional templates and not removing any templates at the end, enter:

```
sinv -c "xdsh node1-node4 lslpp -l | grep bos.adt" -s node8 -p /tmp/sinv.  
→template -t 2 -o /tmp/sinv.output
```

5. To execute **sinv** on noderange, *node1-node4*, using the seed node, *node8*, to generate the first template, using the **rinv** command (**-c**), possibly generating up to two additional templates and removing any generated templates at the end, enter:

```
sinv -c "rinv node1-node4 serial" -s node8 -p /tmp/sinv.template -t 2 -r -o /  
→tmp/rinv.output
```

6. To execute **sinv** on noderange, *node1-node4*, using *node1* as the seed node, to generate the **sinv.template** from the **xdsh** command (**-c**), using the exact match option, generating no additional templates, enter:

```
sinv -c "xdsh node1-node4 lslpp -l | grep bos.adt" -s node1 -e -p /tmp/sinv.  
→template -o /tmp/sinv.output
```

Note: the **/tmp/sinv.template** file must be empty, otherwise it will be used as an admin generated template.

7. To execute **sinv** on the Linux osimage defined for *cn1*. First build a template from the **/etc/hosts** on the node. Then run **sinv** to compare.

```
xdsh cn1 "cat /etc/hosts" | xdshcoll > /tmp/sinv2/template"  
  
sinv -c "xdsh -i /install/netboot/rhels6/ppc64/test_ramdisk_statelite/rootimg.  
→cat /etc/hosts" -e -t 1 -p /tmp/sinv.template -o /tmp/sinv.output
```

8. To execute **sinv** on the AIX NIM 611dskls spot and compare **/etc/hosts** to *compute1* node, run the following:

```
xdsh compute1 "cat /etc/hosts" | xdshcoll > /tmp/sinv2/template"  
  
sinv -c "xdsh -i 611dskls cat /etc/hosts" -e -t1 -p /tmp/sinv.template -o /tmp/  
→sinv.output
```

9. To execute **sinv** on the device *mswitch2* and compare to *mswitch1*

```
sinv -c "xdsh mswitch enable;show version" -s mswitch1 -p /tmp/sinv/template -  
→-devicetype IBSwitch::Mellanox -l admin -t 2
```

## Files

/opt/xcat/bin/sinv/

Location of the sinv command.

## SEE ALSO

xdsh(1)|xdsh.1, noderange(3)|noderange.3

## snmove.1

## NAME

**snmove** - Move xCAT compute nodes to a different xCAT service node.

## SYNOPSIS

**snmove** *noderange* [-V] [-l | --liteonly] [-d | --dest *sn2*] [-D | --destn *sn2n*] [-i | --ignorenodes] [-P | --postscripts *script1,script2...* | **all**]

**snmove** [-V] [-l | --liteonly] -s | --source *sn1* [-S | --sourcen *sn1n*] [-d | --dest *sn2*] [-D | --destn *sn2n*] [-i | --ignorenodes] [-P | --postscripts *script1,script2...* | **all**]

**snmove** [-h | --help | -v | --version]

## DESCRIPTION

The **snmove** command may be used to move a node or nodes from one service node to another backup service node.

The use of backup service nodes in an xCAT hierarchical cluster can help improve the overall reliability, availability, and serviceability of the cluster.

Before you run the **snmove** command it is assumed that the backup service node has been configured properly to manage the new node or nodes. (See the xCAT document named “Using xCAT Service Nodes with AIX” for information on how to set up backup AIX service nodes.).

The **snmove** command can use the information stored in the xCAT database or information passed in on the command line to determine the current service node and the backup service node.

To specify the primary and backup service nodes you can set the “servicenode” attribute of the node definitions.

The **servicenode** attribute is the hostname of the xCAT service node as it is known by the management node. The **xcatmaster** attribute is the hostname of the xCAT service node as known by the node. The **servicenode** attribute should be set to a comma-separated list so that the primary service node is first and the backup service node is second. The **xcatmaster** attribute must be set to the hostname of the primary service node as it is known by the node.

When the **snmove** command is run it modifies the xCAT database to switch the primary server to the backup server.

It will also check the other services that are being used for the node (tftpserver, monserver, nfsserver, conserver), and if they were set to the original service node they will be changed to point to the backup service node.

By default the command will modify the nodes so that they will be able to be managed by the backup service node.

If the **-i** option is specified, the nodes themselves will not be modified.

You can also have postscripts executed on the nodes by using the **-P** option if needed.

The xCAT **snmove** command may also be used to synchronize statelite persistent files from the primary service node to the backup service node without actually moving the nodes to the backup servers.

If you run the command with the “-l” option it will attempt to use rsync to update the statelite persistent directory on the backup service node. This will only be done if the server specified in the “statelite” table is the primary service node.

When the **snmove** command is executed the new service node must be running but the original service node may be down.

Note: On a Linux cluster, for NFS statelite nodes that do not use external NFS server, if the original service node is down, the nodes it manages will be down too. You must run nodeset command and then reboot the nodes after running snmove. For stateless nodes and RAMDisk statelite nodes, the nodes will be up even if the original service node is down. However, make sure to run nodeset command if you decide to reboot the nodes later.

## OPTIONS

### **-d|--dest**

Specifies the hostname of the new destination service node as known by (facing) the management node.

### **-D|--destn**

Specifies the hostname of the destination service node as known by (facing) the nodes.

### **-h|--help**

Display usage message.

### **-i|--ignorenodes**

No modifications will be made on the nodes. If not specified, several xCAT postscripts will be run on the nodes to complete the switch to the new service node.

### **-l|--liteonly**

Use this option to ONLY synchronize any AIX statelite files from the primary server to the backup server for the nodes. It will not do the actual moving of the nodes to the backup servers.

### **-P|--postscripts**

Specifies a list of extra postscripts to be run on the nodes after the nodes are moved over to the new service node. If **all** is specified, all the postscripts defined in the postscripts table will be run for the nodes. The specified postscripts must be stored under /install/postscripts directory.

### **-s|--source**

Specifies the hostname of the current (source) service node as known by (facing) the management node.

### **-S|--sourcen**

Specifies the hostname of the current service node adapter as known by (facing) the nodes.

### **-V|--verbose**

Verbose mode.

### **-v|--version**

Command Version.



## EXAMPLES

1. Move the nodes contained in group “group1” to the service node named “xcatsn02”.

```
snmove group1 -d xcatsn02 -D xcatsn02-eth1
```

2. Move all the nodes that use service node xcatsn01 to service node xcatsn02.

```
snmove -s xcatsn01 -S xcatsn01-eth1 -d xcatsn02 -D xcatsn02-eth1
```

3. Move any nodes that have sn1 as their primary server to the backup service node set in the xCAT node definition.

```
snmove -s sn1
```

4. Move all the nodes in the xCAT group named “nodegroup1” to their backup SNs.

```
snmove nodegroup1
```

5. Move all the nodes in xCAT group “sngroup1” to the service node named “xcatsn2”.

```
snmove sngroup1 -d xcatsn2
```

6. Move all the nodes in xCAT group “sngroup1” to the SN named “xcatsn2” and run extra postscripts.

```
snmove sngroup1 -d xcatsn2 -P test1
```

7. Move all the nodes in xCAT group “sngroup1” to the SN named “xcatsn2” and do not run anything on the nodes.

```
snmove sngroup1 -d xcatsn2 -i
```

8. Synchronize any AIX statelite files from the primary server for compute03 to the backup server. This will not actually move the node to it’s backup service node.

```
snmove compute03 -l -V
```

## FILES

/opt/xcat/sbin/snmove

## SEE ALSO

noderange(3)|noderange.3

## **swapnodes.1**

### **NAME**

**swapnodes** - swap the location info in the db (all the attributes in the ppc table and the nodepos table) between 2 nodes. If swapping within a cec, it will assign the IO adapters that were assigned to the defective node to the available node.

### **SYNOPSIS**

**swapnodes** [-h| --help]

**swapnodes** -c *current\_node* -f *fip\_node* [-o]

### **DESCRIPTION**

This command is only for Power 775 using Direct FSP Management, and used in Power 775 Availability Plus.

The **swapnodes** command will keep the **current\_node** name in the xCAT table, and use the *fip\_node*'s hardware resource. Besides that, the IO adapters will be assigned to the new hardware resource if they are in the same CEC. So the swapnodes command will do 2 things:

1. Swap the location info in the db between 2 nodes:

All the ppc table attributes (including hcp, id, parent, supernode and so on). All the nodepos table attributes (including rack, u, chassis, slot, room and so on).

2. Assign the I/O adapters from the defective node (the original *current\_node*) to the available node (the original *fip\_node*) if the nodes are in the same cec.

The **swapnodes** command shouldn't make the decision of which 2 nodes are swapped. It will just received the 2 node names as cmd line parameters.

After running **swapnodes** command, the order of the I/O devices may be changed after IO re-assignment, so the administrator needs to run **rbootseq** to set the boot string for the *current\_node*. And then boot the node with the same image and same postscripts because they have the same attributes.

Without -o option, it's used to swap the location info in the db between 2 nodes. With -o option, it's used to move the *current\_node* definition to *fip\_node* (the 2nd octant), not move the *fip\_node* definition to the 1st octant. If the two nodes are in a cec, it will assign the IO adapters that were assigned to the defective node to the available node. Originally, the *current\_node* is a defective non-compute node, and *fip\_node* is a available compute node. After the swapping, the *current\_node* will be a available node.

### **OPTIONS**

**-h|--help**

Display usage message.

**-c**

*current\_node* – the defective non-compute node.

**-f**

*fip\_node* – a compute node which will be swapped as the non-compute node.

**-o**

one way. Only move the *current\_node* definition to the *fip\_node*'s hardware resource, and not move the *fip\_node* definition to the *current\_node*. And then the *current\_node* will use the *fip\_node*'s hardware resource, and the *fip\_node* definition is not changed. if the two nodes are in the same CEC, the I/O adapter from the original *current\_node* will be assigned to the *fip\_node*.

## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To swap the service node attributes and IO assignments between sn1 and compute2 which are in the same cec, all the attributes in the ppc table and nodepos table of the two node will be swapped, and the I/O adapters from the defective node (the original sn1) will be assigned to the available node (the original compute2). After the swapping, the sn1 will use the compute2's hardware resource and the I/O adapters from the original sn1.

```
swapnodes -c sn1 -f compute2
```

2. To swap the service node attributes and IO assignments between sn1 and compute2 which are NOT in the same cec, all the attributes in the ppc table and nodepos table of the two node will be swapped. After the swapping, the sn1 will use the compute2's hardware resource.

```
swapnodes -c sn1 -f compute2
```

3. Only to move the service node (sn1) definition to the compute node (compute2)'s hardware resource, and not move the compute2 definition to the sn1. After the swapping, the sn1 will use the compute2's hardware resource, and the compute2 definition is not changed.

```
swapnodes -c sn1 -f compute2 -o
```

## FILES

\$XCATROOT/bin/swapnodes

(The XCATROOT environment variable is set when xCAT is installed. The default value is "/opt/xcat".)

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

lsvm(1)|lsvm.1, mkvm(1)|mkvm.1, chvm(1)|chvm.1

## switchblade.1

## SYNOPSIS

**switchblade** *MM* {**list** | **stat**}

**switchblade** *node* {**media** | **mt** | **kvm** | **video** | **both**} [*slot\_num*]

**switchblade** [-h | --help | -v | --version]

## DESCRIPTION

**switchblade** assigns the BladeCenter media tray and/or KVM to the specified blade, so that they can be used with that blade. If **list** or **stat** are specified instead, **switchblade** will display the current assignment. You can either specify a management module or a node (blade) to **switchblade**. If the latter, **switchblade** will determine the management module of the node.

## OPTIONS

### list | stat

Display which blade the media tray and KVM are currently assigned to.

### media | mt

Assign the media tray to the specified blade.

### kvm | video

Assign the KVM (video display) to the specified blade.

### both

Assign both the media tray and the KVM to the specified blade.

### slot\_num

The slot # of the blade that the resources should be assigned to. If not specified, it will use the slot # of the node specified.

## EXAMPLES

1. Switch the media tray to be assigned to the blade in slot 4 (assume it is node4):

```
switchblade node4 media
```

Output will be like:

```
Media Tray slot: 4
```

## SEE ALSO

rbootseq(1)|rbootseq.1

## switchdiscover.1

## SYNOPSIS

**switchdiscover** [-h] --help]

**switchdiscover** [-v] --version]

**switchdiscover** [*noderange* | --range *ip\_ranges*] [-V] [-w][-r|-x|-z][-s *scan\_methods*] [--setup] [-c *community*]

## DESCRIPTION

The switchdiscover command scans the subnets and discovers all the switches on the subnets. The command takes a list of subnets as input. The default subnets are the ones that the xCAT management node is on. It uses nmap command as default to discover the switches. However, you can specify other discovery methods such as lldp or snmp with -s flag. You can write the discovered switches into xCAT database with -w flag. This command supports may output formats such as xml(-x), raw(-r) and stanza(-z) in addition to the default format.

--setup flag is for switch-based switch discovery. It will find all the discovered switches on the subnets, then match them with predefined switches in the xCATDB. Next, it will set discovered switches with static ip address and hostname based on the predefined switch. It will also enable snmpv3 configuration. The details of the process are defined in the [http://xcat-docs.readthedocs.io/en/latest/advanced/networks/switchdiscover/switches\\_discovery.html](http://xcat-docs.readthedocs.io/en/latest/advanced/networks/switchdiscover/switches_discovery.html).

To view all the switches defined in the xCAT database use **lsdef -w "nodetype=switch"** command.

For lldp method, make sure that lldpd package is installed and lldpd is running on the xCAT management node. lldpd comes from xcat-dep package or you can get it from <http://vincentbernat.github.io/lldpd/installation.html>.

For snmp method, make sure that snmpwalk command is installed and snmp is enabled for switches. To install snmpwalk, "yum install net-snmp-utils" for redhat and sles, "apt-get install snmp" for Ubuntu. The switchdiscover command only check the switches with default community string, if user already configured switch with other community string, need to pass in with -c option for switchdiscover command to be able to discover.

## OPTIONS

*noderange*

The switches which the user want to discover. If the user specify the noderange, switchdiscover will just return the switches in the node range. Which means it will help to add the new switches to the xCAT database without modifying the existed definitions. But the switches' name specified in noderange should be defined in database in advance. The ips of the switches will be defined in /etc/hosts file. This command will fill the switch attributes for the switches defined.

**-h|--help**

Display usage message.

**--range**

Specify one or more IP ranges. Each can be an ip address (10.1.2.3) or an ip range (10.1.2.0/24). If the range is huge, for example, 192.168.1.1/8, the switch discover may take a very long time to scan. So the range should be exactly specified.

For nmap and snmp scan method, it accepts multiple formats. For example: 192.168.1.1/24, 40-41.1-2.3-4.1-100.

If the range is not specified, the command scans all the subnets that the active network interfaces (eth0, eth1) are on where this command is issued.

**-r**

Display Raw responses.

**-s**

It is a comma separated list of methods for switch discovery. The possible switch scan methods are: lldp, nmap or snmp. The default is nmap.

**-v|--version**

Command Version.

**-V**

Verbose output.

**-w**

Writes output to xCAT database.

**-x**

XML formatted output.

**-z**

Stanza formatted output.

**--setup**

Process switch-based switch discovery. Update discovered switch's ip address, hostname and enable snmpv3 configuration based on the predefined switch.

**-c**

User defined community string for snmp scan.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To discover the switches on some subnets:

```
switchdiscover --range 10.2.3.0/24,192.168.3.0/24,11.5.6.7
```

2. To do the switch discovery and save them to the xCAT database:

```
switchdiscover --range 10.2.3.4/24 -w
```

It is recommended to run **makehosts** after the switches are saved in the DB.

3. To use lldp method to discover the switches:

```
switchdiscover -s lldp
```

4. To process switch-based switch discovery, the core switch has to be configured and top-of-rack (edge) switch has to be predefined into xCAT database with attribute **switch** and **switchport** to core switch:

```
switchdiscover --range 192.168.5.150-170 -s snmp --setup
```

## FILES

/opt/xcat/bin/switchdiscover

## SEE ALSO

**tabgrep.1**

## NAME

**tabgrep** - list table names in which an entry for the given node appears.

## SYNOPSIS

**tabgrep** *nodename*

**tabgrep** [-? | -h | --help]

## DESCRIPTION

The **tabgrep** command displays the tables that contain a row for the specified node. Note that the row can either have that *nodename* as the key or it could have a group that contains the node as the key.

## OPTIONS

**-?|-h|--help**

Display usage message.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To display the tables that contain blade1:

```
tabgrep blade1
```

The output would be similar to:

```
nodelist  
nodehm  
mp  
chain  
hosts  
mac  
noderes  
nodetype
```

## FILES

/opt/xcat/bin/tabgrep

## SEE ALSO

models(1)|models.1, tabdump(8)|tabdump.8

## unregnotif.1

## NAME

**unregnotif** - unregister a Perl module or a command that was watching for the changes of the desired xCAT database tables.

## SYNOPSIS

**unregnotif** [-h| --help]

**unregnotif** [-v| --version]

**unregnotif** *filename*



## DESCRIPTION

This command is used to unregister a Perl module or a command that was watching for the changes of the desired xCAT database tables.

## PARAMETERS

*filename* is the path name of the Perl module or command to be registered.

## OPTIONS

**-h | -help** Display usage message.

**-v | -version** Command Version.

**-V | -verbose** Verbose output.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To unregister a Perl module, enter:

```
unregnotif /opt/xcat/lib/perl/xCAT_monitoring/mycode.pm
```

2. To register a command, enter:

```
unregnotif /usr/bin/mycmd
```

## FILES

/opt/xcat/bin/unregnotif

## SEE ALSO

regnotif(1)|regnotif.1

## updateSNimage.1

### NAME

**updateSNimage** - Adds the needed Service Node configuration files to the install image.

### SYNOPSIS

**updateSNimage** [-h | --help ]

**updateSNimage** [-v | --version]

**updateSNimage** [-n *node*] [-p *path*]

### DESCRIPTION

This command is used to add the Service Node configuration files to the install image. It will either copy them locally or scp them to a remote host.

### OPTIONS

**-h | --help** Display usage message.

**-v | --version** Display xCAT version.

**-n | --node** A remote host name or ip address that contains the install image to be updated.

**-p | --path** Path to the install image.

### RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

### EXAMPLES

1. To update the image on the local host.

```
updateSNimage -p /install/netboot/fedora8/x86_64/test/rootimg
```

2. To update the image on a remote host.

```
updateSNimage -n 9.112.45.6 -p /install/netboot/fedora8/x86_64/test/rootimg
```

## updatenode.1

### NAME

**updatenode** - Update nodes in an xCAT cluster environment.

### SYNOPSIS

**updatenode** *noderange* [-V | --verbose] [-F | --sync] [-f | --snsync] [-r | --node-rcp [*node\_remote\_copy\_command*]] [-S | --sw] [-l *userID*] [-P | --scripts [*script1,script2...*]] [-s | --sn] [-A | --updateallsw] [-c | --cmdlineonly] [-d *alt\_source\_dir*] [--fanout=*fanout\_value*] [-t *timeout*] [*attr=val* [*attr=val...*]] [-n | --noverify]

**updatenode** *noderange* [-k | --security] [-t *timeout*]

**updatenode** *noderange* [-g | --genmypost]

**updatenode** *noderange* [-V | --verbose] [-t *timeout*] [*script1,script2...*]

**updatenode** *noderange* [-V | --verbose] [-f | --snsync]

**updatenode** [-h | --help] [-v | --version]

### DESCRIPTION

The **updatenode** command is run on the xCAT management node and can be used to perform the following node updates:

1. Distribute and synchronize files.
2. Install or update software on diskful nodes.
3. Run postscripts.
4. Update the ssh keys and host keys for the service nodes and compute nodes; Update the ca and credentials for the service nodes.

The default behavior when no options are input to **updatenode** will be to run the following options **-S**, **-P** and **-F** options in this order. If you wish to limit **updatenode** to specific actions you can use combination of the **-S**, **-P**, and **-F** flags.

For example, If you just want to synchronize configuration file you could specify the **-F** flag. If you want to synchronize files and update software you would specify the **-F** and **-S** flags. See the descriptions of these flags and examples below.

The flag **-k** (**--security**) can NOT be used together with **-S**, **-P**, and **-F** flags.

The flag **-f** (**--snsync**) can NOT be used together with **-S**, **-P**, and **-F** flags.

Note: In a large cluster environment the updating of nodes in an ad hoc manner can quickly get out of hand, leaving the system administrator with a very confusing environment to deal with. The **updatenode** command is designed to encourage users to handle cluster updates in a manner that is recorded and easily repeatable.

## To distribute and synchronize files

The basic process for distributing and synchronizing nodes is:

- \* Create a synclist file.
- \* Indicate the location of the synclist file.
- \* Run the **updatenode** command to update the nodes.

Files may be distributed and synchronized for both diskless and diskful nodes. Syncing files to NFS-based statelite nodes is not supported.

More information on using the synchronization file function is in the following document: [https://xcat-docs.readthedocs.io/en/stable/guides/admin-guides/manage\\_clusters/common/deployment/syncfile/syncfile\\_updatenode.html](https://xcat-docs.readthedocs.io/en/stable/guides/admin-guides/manage_clusters/common/deployment/syncfile/syncfile_updatenode.html)

## Create the synclist file

The synclist file contains the configuration entries that specify where the files should be synced to. In the synclist file, each line is an entry which describes the location of the source files and the destination location for the files on the target node.

For more information on creating your synclist files and where to put them, read here: [https://xcat-docs.readthedocs.io/en/stable/guides/admin-guides/manage\\_clusters/common/deployment/syncfile/syncfile\\_synclist\\_file.html](https://xcat-docs.readthedocs.io/en/stable/guides/admin-guides/manage_clusters/common/deployment/syncfile/syncfile_synclist_file.html)

## Run updatenode to synchronize the files

```
updatenode <noderange> -F
```

## To install or update software

updatenode can be use to install or update software on the nodes. See the following documentation for setting up otherpkgs: `Install_Additional_Packages`

To install/update the packages, run:

```
updatenode <noderange> -S
```

### For Linux systems:

It this is equivalent to running the following command:

```
updatenode noderange -P ospkgs,otherpkgs
```

It will update all the rpms specified in the .pkglist file and .otherpkgs.pkglist file. ospkgs postscript will normally remove all the existing rpm repositories before adding server:/install/<os>/<arch/ as the new repository. To preserve the existing repositories, you can run the following command instead:

```
updatenode noderange -P "ospkgs --keeprepo,otherpkgs"
```

### For AIX systems:

Note: The `updatenode` command is used to update AIX diskful nodes only. For updating diskless AIX nodes refer to the xCAT for AIX update documentation and use the `xCAT mknimimage` command. For information on updating software on AIX cluster: For diskful installs, read: `XCAT_AIX_RTE_Diskful_Nodes` For diskless installs, read: `XCAT_AIX_Diskless_Nodes`

`updatenode` can also be used in Sysclone environment to push delta changes to target node. After capturing the delta changes from the golden client to management node, just run below command to push delta changes to target nodes.

```
updatenode <targetnoderange> -S
```

## To run postscripts

The scripts must be copied to the `/install/postscripts` directory on the xCAT management node. (Make sure they are executable and world readable.)

To run scripts on a node you must either specify them on the command line or you must add them to the “postscripts” attribute for the node.

To set the postscripts attribute of the node (or group) definition you can use the xCAT `chdef` command. Set the value to be a comma separated list of the scripts that you want to be executed on the nodes. The order of the scripts in the list determines the order in which they will be run. You can use the `lsdef` command to check the postscript order.

Scripts can be run on both diskless and diskful nodes.

To run all the customization scripts that have been designated for the nodes, (in the “postscripts and postbootscripts” attributes), type:

```
updatenode <noderange> -P
```

To run the “syslog” script for the nodes, type:

```
updatenode <noderange> -P syslog
```

To run a list of scripts, type:

```
updatenode <noderange> -P "script1 p1 p2,script2"
```

where `p1 p2` are the parameters for `script1`.

The flag ‘-P’ can be omitted when only scripts names are specified.

Note: `script1,script2` may or may not be designated as scripts to automatically run on the node. However, if you want `script1` and `script2` to get invoked next time the nodes are deployed then make sure to add them to the “postscripts/postbootscripts” attribute in the database for the nodes.

## Update security

The basic functions of update security for nodes:

- \* Setup the ssh keys for the target nodes. It enables the management node and service nodes to ssh to the target nodes without password.
- \* Redeliver the host keys to the target nodes.
- \* Redeliver the ca and certificates files to the service node. These files are used to authenticate the ssl connection between `xcatd`’s of management node and service node.
- \* Remove the entries of target nodes from `known_hosts` file.

### *Set up the SSH keys*

A password for the user who is running this command is needed to setup the ssh keys. This user must have the same uid and gid as the userid on the target node where the keys will be setup.

If the current user is root, root's public ssh keys will be put in the `authorized_keys*` files under root's `.ssh` directory on the node(s). If the current user is non-root, the user must be in the policy table and have credential to run the `xdsh` command. The non-root users public ssh keys and root's public ssh keys will be put in the `authorized_keys*` files under the non-root users `.ssh` directory on the node(s).

### *Handle the hierarchical scenario*

When update security files for the node which is served by a service node, the service node will be updated automatically first, and then the target node.

The certificates files are needed for a service node to authenticate the ssl connections between the xCAT client and `xcatd` on the service node, and the `xcatd`'s between service node and management node. The files in the directories `/etc/xcat/cert/` and `~/.xcat/` will be updated.

Since the certificates have the validity time, the `ntp` service is recommended to be set up between management node and service node.

Simply running following command to update the security keys:

```
updatenode <noderange> -k
```

## PARAMETERS

### *noderange*

A set of comma delimited xCAT node names and/or group names. See the xCAT “noderange” man page for details on additional supported formats.

### *script1,script2...*

A comma-separated list of script names. The scripts must be executable and copied to the `/install/postscripts` directory. Each script can take zero or more parameters. If parameters are specified, the whole list needs to be quoted by double quotes. For example:

```
"script1 p1 p2,script2"
```

### *[attr=val [attr=val...]]*

Specifies one or more “attribute equals value” pairs, separated by spaces. `Attr=val` pairs must be specified last on the command line. The currently supported attributes are: “`installp_bundle`”, “`otherpkgs`”, “`installp_flags`”, “`emgr_flags`” and “`rpm_flags`”. These attributes are only valid for AIX software maintenance support.

## OPTIONS

### **--fanout**=*fanout\_value*

Specifies a fanout value for the maximum number of concurrently executing remote shell processes. Serial execution can be specified by indicating a fanout value of **1**. If **--fanout** is not specified, a default fanout value of **64** is used.

### **-A|--updateallsw**

Install or update all software contained in the source directory. (AIX only)

### **-c|--cmdlineonly**

Specifies that the `updatenode` command should only use software maintenance information provided on the command line. This flag is only valid when using AIX software maintenance support.

### **-d** *alt\_source\_dir*

Used to specify a source directory other than the standard `lpp_source` directory specified in the xCAT osimage definition. (AIX only)

### **-F|--sync**

Specifies that file synchronization should be performed on the nodes. **rsync/scp** and **ssh** must be installed and configured on the nodes. The function is not supported for NFS-based statelite installations. For NFS-based statelite installations to sync files, you should use the read-only option for files/directories listed in **litefile** table with source location specified in the **litetree** table.

### **-f|--snsync**

Specifies that file synchronization should be performed to the service nodes that service the nodes in the noderange. This updates the service nodes with the data to sync to the nodes. **rsync/scp** and **ssh** must be installed and configured on the service nodes. For hierarchy, this optionally can be done before syncing the files to the nodes with the **-F** flag. If the **-f** flag is not used, then the **-F** flag will sync the service nodes before the nodes automatically. When installing nodes in a hierarchical cluster, this flag should be used to sync the service nodes before the install, since the files will be synced from the service node by the **synfiles** postscript during the install. The function is not supported for NFS-based statelite installations. For statelite installations to sync files, you should use the read-only option for files/directories listed in **litefile** table with source location specified in the **litetree** table.

### **[-r | --node-rcp** [*node\_remote\_copy\_command*]]

Specifies the full path of the remote copy command used for syncing files to node targets, such as `/usr/bin/rsync` or `/usr/bin/scp`. If not specified, **rsync** will be used by default.

Note: The synclist processing for **-r /usr/bin/scp** has some differences with **-r /usr/bin/rsync**:

- 1) the **EXECUTE** clause in synclist file is not supported with **-r /usr/bin/scp** flag
- 2) if the destination directory specified in synclist file is an existing file on target node, **updatenode -r /usr/bin/scp** will fail with “scp: <destination directory>: Not a directory”
- 3) if the destination file specified in synclist file is an existing directory on target node, **updatenode -r /usr/bin/scp** will fail with “scp: <destination file>: Is a directory”

### **-gl|--genmypost**

Will generate a new mypostscript file for the nodes in the noderange, if `site precreatemy postscrips` is 1 or YES.

### **-h|--help**

Display usage message.

**-k|--security**

Update the ssh keys and host keys for the service nodes and compute nodes; Update the ca and credentials to the service nodes. Never run this command to the Management Node, it will take down xcatd. You must be running updatenode as root to use the -k flag.

**-l | --user *user\_ID***

Specifies a non-root user name to use for remote command execution. This option is only available when running postscripts (-P) for AIX and Linux and updating software (-S) for Linux only. The non-root userid must be previously defined as an xCAT user. The userid sudo setup will have to be done by the admin on the node. This is not supported in a hierarchical cluster, that is the node is serviced by a service node. See the document <https://xcat-docs.readthedocs.io/en/stable/advanced/security/security.html#granting-users-xcat-privileges> for required xcat/sudo setup.

**-P|--scripts**

Specifies that postscripts and postbootscripts should be run on the nodes. File sync with **updatenode -P syncfiles** is not supported. The **syncfiles** postscript can only be run during install. You should use **updatenode -F** instead.

**-S|--sw**

Specifies that node software should be updated. In Sysclone environment, specifies pushing the delta changes to target nodes.

**-n|--noverify**

Specifies that node network availability verification will be skipped.

**-s|--sn**

Set the server information stored on the nodes in /opt/xcat/xcatinfo on Linux.

**-t *timeout***

Specifies a timeout in seconds the command will wait for the remote targets to complete. If timeout is not specified it will wait indefinitely. updatenode -k is the exception that has a timeout of 10 seconds, unless overridden by this flag.

**-v|--version**

Command Version.

**-V|--verbose**

Verbose mode.

**RETURN VALUE**

0 The command completed successfully.

1 An error has occurred.



## EXAMPLES

1. To perform all updatenode features for the Linux nodes in the group “compute”:

```
updatenode compute
```

The command will: run any scripts listed in the nodes “postscripts and postbootscripts” attribute, install or update any software indicated in the /install/custom/install/<ostype>/profile.otherpkgs.pkglist (refer to the **To install or update software part**), synchronize any files indicated by the synclist files specified in the osimage “synclists” attribute.

2. To run postscripts,postbootscripts and file synchronization only on the node “clstrn01”:

```
updatenode clstrn01 -F -P
```

3. Running **updatenode -P** with the **syncfiles** postscript is not supported. You should use **updatenode -F** instead.

Do not run:

```
updatenode clstrn01 -P syncfiles
```

Run:

```
updatenode clstrn01 -F
```

4. To run the postscripts and postbootscripts indicated in the postscripts and postbootscripts attributes on the node “clstrn01”:

```
updatenode clstrn01 -P
```

5. To run the postscripts script1 and script2 on the node “clstrn01”:

```
cp script1,script2 /install/postscripts
updatenode clstrn01 -P "script1 p1 p2,script2"
```

Since flag ‘-P’ can be omitted when only script names are specified, the following command is equivalent:

```
updatenode clstrn01 "script1 p1 p2,script2"
```

p1 p2 are parameters for script1.

6. To synchronize the files on the node “clstrn01”: Prepare the synclist file. For AIX, set the full path of synclist in the osimage table synclists attribute. For Linux, put the synclist file into the location: /install/custom/<inst\_type>/<distro>/<profile>.<os>.<arch>.synclist Then:

```
updatenode clstrn01 -F
```

7. To perform the software update on the Linux node “clstrn01”: Copy the extra rpm into the /install/post/otherpkgs/<os>/<arch>/ and add the rpm names into the /install/custom/install/<ostype>/profile.otherpkgs.pkglist . Then:

```
updatenode clstrn01 -S
```

8. To update the AIX node named “xcatn11” using the “installp\_bundle” and/or “otherpkgs” attribute values stored in the xCAT database. Use the default installp, rpm and emgr flags.

```
updatenode xcatn11 -V -S
```

Note: The xCAT “xcatn11” node definition points to an xCAT osimage definition which contains the “installp\_bundle” and “otherpkgs” attributes as well as the name of the NIM lpp\_source resource.

9. To update the AIX node “xcatn11” by installing the “bos.cpr” fileset using the “-agQXY” installp flags. Also display the output of the installp command.

```
updatenode xcatn11 -V -S otherpkgs="I:bos.cpr" installp_flags="-agQXY"
```

Note: The ‘I:’ prefix is optional but recommended for installp packages.

10. To uninstall the “bos.cpr” fileset that was installed in the previous example.

```
updatenode xcatn11 -V -S otherpkgs="I:bos.cpr" installp_flags="-u"
```

11. To update the AIX nodes “xcatn11” and “xcatn12” with the “gpfs.base” fileset and the “rsync” rpm using the installp flags “-agQXY” and the rpm flags “-i --nodeps”.

```
updatenode xcatn11,xcatn12 -V -S otherpkgs="I:gpfs.base,R:rsync-2.6.2-1.aix5.1.  
→ppc.rpm" installp_flags="-agQXY" rpm_flags="-i --nodeps"
```

Note: Using the “-V” flag with multiple nodes may result in a large amount of output.

12. To uninstall the rsync rpm that was installed in the previous example.

```
updatenode xcatn11 -V -S otherpkgs="R:rsync-2.6.2-1" rpm_flags="-e"
```

13. Update the AIX node “node01” using the software specified in the NIM “sslbnd” and “sshbnd” installp\_bundle resources and the “-agQXY” installp flags.

```
updatenode node01 -V -S installp_bundle="sslbnd,sshbnd" installp_flags="-agQXY"
```

14. To get a preview of what would happen if you tried to install the “rsct.base” fileset on AIX node “node42”. (You must use the “-V” option to get the full output from the installp command.)

```
updatenode node42 -V -S otherpkgs="I:rsct.base" installp_flags="-apXY"
```

15. To check what rpm packages are installed on the AIX node “node09”. (You must use the “-c” flag so updatenode does not get a list of packages from the database.)

```
updatenode node09 -V -c -S rpm_flags="-qa"
```

16. To install all software updates contained in the /images directory.

```
updatenode node27 -V -S -A -d /images
```

Note: Make sure the directory is exportable and that the permissions are set correctly for all the files. (Including the .toc file in the case of installp filesets.)

17. Install the interim fix package located in the /efixes directory.

```
updatenode node29 -V -S -d /efixes otherpkgs=E:IZ38930TL0.120304.epkg.Z
```

18. To uninstall the interim fix that was installed in the previous example.

```
updatenode xcatsn11 -V -S -c emgr_flags="-r -L IZ38930TL0"
```

19. To update the security keys for the node “node01”

```
updatenode node01 -k
```

20. To update the service nodes with the files to be synchronized to node group compute:

```
updatenode compute -f
```

21. To run updatenode with the non-root userid “user1” that has been setup as an xCAT userid with sudo on node1 to run as root, do the following: See [Granting\\_Users\\_xCAT\\_privileges](#) for required sudo setup.

```
updatenode node1 -l user1 -P syslog
```

22. In Sysclone environment, after capturing the delta changes from golden client to management node, to run updatenode to push these delta changes to target nodes.

```
updatenode target-node -S
```

## FILES

/opt/xcat/bin/updatenode

## wcons.1

### Name

wcons - windowed remote console

### Synopsis

**wcons** [-t | --tile=*n*] [*xterm-options*] *noderange*

**wcons** [-h | --help | -v | --version]

### Description

**wcons** provides access to the remote node serial console of a single or range or nodes or groups.

**wcons** is a simple front-end to rcons in an xterm session for each console.

## Options

### **-t | --tile=*n***

Tile **wcons** windows from top left to bottom right. If *n* is specified then tile *n* across. If *n* is not specified then tile to edge of screen. If tiled **wcons** windows reach bottom right, then the windows start at top left overlaying existing **wcons** windows.

### **-h | --help**

Print help.

### **-v | --version**

Print version.

### *xterm options*

See `xterm(1)`. Any options other than those listed above are passed directly to `xterm`. **Note:** when given multiple nodes, `wcons` will override **-title** and tries to figure out optimal **-geometry** options for the `xterms` (however, **-geometry** can still be specified).

## Files

**nodehm** table - xCAT node hardware management table. See `nodehm(5)`|`nodehm.5` for further details. This is used to determine the console access method.

## Examples

**wcons** *node1-node5*

**wcons --tile --font=nl2** *all*

**wcons -t 4** *node1-node16*

**wcons -f** *vs* **-t 4** *node1-node4*

## Bugs

Tile mode assumes that the width of the left window border is also the width of the right and bottom window border. Most window managers should not have a problem. If you really need support for a screwy window manager let me know.

## See Also

`noderange(3)`|`noderange.3`, `rcons(1)`|`rcons.1`, `xterm(1)`

## wkill.1

### Name

**wkill** - kill windowed remote consoles

### Synopsis

**wkill** [*noderange*]

**wkill** [-h | --help | -v | --version]

### Description

**wkill** will kill the wcons windows on your \$DISPLAY for a single or range or nodes or groups.

**wkill** was written because I'm too lazy to point and click off 64 windows.

**wkill** will only kill windows on your display and for only the noderange(3)|noderange.3 you specify. If no noderange(3)|noderange.3 is specified, then all wcons windows on your \$DISPLAY will be killed.

### Options

**-h | --help**

Print help.

**-v | --version**

Print version.

### Examples

```
wkill node1-node5
```

### See Also

noderange(3)|noderange.3, wcons(1)|wcons.1

## wvid.1

### Name

**wvid** - windowed remote video console for nodes

## Synopsis

**wvid** *noderange*

## Description

**wvid** provides access to the remote node video console of a single node, or range of nodes or groups. **wvid** provides a simple front-end to the hardware's remote console capability. Currently this command is supported for: blades, BMC/IMM, KVM, and Xen

The **nodehm.cons** attribute of the node determines the method used to open the console. See `nodehm(5)|nodehm.5` for further details.

## Options

No options are supported at this time.

## Examples

1. To open video consoles for the 1st 2 nodes:

```
wvid node1,node2
```

## See Also

`noderange(3)|noderange.3`, `rcons(1)|rcons.1`, `wcons(1)|wcons.1`

## xCATWorld.1

### NAME

**xCATWorld** - Sample client program for xCAT.

### SYNOPSIS

**xCATWorld** *noderange*

### DESCRIPTION

The `xCATWorld` program gives you a sample client program that interfaces to the `/opt/xcat/lib/perl/xCAT_plugin/xCATWorld.pm` plugin. For debugging purposes we have an Environment Variable `XCATBYPASS`. If `export XCATBYPASS=yes`, the client will call the plugin without going through the `xcat` daemon, `xcatd`.

## OPTIONS

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1.To run , enter:

```
xCATWorld nodegrp1
```

## FILES

/opt/xcat/bin/xCATWorld

## NOTES

This command is part of the xCAT software product.

### xcat2nim.1

## NAME

**xcat2nim** - Use this command to create and manage AIX NIM definitions based on xCAT node, group and network object definitions.

## SYNOPSIS

**xcat2nim** [-h|--help]

**xcat2nim** [-V|--verbose] [-u|--update] [-l|--list] [-r|--remove] [-f|--force] [-t *object-types*] [-o *object-names*] [-a|--allobjects] [-p|--primarySN] [-b|--backupSN] [*noderange*] [*attr=val* [*attr=val...*]]

## DESCRIPTION

The **xcat2nim** command uses xCAT node, group and network object definitions to create, update, list, or remove corresponding NIM definitions.

Before you create or update NIM definitions the xCAT definitions must be created and NIM must be configured.

The **xcat2nim** command uses xCAT database information, command line input, and default values to run the appropriate NIM commands.

The xCAT node, group and network definition names will correspond to the NIM machine, machine group and network definitions.

Note: The length of a NIM object name must be no longer than 39 characters.

To create or update a NIM definition you must provide the names of the xCAT definitions to use. The default behavior is to create new NIM definitions but not apply updates to existing definitions. If you wish to update existing NIM

definitions then you must use the “update” option. If you wish to completely remove the old definition and re-create it you must use the “force” option.

The xCAT code uses the appropriate NIM commands to create the NIM definitions. To create definitions the “nim -o define” operation is used. To update definitions the “nim -o change” operation is used. If you wish to specify additional information to pass to the NIM commands you can use the “attr=val” support. The attribute names must correspond to the attributes supported by the relevant NIM commands. (For example. “netboot\_kernel=mp”)

If the object type you are creating is a node then the object names can be a nodename value.

If you are using xCAT service nodes the **xcatsnim** command will automatically determine the correct server for the node and create the NIM definitions on that server.

The **xcatsnim** command support for NIM networks is limited to creating and listing.

When creating network definitions the command will check to make sure the network definition (or it’s equivalent) does not exist and then create the required NIM network, route and interface definitions. In some cases the equivalent network definition may exist using a different name. In this case a new definition WILL NOT be created.

To list the NIM definitions that were created you must specify the “list” option and the names of the xCAT objects that were used to create the NIM definitions. The **xcatsnim** command will list the corresponding NIM machine, machine group or network definitions using the “lsnim -l” command.

To remove NIM definitions you must specify the “remove” option and the names of the xCAT objects that were used to create the NIM definitions.

The remove(“-r”), force(“-f”) and update(“-u”) options are not supported for NIM network definitions.

## OPTIONS

**-a|--all** The list of objects will include all xCAT node, group and network objects.

*attr=val [attr=val ...]* Specifies one or more “attribute equals value” pairs, separated by spaces. Attr=val pairs must be specified last on the command line. The attribute names must correspond to the attributes supported by the relevant NIM commands. When providing attr=val pairs on the command line you must not specify more than one object type.

**-b|--backupSN** When using backup service nodes only update the backup. The default is to update both the primary and backup service nodes.

**-f|--force** The force option will remove the existing NIM definition and create a new one.

**-h|--help** Display the usage message.

**-l|--list** List NIM definitions corresponding to xCAT definitions.

**-o object-names** A set of comma delimited xCAT object names. Objects must be of type node, group, or network.

**-p|--primarySN** When using backup service nodes only update the primary. The default is to update both the primary and backup service nodes.

**-r|--remove** Remove NIM definitions corresponding to xCAT definitions.

**-t object-types** A set of comma delimited xCAT object types. Supported types include: node, group, and network.

Note: If the object type is “group”, it means that the **xcatsnim** command will operate on a NIM machine group definition corresponding to the xCAT node group definition. Before creating a NIM machine group, all the NIM client nodes definition must have been created.

**-u|--update** Update existing NIM definitions based on xCAT definitions.

**-V|--verbose** Verbose mode.



## RETURN VALUE

- 0 The command completed successfully.
- 1 An error has occurred.

## EXAMPLES

1. To create a NIM machine definition corresponding to the xCAT node “clstrn01”.

```
xcat2nim -t node -o clstrn01
```

2. To create NIM machine definitions for all xCAT node definitions.

```
xcat2nim -t node
```

3. Update all the NIM machine definitions for the nodes contained in the xCAT “compute” node group and specify attribute values that will be applied to each definition.

```
xcat2nim -u -t node -o compute netboot_kernel=mp cable_type="N/A"
```

4. To create a NIM machine group definition corresponding to the xCAT group “compute”.

```
xcat2nim -t group -o compute
```

5. To create NIM network definitions corresponding to the xCAT “clstr\_net” an “publc\_net” network definitions. Also display verbose output.

```
xcat2nim -V -t network -o "clstr_net,publc_net"
```

6. To list the NIM definition for node clstrn02.

```
xcat2nim -l -t node clstrn02
```

7. To re-create a NIM machine definition and display verbose output.

```
xcat2nim -V -t node -f clstrn05
```

8. To remove the NIM definition for the group “AIXnodes”.

```
xcat2nim -t group -r -o AIXnodes
```

9. To list the NIM “clstr\_net” definition.

```
xcat2nim -l -t network -o clstr_net
```

## FILES

\$XCATROOT/bin/xcat2nim

## NOTES

This command is part of the xCAT software product.

## SEE ALSO

mkdef(1)|mkdef.1

## xcatchroot.1

## NAME

**xcatchroot** - Use this xCAT command to modify an xCAT AIX diskless operating system image.

## SYNOPSIS

**xcatchroot -h**

**xcatchroot [-V] -i *osimage\_name cmd\_string***

## DESCRIPTION

For AIX diskless images this command will modify the AIX SPOT resource using the chroot command. You must include the name of an xCAT osimage definition and the command that you wish to have run in the spot.

### WARNING:

Be very careful when using this command!!! Make sure you are very clear about exactly what you are changing so that you do not accidentally corrupt the image.

As a precaution it is advisable to make a copy of the original spot in case your changes wind up corrupting the image.

When you are done updating a NIM spot resource you should always run the NIM check operation on the spot.

```
nim -Fo check <spot_name>
```

The xcatchroot command will take care of any of the required setup so that the command you provide will be able to run in the spot chroot environment. It will also mount the lpp\_source resource listed in the osimage definition so that you can access additional software that you may wish to install.

For example, assume that the location of the spot named in the xCAT osimage definition is /install/nim/spot/614spot/usr. The associated root directory in this spot would be /install/nim/spot/614spot/usr/lpp/bos/inst\_root. The chroot is automatically done to this new root directory. The spot location is mounted on /../inst\_root/usr so that when your command is run in the chroot environment it is actually running commands from the spot usr location.

Also, the location of the lpp\_source resource specified in the osimage definition will be mounted to a subdirectory of the spot /../inst\_root directory. For example, if the lpp\_source location is /install/nim/lpp\_source/614lpp\_lpp\_source then that would be mounted over /install/nim/spot/614spot/usr/lpp/bos/inst\_root/lpp\_source.

When you provide a command string to run make sure you give the full paths of all commands and files assuming the `../../inst_root` directory is your root directory.

If you wish to install software from the `lpp_source` location you would provide a directory location of `/lpp_source` (or `/lpp_source/install/ppc` or `/lpp_source/RPMS/ppc` etc.) See the example below.

Always run the NIM check operation after you are done updating your spot. (ex. “nim -o check <spot\_name>”)

## OPTIONS

*cmd\_string*

The command you wish to have run in the chroot environment. (Use a quoted string.)

**-h |--help**

Display usage message.

**-i osimage\_name**

The name of the xCAT osimage definition.

**-V |--verbose**

Verbose mode.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1) Set the root password to “cluster” in the spot so that when the diskless node boots it will have a root password set.

```
xcatchroot -i 614spot "/usr/bin/echo root:cluster | /usr/bin/chpasswd -c"
```

2) Install the bash rpm package.

```
xcatchroot -i 614spot "/usr/bin/rpm -Uvh /lpp_source/RPMS/ppc bash-3.2-1.aix5.2.ppc.rpm"
```

3) To enable system debug.

```
xcatchroot -i 614spot "bosdebug -D -M"
```

4) To set the “ipforwarding” system tunable.

```
xcatchroot -i 614spot "/usr/sbin/no -r -o ipforwarding=1"
```

## FILES

/opt/xcat/bin/xcatchroot

## NOTES

This command is part of the xCAT software product.

### **xcatperftest.1**

## NAME

**xcatperftest** - Run xCAT command performance baseline testing on fake nodes.

## SYNOPSIS

**xcatperftest** [-?|-h]

[PERF\_DRYRUN=y] **xcatperftest** run [*command-list-file*]

[PERF\_DRYRUN=y] [PERF\_NOCREATE=y] **xcatperftest** <total> [*command-list-file*]

## DESCRIPTION

The **xcatperftest** command runs commands defined in a command list file and get their execution response time baseline for performance purpose. The **xcatperftest** command is part of the xCAT package **xCAT-test**, and you can run it standalone or leverage it to build up your automation test cases.

Any command could be defined in the command list file, however, it is recommended that the one-time initial configuration is well prepared prior to running **xcatperftest** command. For example, the network object, osdistro and osimage image objects.

Follow the steps below to run **xcatperftest** command:

1. Install **xCAT-test** on a xCAT management node.
2. Prepare a command list with the commands you want to measure.
3. Prepare the initial configuration based on the command list to make sure all commands could be executed in technical.
4. Run **xcatperftest** with the total fake nodes number and the above command list file.

Node: It is suggested to run the command in background as it normally takes long time to finish all the performance testing with large amount of fake nodes.

## OPTIONS

**-?|-h**

Display usage message.

*command-list-file*

Specifies the command list file with full-path. xCAT supports an example command file:  
/opt/xcats/share/xcats/tools/autotest/perfcmds.lst

**<total>**

Total number of fake nodes will be defined during the testing.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## COMMAND LIST FILE

The command list file is in flat text format, the testing framework will parse the file line by line, here is an example of the command list file:

```
#SERIES# 1,50,100,250,500,1000,2500,5000
mkdef -z -f < #STANZ#
lsdef #NODES#
makehosts #NODES#
makedns -n #NODES#
makedhcp #NODES#
makeknownhosts #NODES#
nodech #NODES# groups,=group1
nodeis #NODES# noderes
nodeset #NODES# osimage=rhels7.3-GA-ppc64le-install-compute
chdef -t node -o #NODES# postscripts="fake" profile="install" netboot="grub2"
rmdef -t node #PERFGRP#
mkdef -z < #STANZ#
noderm #PERFGRP#
```

**Note:** Each line defines one command, and the commands dependency should be handled by the line order. If you define a node range series line (started with **#SERIES#**) in this file, xcatperftest will run the command for each node range defined in series line.

**#SERIES#** To define a node range series, and the series should be an comma split incremental number sequence.

**#STANZ#** It will be replaced with real stanz file path when this command line runs.

**#NODES#** It will be replaced with real node range defined in **#SERIES#** line when this command line runs. If no series line, the node group will be used.

**#PERFGRP#** It will be replaced with node group when this command line runs.

## ENVIRONMENT VARIABLE

The **xcatperftest** command supports customization by some environment variables.

### **FAKE\_NODE\_PREFIX**

Optional, the prefix of the fake compute node name. By default, the value is 'fake'

### **FAKE\_NODE\_GROUP**

# Optional, the group name of all the fake compute nodes. By default, the value is 'perftest'

### **FAKE\_NETWORK\_PRO**

Mandatory, the Provision network for all the fake compute nodes. By default, the value is '192.168'. It must be a string like 'A.B', and be matched with `tabdump networks`

### **FAKE\_NETWORK\_BMC**

Mandatory, the BMC network for all the fake compute nodes. By default, the value is '192.168'. Note: It could not be the same subnet as 'FAKE\_NETWORK\_PRO' It must be a string like 'A.B' and no need to be defined in 'networks' table.

### **PERF\_NODETEMPL**

Optional, the node template name used for generating fake nodes. By default, it will be auto-detected according to the current arch.

### **PERF\_DRYRUN**

Optional, indicate no real commands will be executed if the environment variable is set.

### **PERF\_NOCREATE**

Optional, indicate no new fake nodes will be created if the environment variable is set.

## EXAMPLES

1. To run the performance testing for the commands defined in /tmp/cmd.lst on 5000 fake nodes:

```
xcatperftest 5000 /tmp/cmd.lst
```

2. To generate an xCAT node object stanz file for 10000 nodes in subnet 10.100.0.0:

```
FAKE_NETWORK_PRO=10.100 FAKE_NETWORK_BMC=10.200 xcatperftest 10000
```

3. To run the performance testing for the commands defined in /opt/xcat/share/xcat/tools/autotest/perfcmds.lst on 5000 existing fake nodes:

```
PERF_NOCREATE=y xcatperftest 5000 /opt/xcat/share/xcat/tools/autotest/perfcmds.lst
```

4. To run the performance testing for the commands defined in /tmp/cmd.lst in existing xCAT environment:

```
xcatperftest run /tmp/cmd.lst
```

## FILES

/opt/xcat/bin/xcatperftest

/opt/xcat/share/xcat/tools/autotest/perfcmds.lst

## xcattest.1

## NAME

**xcattest** - Run automated xCAT test cases.

## SYNOPSIS

**xcattest** [-?|-h]

**xcattest** [-f *configure file*[:System]] [-l [{caselist|caseinfo|casenum}]] [-r] [-q] [-b *testcase bundle list*]

**xcattest** [-f *configure file*[:System]] [-l [{caselist|caseinfo|casenum}]] [-r] [-q] [-t *testcase name list*]

**xcattest** [-f *configure file*[:System]] [-l [{caselist|caseinfo|casenum}]] [-r] [-q] [-c *testcase command list*]

**xcattest** [-f *configure file*[:System]] [-l [{caselist|caseinfo|casenum}]] [-r] [-q] [-s *testcase filter expression*]

**xcattest** [-f *configure file*[:System]] -l *bundleinfo*

## DESCRIPTION

The **xcattest** command runs test cases to verify the xCAT functions. It can be used to ensure the code changes you made do not break the existing commands; to run acceptance test for new build you got; to verify the xCAT snapshot build or development build before putting it onto your production system. The **xcattest** command is part of the xCAT package *xCAT-test*.

The root directory for the *xCAT-test* package is */opt/xcat/share/xcat/tools/autotest/*. All test cases are in the sub directory *testcase*, indexed by the xCAT command, you can add your own test cases according to the test cases format below. The subdirectory *bundle* contains all the test cases bundle definition files, you can customize or create any test cases bundle file as required. The testing result information will be written into the subdirectory *result*, the timestamps are used as the postfixes for all the result files. *xCAT-test* package ships two configuration file templates: *aix.conf.template* and *linux.conf.template* for AIX and Linux environment, you can use the template files as the starting point of making your own configuration file.

## OPTIONS

-?|-h

Display usage message.

-f *configure file*

Specifies the configuration file with full-path. If not specified, an example config file: */opt/xcat/share/xcat/tools/autotest/linux.conf.template* is used by default. If **System** tag is used, only [System] section in the configuration file will be used. If **System** is not used, all other sections of the configuration file will be used, like [Table], [Object], etc.

-b *testcase bundle list*

Comma separated list of test case bundle files, each test cases bundle can contain multiple lines and each line for one test case name. The bundle files should be placed in `/opt/xcat/share/xcat/tools/autotest/bundle`.

**-t** *testcase name list*

Comma separated list of test cases to run.

**-c** *testcase command list*

Comma separated list of commands which will be tested, i.e., all the test cases under the command sub directory will be run.

**-s** *filter expression*

Run testcases with testcase **label** attribute matching *filter expression*. Operators `|`, `+`, and `-` can be used. Expresson `"label1+label2-label3|label4|label5"` will match testcases that have **label** attribute matching "label1" and "label2", but not "label3" or testcases that have **label** attribute matching "label4" or testcases that have **label** attribute matching "label5"

**-l** {**caselist**|**caseinfo**|**casenum**|**bundleinfo**}

Display rather than run the test cases. The **caselist** is a default and will display a list of testcase names. **caseinfo** will display testcase names and descriptions. **casenum** will display the number of testcases. **bundleinfo** will display testcase bundle names and descriptions.

**-r**

Back up the original environment settings before running test, and restore them after running test.

**-q**

Do not print output of test cases to STDOUT, instead, log output to `/opt/xcat/share/xcat/tools/autotest/result`.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## TEST CASE FORMAT

The xCAT-test test cases are in flat text format, the testing framework will parse the test cases line by line, here is an example of the test case:

```
#required, case name
start:case name
#optional, description of the test case
description: what the test case is for?
#optional, environment requirements
os:AIX/Linux/sles/ubuntu/rhels/rhels7/rhels8
#optional, environment requirements
arch:ppc/ppc64/ppc64le/x86_64
#optional, environment requirements
hcp:hmc/mm/bmc/fsp/ipmi/openbmc
#optional, label
label:label1
#required, command need to run
```

(continues on next page)



(continued from previous page)

```
cmd:command
#optional, check return code of last executed command
check:rc == or != return code
#optional, check output of last executed command
check:output== or != or =~ or !~ output check string
end
```

**Note:** Each test case can have more than one *cmd* sections and each *cmd* section can have more than one *check:rc* sections and more than one *check:output* sections, the *output check string* can include regular expressions.

## EXAMPLES

1. To run all test cases related to command **rpower**:

```
xcattest -f /tmp/config -c rpower
```

2. To run customized bundle with */tmp/config* file:

```
xcattest -c lsdef -l > /opt/xcat/share/xcat/tools/autotest/bundle/custom.bundle
Modify custom.bundle
xcattest -f /tmp/config -b custom.bundle
```

3. To run specified test cases with */tmp/config* file:

```
xcattest -f /tmp/config -t lsdef_t_o_l_z
```

4. To add a new test case to test **chvm**. In this example, we assume that the **min\_mem** should not be equal to 16 in the lpar profile of computenode. The case name is **chvm\_custom**. It creates a test lpar named **testnode** first, then changes the **min\_mem** of the lpar to 16 using **chvm**, then checks if **min\_mem** have changed correctly. Finally, the **testnode** is removed.

```
add a new test case file in /opt/xcat/share/xcat/tools/autotest/chvm
edit filename
start:chvm_custom
hcp:hmc
cmd:lsvm $$CN > /tmp/autotest.profile
check:rc==0
cmd:mkdef -t node -o testnode mgt=hmc groups=all
cmd:mkvm testnode -i $$MaxLparID -l $$CN
check:rc==0
cmd:perl -pi -e 's/min_mem=\d+/min_mem=16/g' /tmp/autotest.profile
cmd:cat /tmp/autotest.profile|chvm testnode
check:rc==0
cmd:lsvm testnode
check:output=~min_mem=16
cmd:rmvm testnode
cmd:rm -f /tmp/autotest.profile
end
```

5. To run all test cases that have *label:kdump* or *label:parallel\_cmds*:

```
xcattest -s kdump|parallel_cmds
```

6. To display all bundles and their descriptions:

```
xcattest -l bundleinfo
```

## INLINE FUNCTIONS

The xCAT-test testing framework provides some inline functions. The inline functions can be called in test cases as `__FUNCTIONNAME(PARAMETERLIST)__` to get some necessary attributes defined in the configuration file. The inline functions can be used in *cmd* section and the *check:output* section.

1. **GETNODEATTR(nodename, attribute)** To get the value of specified node's attribute
2. **INC(digit)** To get value of digit+1.

For example, to run **rscan** command against the hardware control point of compute node specified in the configuration file:

```
rscan __GETNODEATTR($$CN, hcp)__ -z
```

3. **GETTABLEVALUE(keyname, key, colname, table)** To get the value of column where keyname == key in specified table.

## FILES

/opt/xcat/bin/xcattest

## xcoll.1

## NAME

**xcoll** - Formats and consolidates the output of the **psh**, **rinv** commands.

## SYNOPSIS

**xcoll** [-n] [-c]

## DESCRIPTION

The **xcoll** command formats and consolidates output from the **psh**, **rinv** command. The **xcoll** command takes, as input, lines in the following format:

groupname: line of output from remote command, will use group name, if defined

The **xcoll** command formats the lines as follows and writes them to standard output. Assume that the output from node3 and node4 is identical:

```
=====
node1 or nodegroup name
=====
.
```

(continues on next page)

(continued from previous page)

```

.
lines from psh for node1 with hostnames stripped off
.
.

=====
node2 or nodegroup name
=====
.
.
lines from psh for node2 with hostnames stripped off
.
.

=====
node3, node4 or nodegroup name
=====
.
.
lines from psh for node 3 with hostnames stripped off
.
.

```

## OPTIONS

**-c**

Display a total nodecount for each set of output.

**-n**

Display output as nodenames instead of groupnames.

## EXAMPLES

1. To display the results of a command issued on several nodes, in the format used in the Description, enter:

```
psh node1,node2,node3 cat /etc/passwd | xcoll
```

## SEE ALSO

psh(1)|psh.1, xdshbak(1)|xdshbak.1 ,xdshcoll(1)|xdshcoll.1

## xdcp.1

### NAME

**xdcp** - Concurrently copies files to or from multiple nodes. In addition, provides an option to use **rsync** to update the files on the managed nodes, or to an installation image on the local node.

### SYNOPSIS

**xdcp** *noderange* [[**-B** | **--bypass**] [**-f** *fanout*] [**-L**] [**-l** *user\_ID*] [**-o** *node\_options*] [**-p**] [**-P**] [**-r** *node remote copy command*] [**-R**] [**-t** *timeout*] [**-T**] [**-v**] [**-q**] [**-X** *env\_list*] *sourcefile...* *targetpath*

**xdcp** *noderange* [**-F** *rsynclist input file*] [**-r** *node remote copy command*]

**xdcp** *computenoderange* [**-s** **-F** *synclist input file*] [**-r** *node remote copy command*]

**xdcp** [**-i** *install image*] [**-F** *synclist input file*] [**-r** *node remote copy command*]

**xdcp** [**-h** | **-V** | **-q**]

### DESCRIPTION

The **xdcp** command concurrently copies files to or from remote target nodes. The command issues a remote copy command for each node or device specified. When files are pulled from a target, they are placed into the *targetpath* with the name of the remote node or device appended to the copied *sourcefile* name. The **/usr/bin/rcp** command is the model for syntax and security. If using hierarchy, then **xdcp** runs on the service node that is servicing the compute node. The file will first be copied to the path defined in the site table, **SNsyncfiledir** attribute, or the default path **/var/xcat/syncfiles** on the service node, if the attribute is not defined. The **-P** flag will not automatically copy the files from the compute node to the Management node, hierarchically. There is a two step process, see **-P** flag. If the Management Node is target node, it must be defined in the xCAT database with **nodetype=mn**. When the **xdcp** command runs with the Management Node as the target, it does not use remote commands but uses the local OS copy (**cp**) command.

#### REMOTE USER:

A *user\_ID* can be specified for the remote copy command. Remote user specification is identical for the **xdcp** and **xdsh** commands. See the **xdsh** command for more information.

#### REMOTE COMMAND COPY:

The **xdcp** command uses a configurable remote copy command to execute remote copies on remote targets. Support is explicitly provided for Remote Shell **rcp** command, the OpenSSH **scp** command and the **/usr/bin/rsync** command.

For node targets, the remote copy command is determined by the following order of precedence:

1. The **-r** flag.
2. The **/usr/bin/rsync** command.

#### COMMAND EXECUTIONS:

The maximum number of concurrent remote copy command processes (the fanout) can be specified with the **-f** flag or the **DSH\_FANOUT** environment variable. The fanout is only restricted by the number of remote shell commands that can be run in parallel. You can experiment with the **DSH\_FANOUT** value on your management server to see if higher values are appropriate.

A timeout value for remote copy command execution can be specified with the **-t** flag or DSH\_TIMEOUT environment variable. If any remote target does not respond within the timeout value, the **xdep** command displays an error message and exits.

The **-T** flag provides diagnostic trace information for **xdep** command execution. Default settings and the actual remote copy commands that are executed to the remote targets are displayed.

The **xdep** command can be executed silently using the **-Q** flag; no target standard output or standard error is displayed.

## OPTIONS

*sourcefile...*

Specifies the complete path for the file to be copied to or from the target. Multiple files can be specified. When used with the **-R** flag, only a single directory can be specified. When used with the **-P** flag, only a single file can be specified.

*targetpath*

If one source file, then it specifies the file to copy the source file to on the target. If multiple source files, it specifies the directory to copy the source files to on the target. If the **-P** flag is specified, the *targetpath* is the local host location for the copied files. The remote file directory structure is recreated under *targetpath* and the remote target name is appended to the copied *sourcefile* name in the *targetpath* directory. Note: the *targetpath* directory must exist.

**-B | --bypass**

Runs in bypass mode, use if the **xcatd** daemon is not responding.

**-f | --fanout** *fanout\_value*

Specifies a fanout value for the maximum number of concurrently executing remote shell processes. Serial execution can be specified by indicating a fanout value of **1**. If **-f** is not specified, a default fanout value of **64** is used.

**-F | --File** *synclist input file*

Specifies the path to the file that will be used to build the **rsync** command. The format of the input file is described here: <[https://xcat-docs.readthedocs.io/en/stable/guides/admin-guides/manage\\_clusters/common/deployment/syncfile/syncfile\\_synclist\\_file.html](https://xcat-docs.readthedocs.io/en/stable/guides/admin-guides/manage_clusters/common/deployment/syncfile/syncfile_synclist_file.html)>

On Linux **rsync** always uses ssh remoteshell. On AIX, **ssh** or **rsh** is used depending on the **site.useSSHonAIX** table attribute.

**-h | --help**

Displays usage information.

**-i | --rootimg** *install image*

Specifies the path to the install image on the local Linux node.

**-o | --node-options** *node\_options*

Specifies options to pass to the remote shell command for node targets. The options must be specified within double quotation marks (") to distinguish them from **xdep** options.

**-p | --preserve**

Preserves the source file characteristics as implemented by the configured remote copy command.

**-P | --pull**

Pulls (copies) the files from the targets and places them in the *targetpath* directory on the local host. The *targetpath* must be a directory. Files pulled from remote machines have **.\_target** appended to the file name to distinguish between them. When the **-P** flag is used with the **-R** flag, **.\_target** is appended to the directory. Only one file per invocation of the **xdcp** pull command can be pulled from the specified targets. In hierarchy, you must first pull the file to the service node and then pull the file to the management node.

**-q | --show-config**

Displays the current environment settings for all DSH Utilities commands. This includes the values of all environment variables and settings for all currently installed and valid contexts. Each setting is prefixed with *context*: to identify the source context of the setting.

**-r | --node-rcp** *node remote copy command*

Specifies the full path of the remote copy command used for syncing files to node targets, such as **/usr/bin/rsync** or **/usr/bin/scp**. If not specified, **rsync** will be used by default.

Note: The synclist processing for **-r /usr/bin/scp** has some differences with **-r /usr/bin/rsync**:

- 1) the **EXECUTE** clause in synclist file is not supported with **-r /usr/bin/scp** flag
- 2) if the destination directory specified in synclist file is an existing file on target node, **xdcp -r /usr/bin/scp** will fail with “scp: <destination directory>: Not a directory”
- 3) if the destination file specified in synclist file is an existing directory on target node, **xdcp -r /usr/bin/scp** will fail with “scp: <destination file>: Is a directory”

**-R | --recursive**

Recursively copies files from a local directory to the remote targets, or when specified with the **-P** flag, recursively pulls (copies) files from a remote directory to the local host. A single source directory can be specified using the *sourcefile* parameter.

**-s** *synch service nodes*

Will only sync the files listed in the synclist (**-F**), to the service nodes for the input compute node list. The files will be placed in the directory defined by the **site.SNsyncfiledir** table attribute, or the default **/var/xcatsyncfiles** directory.

**-t | --timeout** *timeout*

Specifies the time, in seconds, to wait for output from any currently executing remote targets. If no output is available from any target in the specified *timeout*, **xdsh** displays an error and terminates execution for the remote targets that failed to respond. If *timeout* is not specified, **xdsh** waits indefinitely to continue processing output from all remote targets. When specified with the **-i** flag, the user is prompted for an additional timeout interval to wait for output.

**-T | --trace**

Enables trace mode. The **xdcp** command prints diagnostic messages to standard output during execution to each target.

**-v | --verify**

Verifies each target before executing any remote commands on the target. If a target is not responding, execution of remote commands for the target is canceled.

**-V | --version**

Displays the **xdcp** command version information.

## Environment Variables

### DSH\_ENVIRONMENT

Specifies a file that contains environment variable definitions to export to the target before executing the remote command. This variable is overridden by the **-E** flag.

### DSH\_FANOUT

Specifies the fanout value. This variable is overridden by the **-f** flag.

### DSH\_NODE\_OPTS

Specifies the options to use for the remote shell command with node targets only. This variable is overridden by the **-o** flag.

### DSH\_NODE\_RCP

Specifies the full path of the remote copy command to use to copy local scripts and local environment configuration files to node targets.

### DSH\_NODE\_RSH

Specifies the full path of the remote shell to use for remote command execution on node targets. This variable is overridden by the **-r** flag.

### DSH\_NODEGROUP\_PATH

Specifies a colon-separated list of directories that contain node group files for the **DSH** context. When the **-a** flag is specified in the **DSH** context, a list of unique node names is collected from all node group files in the path.

### DSH\_PATH

Sets the command path to use on the targets. If **DSH\_PATH** is not set, the default path defined in the profile of the remote *user\_ID* is used.

### DSH\_SYNTAX

Specifies the shell syntax to use on remote targets; **ksh** or **csh**. If not specified, the **ksh** syntax is assumed. This variable is overridden by the **-S** flag.

### DSH\_TIMEOUT

Specifies the time, in seconds, to wait for output from each remote target. This variable is overridden by the **-t** flag.

## Exit Status

Exit values for each remote copy command execution are displayed in messages from the **xdcp** command, if the remote copy command exit value is non-zero. A non-zero return code from a remote copy command indicates that an error was encountered during the remote copy. If a remote copy command encounters an error, execution of the remote copy on that target is bypassed.

The **xdcp** command exit code is 0, if the **xdcp** command executed without errors and all remote copy commands finished with exit codes of 0. If internal **xdcp** errors occur or the remote copy commands do not complete successfully, the **xdcp** command exit value is greater than 0.

## Security

The **xdcp** command has no security configuration requirements. All remote command security requirements - configuration, authentication, and authorization - are imposed by the underlying remote command configured for **xdsh**. The command assumes that authentication and authorization is configured between the local host and the remote targets. Interactive password prompting is not supported; an error is displayed and execution is bypassed for a remote target if password prompting occurs, or if either authorization or authentication to the remote target fails. Security configurations as they pertain to the remote environment and remote shell command are userdefined.

## Examples

1. To copy the `/etc/hosts` file from all nodes in the cluster to the `/tmp/hosts.dir` directory on the local host, enter:

```
xdcp all -P /etc/hosts /tmp/hosts.dir
```

A suffix specifying the name of the target is appended to each file name. The contents of the `/tmp/hosts.dir` directory are similar to:

```
hosts._node1  hosts._node4  hosts._node7
hosts._node2  hosts._node5  hosts._node8
hosts._node3  hosts._node6
```

2. To copy the directory `/var/log/testlogdir` from all targets in `NodeGroup1` with a fanout of 12, and save each directory on the local host as `/var/log._target`, enter:

```
xdcp NodeGroup1 -f 12 -RP /var/log/testlogdir /var/log
```

3. To copy `/localnode/smallfile` and `/tmp/bigfile` to `B/tmp` on `node1` using `rsync` and input `-t` flag to `rsync`, enter:

```
xdcp node1 -r /usr/bin/rsync -o "-t" /localnode/smallfile /tmp/bigfile /tmp
```

4. To copy the `/etc/hosts` file from the local host to all the nodes in the cluster, enter:

```
xdcp all /etc/hosts /etc/hosts
```

5. To copy all the files in `/tmp/testdir` from the local host to all the nodes in the cluster, enter:

```
xdcp all /tmp/testdir/* /tmp/testdir
```

6. To copy all the files in `/tmp/testdir` and it's subdirectories from the local host to `node1` in the cluster, enter:

```
xdcp node1 -R /tmp/testdir /tmp/testdir
```

7. To copy the `/etc/hosts` file from `node1` and `node2` to the `/tmp/hosts.dir` directory on the local host, enter:

```
xdcp node1,node2 -P /etc/hosts /tmp/hosts.dir
```

8. To `rsync` the `/etc/hosts` file to your compute nodes:

First create a syncfile `/tmp/myrsync`, with this line:

```
/etc/hosts -> /etc/hosts
```

or



```
/etc/hosts -> /etc/ (last / is required)
```

Then run:

```
xdcp compute -F /tmp/myrsync
```

9. To rsync all the files in /home/mikev to the compute nodes:

First create a rsync file /tmp/myrsync, with this line:

```
/home/mikev/* -> /home/mikev/ (last / is required)
```

Then run:

```
xdcp compute -F /tmp/myrsync
```

10. To rsync to the compute nodes, using service nodes:

First create a rsync file /tmp/myrsync, with this line:

```
/etc/hosts /etc/passwd -> /etc
```

or

```
/etc/hosts /etc/passwd -> /etc/
```

Then run:

```
xdcp compute -F /tmp/myrsync
```

11. To rsync to the service nodes in preparation for rsyncing the compute nodes during an install from the service node.

First create a rsync file /tmp/myrsync, with this line:

```
/etc/hosts /etc/passwd -> /etc
```

Then run:

```
xdcp compute -s -F /tmp/myrsync
```

12. To rsync the /etc/file1 and file2 to your compute nodes and rename to filex and filey:

First create a rsync file /tmp/myrsync, with these line:

```
/etc/file1 -> /etc/filex
```

```
/etc/file2 -> /etc/filey
```

Then run:

```
xdcp compute -F /tmp/myrsync
```

to update the Compute Nodes

13. To rsync files in the Linux image at /install/netboot/fedora9/x86\_64/compute/rootimg on the MN:

First create a rsync file /tmp/myrsync, with this line:

```
/etc/hosts /etc/passwd -> /etc
```

Then run:

```
xdcp -i /install/netboot/fedora9/x86_64/compute/rootimg -F /tmp/myrsync
```

14. To define the Management Node in the database so you can use xdc, run

```
xcatconfig -m
```

## Files

## SEE ALSO

xdsh(1)|xdsh.1, noderange(3)|noderange.3

## xdsh.1

## NAME

**xdsh** - Concurrently runs remote commands on multiple nodes (Management Node, Service Nodes, compute nodes), or an install image.

## SYNOPSIS

**xdsh** *noderange* [-B | --bypass] [--devicetype *type\_of\_device*] [-e] [-E *environment\_file*] [-f *fanout*] [-L] [-I *userID*] [-m] [-o *node\_options*] [-Q] [-r *node\_remote\_shell*] [-s] [-S {*cs*h | *k*sh}] [-t *timeout*] [-T] [-v] [-X *env\_list*] [-z] [--sudo] *command\_list*

**xdsh** *noderange* [-K]

**xdsh** *noderange* [-K] [-I *userID*] --devicetype *type\_of\_device*

**xdsh** [-i *image path* | *nim image name*] *command\_list*

**xdsh** *noderange* [-c]

**xdsh** [-h | -V | -q]

## DESCRIPTION

The **xdsh** command runs commands in parallel on remote nodes and/or the Management Node. The **xdsh** command issues a remote shell command for each target specified, and returns the output from all targets, formatted so that command results from all nodes can be managed. If the command is to be executed on the Management Node, it does not use a remote shell command, but uses the local OS copy or shell command. The Management Node must be defined in the xCAT database. The best way to do this is to use the **xcatconfig -m** option. The **xdsh** command is an xCAT Distributed Shell Utility.

## COMMAND SPECIFICATION:

The commands to execute on the targets are specified by the *command\_list* **xdsh** parameter, or executing a local script using the **-e** flag.

The syntax for the *command\_list* **xdsh** parameter is as follows:

```
command[; command]...
```

where *command* is the command to run on the remote target. Quotation marks are required to ensure that all commands in the list are executed remotely, and that any special characters are interpreted correctly on the remote target. A script file on the local host can be executed on each of the remote targets by using the **-e** flag. If **-e** is specified, *command\_list* is the script name and arguments to the script. For example:

```
xdsh hostname -e script_filename [arguments]...
```

The *script\_filename* file is copied to a random filename in the **/tmp** directory on each remote target and then executed on the targets.

The **xdsh** command does not work with any interactive commands, including those that read from standard input.

### REMOTE SHELL COMMAND:

The **xdsh** command uses a configurable remote shell command to execute remote commands on the remote targets. Support is explicitly provided for AIX Remote Shell and OpenSSH, but any secure remote command that conforms to the IETF (Internet Engineering Task Force) Secure Remote Command Protocol can be used.

The remote shell is determined as follows, in order of precedence:

1. The **-r** flag.
2. The **DSH\_NODE\_RSH** environment variable.
3. The default node remote shell as defined by the target *context*.
4. The **/usr/bin/ssh** command.

The remote shell options are determined as follows, in order of precedence:

1. The **-o** flag.
2. The **DSH\_NODE\_OPTS** environment variable.

### REMOTE SHELL ENVIRONMENT:

The shell environment used on the remote target defaults to the shell defined for the *user\_ID* on the remote target. The command syntax that **xdsh** uses to form the remote commands can be specified using the **-S** flag. If **-S** is not specified, the syntax defaults to **sh** syntax.

When commands are executed on the remote target, the path used is determined by the **DSH\_PATH** environment variable defined in the shell of the current user. If **DSH\_PATH** is not set, the path used is the remote shell default path. For example, to set the local path for the remote targets, use:

```
DSH_PATH=$PATH
```

The **-E** flag exports a local environment definition file to each remote target. Environment variables specified in this file are defined in the remote shell environment before the *command\_list* is executed. The file should be executable and contain one environment variable per line.

### COMMAND EXECUTION:

The maximum number of concurrent remote shell command processes (the fanout) can be specified with the **-f** flag or with the **DSH\_FANOUT** environment variable. The fanout is only restricted by the number of remote shell commands that can be run in parallel. You can experiment with the **DSH\_FANOUT** value on your management server to see if higher values are appropriate.

A timeout value for remote command execution can be specified with the **-t** flag or with the **DSH\_TIMEOUT** environment variable. If any remote target does not provide output to either standard output or standard error within the timeout value, **xdsh** displays an error message and exits.

If streaming mode is specified with the **-s** flag, output is returned as it becomes available from each target, instead of waiting for the *command\_list* to complete on all targets before returning output. This can improve performance but causes the output to be unsorted.

The **-z** flag displays the exit code from the last command issued on the remote node in *command\_list*. Note that OpenSSH behaves differently; it returns the exit status of the last remote command issued as its exit status. If the command issued on the remote node is run in the background, the exit status is not displayed.

The **-m** flag monitors execution of the **xdsh** command by printing status messages to standard output. Each status message is preceded by **dsh**.

The **-T** flag provides diagnostic trace information for the execution of the **xdsh** command. Default settings and the actual remote shell commands executed on the remote targets are displayed.

No error detection or recovery mechanism is provided for remote targets. The **xdsh** command output to standard error and standard output can be analyzed to determine the appropriate course of action.

### **COMMAND OUTPUT:**

The **xdsh** command waits until complete output is available from each remote shell process and then displays that output before initiating new remote shell processes. This default behavior is overridden by the **-s** flag.

The **xdsh** command output consists of standard error and standard output from the remote commands. The **xdsh** standard output is the standard output from the remote shell command. The **xdsh** standard error is the standard error from the remote shell command. Each line is prefixed with the host name of the node that produced the output. The host name is followed by the **:** character and a command output line. A filter for displaying identical outputs grouped by node is provided separately. See the **xdshbak** command for more information.

A command can be run silently using the **-Q** flag; no output from each target's standard output or standard error is displayed.

### **SIGNALS:**

Signal 2 (INT), Signal 3 (QUIT), and Signal 15 (TERM) are propagated to the commands executing on the remote targets.

Signal 19 (CONT), Signal 17 (STOP), and Signal 18 (TSTP) default to **xdsh**; the **xdsh** command responds normally to these signals, but the signals do not have an effect on remotely executing commands. Other signals are caught by **xdsh** and have their default effects on the **xdsh** command; all current child processes, through propagation to remotely running commands, are terminated (SIGTERM).

## **OPTIONS**

### **-B | --bypass**

Runs in bypass mode, use if the xcatd daemon is hung.

### **-c | --cleanup**

This flag will have **xdsh** remove all files from the subdirectories of the the directory on the service nodes, where **xdcp** stages the copy to the compute nodes as defined in the site table *SNsyncfiledir* and *nodesyncfiledir* attribute, when the target is a service node.

It can also be used to remove the *nodesyncfiledir* directory on the compute nodes, which keeps the backup copies of files for the **xdcp** APPEND function support, if a compute node is the target.

### **-e | --execute**

Indicates that *command\_list* specifies a local script filename and arguments to be executed on the remote targets. The script file is copied to the remote targets and then remotely executed with the given arguments.

The **DSH\_NODE\_RCP** environment variables specify the remote copy command to use to copy the script file to node targets.

**-E | --environment** *environment\_file*

Specifies that the *environment\_file* contains environment variable definitions to export to the target before executing the *command\_list*.

**--devicetype** *type\_of\_device*

Specify a user-defined device type that references the location of relevant device configuration file. The devicetype value must correspond to a valid device configuration file. xCAT ships some default configuration files for Ethernet switches and IB switches under */opt/xcat/share/xcat/devicetype* directory. If you want to overwrite any of the configuration files, copy them to */var/opt/xcat/* directory and customize. For example, *base/IBSwitch/Qlogic/config* is the configuration file location if devicetype is specified as IB-Switch::Qlogic. xCAT will first search config file using */var/opt/xcat/* as the base. If not found, it will search for it using */opt/xcat/share/xcat/devicetype/* as the base.

**-f | --fanout** *fanout\_value*

Specifies a fanout value for the maximum number of concurrently executing remote shell processes. Serial execution can be specified by indicating a fanout value of **1**. If **-f** is not specified, a default fanout value of **64** is used.

**-h | --help**

Displays usage information.

**-i | --rootimg** *install image*

For Linux, Specifies the path to the install image on the local node. For AIX, specifies the name of the osimage on the local node. Run **lsnim** for valid names. **xdsh** will **chroot** (**xcatchroot** for AIX) to this path and run the **xdsh** command against the install image. No other **xdsh** flags, environment variables apply with this input. A nodename is not accepted. Only runs on the local host, normally the Management Node. The command you run must not prompt for input, the prompt will not be returned to you, and it will appear that **xdsh** hangs.

**-K | --ssh-setup**

**-K | --ssh-setup -l | --user** *user\_ID* **--devicetype** *type\_of\_device*

Set up the SSH keys for the user running the command to the specified node list. The userid must have the same uid, gid and password as the userid on the node where the keys will be setup.

If the current user is root, root's public ssh keys will be put in the *authorized\_keys\** files under roots *.ssh* directory on the node(s). If the current user is non-root, the user must be in the policy table and have credential to run the **xdsh** command. The non-root users public ssh keys and root's public ssh keys will be put in the *authorized\_keys\** files under the non-root users *.ssh* directory on the node(s). Other device types, such as IB switch, are also supported. The device should be defined as a node and nodetype should be defined as switch before connecting. The **xdsh -K** command must be run from the Management Node.

**-l | --user** *user\_ID*

Specifies a remote user name to use for remote command execution.

**-L | --no-locale**

Specifies to not export the locale definitions of the local host to the remote targets. Local host locale definitions are exported by default to each remote target.

**-m | --monitor**

Monitors remote shell execution by displaying status messages during execution on each target.

**-o | --node-options** *node\_options*

Specifies options to pass to the remote shell command for node targets. The options must be specified within double quotation marks (""") to distinguish them from **xdsh** options.

**-q | --show-config**

Displays the current environment settings for all DSH Utilities commands. This includes the values of all environment variables and settings for all currently installed and valid contexts. Each setting is prefixed with *context*: to identify the source context of the setting.

**-Q | --silent**

Specifies silent mode. No target output is written to standard output or standard error. Monitoring messages are written to standard output.

**-r | --node-rsh** *node\_remote\_shell*

Specifies the path of the remote shell command used for remote command execution on node targets.

**-s | --stream**

Specifies that output is returned as it becomes available from each target, instead of waiting for the *command\_list* to be completed on a target before returning output.

**-S | --syntax** {**csh** | **ksh**}

Specifies the shell syntax to be used on the remote target. If not specified, the **ksh** syntax is used.

**--sudo**

Adding the **--sudo** flag to the **xdsh** command will have **xdsh** run **sudo** before running the command. This is particular useful when using the **-e** option. This is required when you input **-l** with a non-root user id and want that id to be able to run as root on the node. The non-root userid will must be previously defined as an xCAT user, see process for defining non-root ids in xCAT and setting up for using **xdsh**. The userid **sudo** setup will have to be done by the admin on the node. This includes, allowing all commands that you would like to run with **xdsh** by using **visudo** to edit the */etc/sudoers* file. You must disable ssh tty requirements by commenting out or removing this line in the */etc/sudoers* file “#Defaults requiretty”. See the document <https://xcat-docs.readthedocs.io/en/stable/advanced/security/security.html#granting-users-xcat-privileges> for **sudo** setup requirements. This is not supported in a hierarchical cluster, where the nodes are serviced by service nodes.

**-t | --timeout** *timeout*

Specifies the time, in seconds, to wait for output from any currently executing remote targets. If no output is available from any target in the specified *timeout*, **xdsh** displays an error and terminates execution for the remote targets that failed to respond. If *timeout* is not specified, **xdsh** waits indefinitely to continue processing output from all remote targets. The exception is the **-K** flag which defaults to 10 seconds.

**-T | --trace**

Enables trace mode. The **xdsh** command prints diagnostic messages to standard output during execution to each target.

**-v | --verify**

Verifies each target before executing any remote commands on the target. If a target is not responding, execution of remote commands for the target is canceled. When specified with the **-i** flag, the user is prompted to retry the verification request.

**-V | --version**

Displays the **xdsh** command version information.

**-X** *env\_list*

Ignore **xdsh** environment variables. This option can take an argument which is a comma separated list of environment variable names that should **NOT** be ignored. If there is no argument to this option, or the argument is an empty string, all **xdsh** environment variables will be ignored. This option is useful when running **xdsh** from within other scripts when you don't want the user's environment affecting the behavior of **xdsh**.

#### **-z | --exit-status**

Displays the exit status for the last remotely executed non-asynchronous command on each target. If the command issued on the remote node is run in the background, the exit status is not displayed.

Exit values for each remote shell execution are displayed in messages from the **xdsh** command, if the remote shell exit values are non-zero. A non-zero return code from a remote shell indicates that an error was encountered in the remote shell. This return code is unrelated to the exit code of the remotely issued command. If a remote shell encounters an error, execution of the remote command on that target is bypassed.

The **xdsh** command exit code is **0** if the command executed without errors and all remote shell commands finished with exit codes of **0**. If internal **xdsh** errors occur or the remote shell commands do not complete successfully, the **xdsh** command exit value is greater than **0**. The exit value is increased by **1** for each successive instance of an unsuccessful remote command execution. If the remotely issued command is run in the background, the exit code of the remotely issued command is **0**.

## Environment Variables

### **DEVICETYPE**

Specify a user-defined device type. See **--devicetype** flag.

### **DSH\_ENVIRONMENT**

Specifies a file that contains environment variable definitions to export to the target before executing the remote command. This variable is overridden by the **-E** flag.

### **DSH\_FANOUT**

Specifies the fanout value. This variable is overridden by the **-f** flag.

### **DSH\_NODE\_OPTS**

Specifies the options to use for the remote shell command with node targets only. This variable is overridden by the **-o** flag.

### **DSH\_NODE\_RCP**

Specifies the full path of the remote copy command to use to copy local scripts and local environment configuration files to node targets.

### **DSH\_NODE\_RSH**

Specifies the full path of the remote shell to use for remote command execution on node targets. This variable is overridden by the **-r** flag.

### **DSH\_PATH**

Sets the command path to use on the targets. If **DSH\_PATH** is not set, the default path defined in the profile of the remote *user\_ID* is used.

### **DSH\_REMOTE\_PASSWORD**

If **DSH\_REMOTE\_PASSWORD** is set to the password of the userid (usually root) that will **ssh** to the node, then when you use the **-K** flag, you will not be prompted for a password.

## **DSH\_SYNTAX**

Specifies the shell syntax to use on remote targets; **ksh** or **csh**. If not specified, the **ksh** syntax is assumed. This variable is overridden by the **-S** flag.

## **DSH\_TIMEOUT**

Specifies the time, in seconds, to wait for output from each remote target. This variable is overridden by the **-t** flag.

## **DSH\_VERIFY**

Verifies each target before executing any remote commands on the target. If a target is not responding, execution of remote commands for the target is canceled. This variable is overridden by the **-v** flag.

## **Compatibility with AIX dsh**

To provide backward compatibility for scripts written using **dsh** in AIX and CSM, a tool has been provided **groupfiles4dsh**, which will build node group files from the xCAT database that can be used by **dsh**. See **man groupfiles4dsh**.

## **SECURITY**

The **xdsh** command has no security configuration requirements. All remote command security requirements - configuration, authentication, and authorization - are imposed by the underlying remote command configured for **xdsh**. The command assumes that authentication and authorization is configured between the local host and the remote targets. Interactive password prompting is not supported; an error is displayed and execution is bypassed for a remote target if password prompting occurs, or if either authorization or authentication to the remote target fails. Security configurations as they pertain to the remote environment and remote shell command are user defined.

## **EXIT STATUS**

The **xdsh** command exit code is 0 if the command executed without errors and all remote shell commands finished with exit codes of 0. If internal **dsh** errors occur or the remote shell commands do not complete successfully, the **dsh** command exit value is greater than 0. The exit value is increased by 1 for each successive instance of an unsuccessful remote command execution. If the remotely issued command is run in the background, the exit code of the remotely issued command is 0.

## **EXAMPLES**

1. To set up the SSH keys for root on node1, run as root:

```
xdsh node1 -K
```

2. To run the **ps -ef** command on node targets **node1** and **node2**, enter:

```
xdsh node1,node2 "ps -ef"
```

3. To run the **ps** command on node targets **node1** and run the remote command with the **-v** and **-t** flag, enter:

```
xdsh node1,node2 -o "-v -t" ps
```

4. To execute the commands contained in **myfile** in the **XCAT** context on several node targets, with a fanout of **1**, enter:



```
xdsh node1,node2 -f 1 -e myfile
```

5. To run the **ps** command on node1 and ignore all the **dsh** environment variable except the **DSH\_NODE\_OPTS**, enter:

```
xdsh node1 -X 'DSH_NODE_OPTS' ps
```

6. To run on Linux, the **xdsh** command **rpm -qa | grep xCAT** on the service node fedora9 diskless image, enter:

```
xdsh -i /install/netboot/fedora9/x86_64/service/rootimg "rpm -qa | grep xCAT"
```

7. To run on AIX, the **xdsh** command **lspp -l | grep bos** on the NIM 611dskls spot, enter:

```
xdsh -i 611dskls "/usr/bin/lspp -l | grep bos"
```

8. To cleanup the service node directory that stages the copy of files to the nodes, enter:

```
xdsh servicenoderange -c
```

9. To define the QLogic IB switch as a node and to set up the SSH keys for IB switch **qswitch** with device configuration file **/var/opt/xcat/IBSwitch/Qlogic/config** and user name **username**, enter

```
chdef -t node -o qswitch groups=all nodetype=switch
xdsh qswitch -K -l username --devicetype IBSwitch::Qlogic
```

10. To define the Management Node in the database so you can use **xdsh**, enter

```
xcatconfig -m
```

11. To define the Mellanox switch as a node and run a command to show the ssh keys. **mswitch** with and user name **username**, enter

```
chdef -t node -o mswitch groups=all nodetype=switch
xdsh mswitch -l admin --devicetype IBSwitch::Mellanox 'enable;configure_
↪terminal;show ssh server host-keys'
```

12. To define a BNT Ethernet switch as a node and run a command to create a new vlan with vlan id 3 on the switch.

```
chdef myswitch groups=all
tabch switch=myswitch switches.sshusername=admin switches.sshpassword=passw0rd_
↪switches.protocol=[ssh|telnet]
```

where *admin* and *passw0rd* are the SSH user name and password for the switch.

If it is for Telnet, add *tn:* in front of the user name: *tn:admin*.

```
dsh myswitch --devicetype EthSwitch::BNT 'enable;configure terminal;vlan 3;end;
↪show vlan'
```

13. To run **xdsh** with the non-root userid “user1” that has been setup as an xCAT userid and with **sudo** on node1 and node2 to run as root, do the following, see xCAT doc <https://xcat-docs.readthedocs.io/en/stable/advanced/security/security.html#granting-users-xcat-privileges>:

```
xdsh node1,node2 --sudo -l user1 "cat /etc/passwd"
```

## Files

## SEE ALSO

xdshbak(1)|xdshbak.1, noderange(3)|noderange.3, groupfiles4dsh(1)|groupfiles4dsh.1

## xdshbak.1

## NAME

**xdshbak** - Formats the output of the **xdsh** command.

## SYNOPSIS

**xdshbak** [-c | -x [ -b ] | -h | -q]

## DESCRIPTION

The **xdshbak** command formats output from the **xdsh** command. The **xdshbak** command takes, as input, lines in the following format:

```
host_name: line of output from remote command
```

The **xdshbak** command formats the lines as follows and writes them to standard output. Assume that the output from node3 and node4 is identical, and the **-c** (collapse) flag was specified:

```
HOSTS -----
node1
-----
.
.
lines from xdsh for node1 with hostnames stripped off
.
.
HOSTS -----
node2
-----
.
.
lines from xdsh for node2 with hostnames stripped off
.
.
HOSTS -----
node3, node4
-----
.
```

(continues on next page)

(continued from previous page)

```

.
lines from xdsh for node 3 with hostnames stripped off
.
.

```

When output is displayed from more than one node in collapsed form, the host names are displayed alphabetically. When output is not collapsed, output is displayed sorted alphabetically by host name.

If the **-q** quiet flag is not set then **xdshbak** command writes “.” for each 1000 lines of output processed (to show progress), since it won’t display the output until it has processed all of it.

If the **-x** flag is specified, the extra header lines that **xdshbak** normally displays for each node will be omitted, and the hostname at the beginning of each line is not stripped off, but **xdshbak** still sorts the output by hostname for easier viewing:

```

node1: lines from xdsh for node1
.
.
node2: lines from xdsh for node2
.
.

```

If the **-b** flag is specified in addition to **-x**, the hostname at the beginning of each line is stripped.

## Standard Error

When the **xdshbak** filter is used and standard error messages are generated, all error messages on standard error appear before all standard output messages. This is true with and without the **-c** flag.

## OPTIONS

### **-b**

Strip the host prefix from the beginning of the lines. This only works with the **-x** option.

### **-c**

If the output from multiple nodes is identical it will be collapsed and displayed only once.

### **-x**

Omit the extra header lines that **xdshbak** normally displays for each node. This provides more compact output, but **xdshbak** still sorts the output by node name for easier viewing. This option should not be used with **-c**.

### **-h**

Displays usage information.

### **-q**

Quiet mode, do not display “.” for each 1000 lines of output.

## EXAMPLES

1. To display the results of a command issued on several nodes, in the format used in the Description, enter:

```
xdsh node1,node2,node3 cat /etc/passwd | xdshbak
```

2. To display the results of a command issued on several nodes with identical output displayed only once, enter:

```
xdsh host1,host2,host3 pwd | xdshbak -c
```

3. To display the results of a command issued on several nodes with compact output and be sorted alphabetically by host name, enter:

```
xdsh host1,host2,host3 date | xdshbak -x
```

## SEE ALSO

xdsh(1)|xdsh.1, xcoll(1)|xcoll.1

### xdshcoll.1

## NAME

**xdshcoll** - Formats and consolidates the output of the **xdsh**, **sinv** commands.

## SYNOPSIS

**xdshcoll**

## DESCRIPTION

The **xdshcoll** command formats and consolidates output from the **xdsh**, **sinv** commands. The **xdshcoll** command takes, as input, lines in the following format:

host\_name: line of output from remote command

The **xdshcoll** command formats the lines as follows and writes them to standard output. Assume that the output from node3 and node4 is identical:

```
=====
node1
=====
.
.
lines from xdsh for node1 with hostnames stripped off
.
.
=====
node2
```

(continues on next page)

(continued from previous page)

```
=====
.
.
lines from xdsh for node2 with hostnames stripped off
.
.
=====
node3, node4
=====
.
.
lines from xdsh for node 3 with hostnames stripped off
.
.
```

## EXAMPLES

1. To display the results of a command issued on several nodes, in the format shown in the Description, enter:

```
xdsh node1,node2,node3 cat /etc/passwd | xdshcoll
```

## SEE ALSO

xdshbak(1)|xdshbak.1

## xpbsnodes.1

## NAME

**xpbsnodes** - PBS pbsnodes front-end for a noderange.

## SYNOPSIS

**xpbsnodes** [{*noderange*}] [{**offline** | **clear** | **stat** | **state**}]

**xpbsnodes** [-h | --help] [-v | --version]

## DESCRIPTION

**xpbsnodes** is a front-end to PBS pbsnode but uses xCAT's noderange to specify nodes.

## OPTIONS

**-h|--help** Display usage message.

**-v|--version** Command Version.

**offline|off** Take nodes offline.

**clear|online|on** Take nodes online.

**stat|state** Display PBS node state.

## RETURN VALUE

0 The command completed successfully.

1 An error has occurred.

## EXAMPLES

1. To display status of all PBS nodes, enter:

```
xpbsnodes all stat
```

## FILES

/opt/torque/x86\_64/bin/xpbsnodes

## SEE ALSO

noderange(3)|noderange.3

**man3**

**noderange.3**

## Name

**noderange** - syntax for compactly expressing a list of node names

## Synopsis

*Examples:*

```
node1,node2,node8,node20,group1
node14-node56,node70-node203,group1-group10
node1,node2,node8,node20,node14-node56,node70-node203
node[14-56]
f[1-3]n[1-20]
all,-node129-node256,-frame01-frame03
/node.*
^/tmp/nodes
node10+5
10-15,-13
group1@group2
table.attribute<operator>value
```

## Description

**noderange** is a syntax that can be used in most xCAT commands to conveniently specify a list of nodes. The result is that the command will be applied to a range of nodes, often in parallel.

If you invoke xCAT **noderange** from a shell you may need to quote the **noderange** if the shell would otherwise treat the punctuation marks in the **noderange** as control operators. The affected punctuation marks may include Asterisk (\*), Left Square Bracket ([), Right Square Bracket (]), Circumflex Accent (^), and Overline (~).

**noderange** is a comma-separated list. Each token (text between commas) in the list can be any of the forms listed below:

Individual node or group:

```
node01
group1
```

A range of nodes or groups:

```
node01-node10    (equivalent to: node01,node02,node03,...node10)
node[01-10]      (same as above)
node01:node10    (same as above)
node[01:10]      (same as above)
f[1-2]n[1-3]     (equivalent to: f1n1,f1n2,f1n3,f2n1,f2n2,f2n3)
group1-group3    (equivalent to: group1,group2,group3)
(all the permutations supported above for nodes are also supported for groups)
```

**nodeRange** tries to be intelligent about detecting padding, so you can specify “node001-node200” and it will add the proper number of zeroes to make all numbers 3 digits.

An incremented range of nodes:

```
node10+3 (equivalent to: node10,node11,node12,node13)
```

A node shorthand range of nodes:

```
10-20 (equivalent to: node10,node11,node12,...node20)
10+3 (equivalent to: node10,node11,node12,node13)
```

Currently, the prefix that will be prepended for the above syntax is always “node”. Eventually, the prefix and optional suffix will be settable via the environment variables XCAT\_NODE\_PREFIX and XCAT\_NODE\_SUFFIX, but currently this only works in bypass mode.

A regular expression match of nodes or groups:

```
/node[345].* (will match any nodes that start with node3, node4, or node5)
/group[12].* (will match any groups that start with group1 or group2)
```

The path of a file containing noderanges of nodes or groups:

```
^/tmp/nodelist
```

where /tmp/nodelist can contain entries like:

```
#my node list (this line ignored)
^/tmp/foo #ignored
node01    #node comment
node02
node03
node10-node20
/group[456].*
-node50
```

Node ranges can contain any combination:

```
node01-node30,node40,^/tmp/nodes,/node[13].*,2-10,node50+5
```

Any individual **noderange** may be prefixed with an exclusion operator (default -) with the exception of the file operator (default ^). This will cause that individual noderange to be subtracted from the total resulting list of nodes.

The intersection operator @ calculates the intersection of the left and right sides:

```
group1@group2 (will result in the list of nodes that group1 and group2 have in common)
```

Any combination or multiple combinations of inclusive and exclusive ranges of nodes and groups is legal. There is no precedence implied in the order of the arguments. Exclusive ranges have precedence over inclusive. Parentheses can be used to explicitly specify precedence of any operators.

Nodes have precedence over groups. If a node range match is made then no group range match will be attempted.

All node and group names are validated against the nodelist table. Invalid names are ignored and return nothing.



## xCAT Node Name Format

Throughout this man page the term **xCAT Node Name Format** is used. **xCAT Node Name Format** is defined by the following regex:

```
^([A-Za-z-]+)([0-9]+)(([A-Za-z-]+[A-Za-z0-9-]*)*)
```

In plain English, a node or group name is in **xCAT Node Name Format** if starting from the beginning there are:

- \* one or more alpha characters of any case and any number of “-” in any combination

- \* followed by one or more numbers

- \* then optionally followed by one alpha character of any case or “-”

- \* followed by any combination of case mixed alphanumerics and “-”

**noderange** supports node/group names in *any* format. **xCAT Node Name Format** is **not** required, however some node range methods used to determine range will not be used for non-conformant names.

Example of **xCAT Node Name Format** node/group names:

NODENAME	PREFIX	NUMBER	SUFFIX
node1	node	1	
node001	node	001	
node-001	node-	001	
node-foo-001-bar	node-foo-	001	-bar
node-foo-1bar	node-foo-	1	bar
foo1bar2	foo	1	bar2
rack01unit34	rack	01	unit34
unit34rack01	unit	34	rack01
pos0134	pos	0134	

## Examples

1. Generates a list of all nodes (assuming all is a group) listed in the **nodelist** table less node5 through node10:

```
all, -node5-node10
```

2. Generates a list of nodes 1 through 10 less nodes 3,4,5. Note that node4 is listed twice, first in the range and then at the end. Because exclusion has precedence node4 will be excluded.

```
node1-node10, -node3-node5, node4
```

3. Generates a list of nodes 1 through 10 less nodes 3 and 5.

```
node1-node10, -node3, -node5
```

4. Generates a list of all (assuming `all` is a group) nodes in the **nodelist** table less 17 through 32.

```
-node17-node32, all
```

5. Generates a list of nodes 1 through 128, and user nodes 1 through 4.

```
node1-node128, user1-user4
```

6. Generates a list of all nodes (assuming ``all`` is a group), less nodes in groups rack1 through rack3 (assuming groups rack1, rack2, and rack3 are defined), less nodes 100 through 200, less nodes in the storage group. Note that node150 is listed but is excluded.

```
all,-rack1-rack3,-node100-node200,node150,-storage
```

7. Generates a list of nodes matching the regex `node[23].\*`. That is all nodes that start with node2 or node3 and end in anything or nothing. E.g. node2, node3, node20, node30, node21234 all match.

```
/node[23].*
```

8. Generates a list of nodes which have the value hmc in the nodehm.cons attribute.

```
nodehm.cons==hmc
```

```
nodehm.cons=~hmc
```

9. Generate a list of nodes in the 1st two frames:

```
f[1-2]n[1-42]
```

## Bugs

The special characters used by xCAT **noderange** are also special characters to many shell programs. In particular, the characters ``*`, `[`, `]`, `^`, and `~`` may have to be escaped from the shell.

## SEE ALSO

`models(1)|models.1`

**man5**

**auditlog.5**

## NAME

**auditlog** - a table in the xCAT database.

## SYNOPSIS

**auditlog Attributes:** *recid, audittime, userid, clientname, clienttype, command, noderange, args, status, comments, disable*

## DESCRIPTION

Audit Data log.

### **auditlog Attributes:**

#### **recid**

The record id.

#### **audittime**

The timestamp for the audit entry.

#### **userid**

The user running the command.

#### **clientname**

The client machine, where the command originated.

#### **clienttype**

Type of command: cli, java, webui, other.

#### **command**

Command executed. See auditskipcmds site table attribute to control which commands get logged.

#### **noderange**

The noderange on which the command was run.

#### **args**

The command argument list.

#### **status**

Allowed or Denied.

#### **comments**

Any user-provided notes.

#### **disable**

Do not use. tabprune will not work if set to yes or 1

## SEE ALSO

**nodels(1), chtag(8), tabdump(8), tabedit(8)**

## bootparams.5

### NAME

**bootparams** - a table in the xCAT database.

### SYNOPSIS

**bootparams Attributes:** *node, kernel, initrd, kcmdline, addkcmdline, dhcpstatements, adddhcpstatements, comments, disable*

### DESCRIPTION

Current boot settings to be sent to systems attempting network boot for deployment, stateless, or other reasons. Mostly automatically manipulated by xCAT.

#### **bootparams Attributes:**

##### **node**

The node or group name

##### **kernel**

The kernel that network boot actions should currently acquire and use. Note this could be a chained boot loader such as memdisk or a non-linux boot loader

##### **initrd**

The initial ramdisk image that network boot actions should use (could be a DOS floppy or hard drive image if using memdisk as kernel)

##### **kcmdline**

(Deprecated, use addkcmdline instead) Arguments to be passed to the kernel.

##### **addkcmdline**

User specified kernel options for os provision process (no prefix) or the provisioned os (with prefix "R::"). Multiple options should be delimited with spaces(" ") and surrounded with quotes. To have the same option used for os provision process and for provisioned os, specify that option with and without the prefix: addkcmdline="R::display=3 display=3"

##### **dhcpstatements**

xCAT manipulated custom dhcp statements (not intended for user manipulation)

##### **adddhcpstatements**

Custom dhcp statements for administrator use (not implemented yet)

##### **comments**

Any user-written notes.

##### **disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## boottarget.5

### NAME

**boottarget** - a table in the xCAT database.

### SYNOPSIS

**boottarget Attributes:** *bprofile*, *kernel*, *initrd*, *kcmdline*, *comments*, *disable*

### DESCRIPTION

Specify non-standard initrd, kernel, and parameters that should be used for a given profile.

#### boottarget Attributes:

##### **bprofile**

All nodes with a `nodetype.profile` value equal to this value and `nodetype.os` set to “boottarget”, will use the associated kernel, initrd, and kcmdline.

##### **kernel**

The kernel that network boot actions should currently acquire and use. Note this could be a chained boot loader such as memdisk or a non-linux boot loader

##### **initrd**

The initial ramdisk image that network boot actions should use (could be a DOS floppy or hard drive image if using memdisk as kernel)

##### **kcmdline**

Arguments to be passed to the kernel

##### **comments**

Any user-written notes.

##### **disable**

Set to ‘yes’ or ‘1’ to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## cfgmgt.5

## NAME

**cfgmgt** - a table in the xCAT database.

## SYNOPSIS

**cfgmgt Attributes:** *node, cfgmgr, cfgserver, roles, comments, disable*

## DESCRIPTION

Configuration management data for nodes used by non-xCAT osimage management services to install and configure software on a node.

### cfgmgt Attributes:

#### **node**

The node being managed by the cfgmgr service

#### **cfgmgr**

The name of the configuration manager service. Currently ‘chef’ and ‘puppet’ are supported services.

#### **cfgserver**

The xCAT node name of the chef server or puppet master

#### **roles**

The roles associated with this node as recognized by the cfgmgr for the software that is to be installed and configured. These role names map to chef recipes or puppet manifest classes that should be used for this node. For example, chef OpenStack cookbooks have roles such as mysql-master,keystone, glance, nova-controller, nova-conductor, cinder-all.

#### **comments**

Any user-written notes.

#### **disable**

Set to ‘yes’ or ‘1’ to comment out this row.

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## chain.5

## NAME

**chain** - a table in the xCAT database.

## SYNOPSIS

**chain Attributes:** *node, currstate, currchain, chain, ondiscover, comments, disable*

## DESCRIPTION

Controls what operations are done (and in what order) when a node is discovered and deployed.

### chain Attributes:

#### node

The node name or group name.

#### currstate

The current or next chain step to be executed on this node by xCAT-genesis. Set by xCAT during node discovery or as a result of nodeset.

#### currchain

The chain steps still left to do for this node. This attribute will be automatically adjusted by xCAT while xCAT-genesis is running on the node (either during node discovery or a special operation like firmware update). During node discovery, this attribute is initialized from the chain attribute and updated as the chain steps are executed.

#### chain

A comma-delimited chain of actions to be performed automatically when this node is discovered for the first time. (xCAT and the DHCP server do not recognize the MAC address of the node when xCAT initializes the discovery process.) The last step in this process is to run the operations listed in the chain attribute, one by one. Valid values: boot, runcmd=<cmd>, runimage=<URL>, shell, standby. For example, to have the genesis kernel pause to the shell, use chain=shell.

#### ondiscover

This attribute is currently not used by xCAT. The “nodediscover” operation is always done during node discovery.

#### comments

Any user-written notes.

#### disable

Set to ‘yes’ or ‘1’ to comment out this row.

## SEE ALSO

**nodes(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## deps.5

## NAME

**deps** - a table in the xCAT database.

## SYNOPSIS

**deps Attributes:** *node, nodedep, msdelay, cmd, comments, disable*

## DESCRIPTION

Describes dependencies some nodes have on others. This can be used, e.g., by `rpower -d` to power nodes on or off in the correct order.

### deps Attributes:

#### **node**

The node name or group name.

#### **nodedep**

Comma-separated list of nodes or node groups it is dependent on.

#### **msdelay**

How long to wait between operating on the dependent nodes and the primary nodes.

#### **cmd**

Comma-separated list of which operation this dependency applies to.

#### **comments**

Any user-written notes.

#### **disable**

Set to 'yes' or '1' to comment out this row.



## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## discoverydata.5

### NAME

**discoverydata** - a table in the xCAT database.

### SYNOPSIS

**discoverydata Attributes:** *uuid, node, method, discoverytime, arch, cpucount, cputype, memory, mtm, serial, nic-driver, nicpv4, nichwaddr, nicpci, nicloc, niconboard, nicfirm, switchname, switchaddr, switchdesc, switchport, otherdata, comments, disable*

### DESCRIPTION

Discovery data which sent from genesis.

#### **discoverydata Attributes:**

##### **uuid**

The uuid of the node which send out the discovery request.

##### **node**

The node name which assigned to the discovered node.

##### **method**

The method which handled the discovery request. The method could be one of: switch, blade, profile, sequential.

##### **discoverytime**

The last time that xCAT received the discovery message.

##### **arch**

The architecture of the discovered node. e.g. x86\_64.

##### **cpucount**

The number of cores multiply by threads core supported for the discovered node. e.g. 192.

##### **cputype**

The cpu type of the discovered node. e.g. Intel(R) Xeon(R) CPU E5-2690 0 @ 2.90GHz

##### **memory**

The memory size of the discovered node. e.g. 198460852

##### **mtm**

The machine type model of the discovered node. e.g. 786310X

**serial**

The serial number of the discovered node. e.g. 1052EFB

**nicdriver**

The driver of the nic. The value should be comma separated <nic name!driver name>. e.g. eth0!be2net,eth1!be2net

**nicipv4**

The ipv4 address of the nic. The value should be comma separated <nic name!ipv4 address>. e.g. eth0!10.0.0.212/8

**nicwaddr**

The hardware address of the nic. The should will be comma separated <nic name!hardware address>. e.g. eth0!34:40:B5:BE:DB:B0,eth1!34:40:B5:BE:DB:B4

**nicpci**

The pic device of the nic. The value should be comma separated <nic name!pci device>. e.g. eth0!0000:0c:00.0,eth1!0000:0c:00.1

**nicloc**

The location of the nic. The value should be comma separated <nic name!nic location>. e.g. eth0!Onboard Ethernet 1,eth1!Onboard Ethernet 2

**niconboard**

The onboard info of the nic. The value should be comma separated <nic name!onboard info>. e.g. eth0!1,eth1!2

**nicfirm**

The firmware description of the nic. The value should be comma separated <nic name!fimware description>. e.g. eth0!ServerEngines BE3 Controller,eth1!ServerEngines BE3 Controller

**switchname**

The switch name which the nic connected to. The value should be comma separated <nic name!switch name>. e.g. eth0!c909f06sw01

**switchaddr**

The address of the switch which the nic connected to. The value should be comma separated <nic name!switch address>. e.g. eth0!192.168.70.120

**switchdesc**

The description of the switch which the nic connected to. The value should be comma separated <nic name!switch description>. e.g. eth0!IBM Flex System Fabric EN4093 10Gb Scalable Switch, flash image: version 7.2.6, boot image: version 7.2.6

**switchport**

The port of the switch that the nic connected to. The value should be comma separated <nic name!switch port>. e.g. eth0!INTA2

**otherdata**

The left data which is not parsed to specific attributes (The complete message comes from genesis)

**comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

**SEE ALSO**

**models(1), chtab(8), tabdump(8), tabedit(8)**

**domain.5****NAME**

**domain** - a table in the xCAT database.

**SYNOPSIS**

**domain Attributes:** *node, ou, authdomain, adminuser, adminpassword, type, comments, disable*

**DESCRIPTION**

Mapping of nodes to domain attributes

**domain Attributes:****node**

The node or group the entry applies to

**ou**

For an LDAP described machine account (i.e. Active Directory), the organizational unit to place the system. If not set, defaults to cn=Computers,dc=your,dc=domain

**authdomain**

If a node should participate in an AD domain or Kerberos realm distinct from domain indicated in site, this field can be used to specify that

**adminuser**

Allow a node specific indication of Administrative user. Most will want to just use passwd table to indicate this once rather than by node.

**adminpassword**

Allow a node specific indication of Administrative user password for the domain. Most will want to ignore this in favor of passwd table.

**type**

Type, if any, of authentication domain to manipulate. The only recognized value at the moment is activedirectory.

**comments**

Any user-written notes.

## disable

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## eventlog.5

## NAME

**eventlog** - a table in the xCAT database.

## SYNOPSIS

**eventlog Attributes:** *recid, eventtime, eventtype, monitor, monnode, node, application, component, id, severity, message, rawdata, comments, disable*

## DESCRIPTION

Stores the events occurred.

### eventlog Attributes:

#### recid

The record id.

#### eventtime

The timestamp for the event.

#### eventtype

The type of the event.

#### monitor

The name of the monitor that monitors this event.

#### monnode

The node that monitors this event.

#### node

The node where the event occurred.

#### application

The application that reports the event.

#### component

The component where the event occurred.

**id**

The location or the resource name where the event occurred.

**severity**

The severity of the event. Valid values are: informational, warning, critical.

**message**

The full description of the event.

**rawdata**

The data that associated with the event.

**comments**

Any user-provided notes.

**disable**

Do not use. tabprune will not work if set to yes or 1

**SEE ALSO**

**nodeids(1), chtag(8), tabdump(8), tabedit(8)**

**firmware.5**

**NAME**

**firmware** - a table in the xCAT database.

**SYNOPSIS**

**firmware Attributes:** *node, cfgfile, comments, disable*

**DESCRIPTION**

Maps node to firmware values to be used for setup at node discovery or later

**firmware Attributes:**

**node**

The node id.

**cfgfile**

The file to use.

**comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

## hosts.5

## NAME

**hosts** - a table in the xCAT database.

## SYNOPSIS

**hosts Attributes:** *node, ip, hostnames, otherinterfaces, comments, disable*

## DESCRIPTION

IP addresses and hostnames of nodes. This info is optional and is only used to populate /etc/hosts and DNS via makehosts and makedns. Using regular expressions in this table can be a quick way to populate /etc/hosts.

### hosts Attributes:

#### **node**

The node name or group name.

#### **ip**

The IP address of the node. This is only used in makehosts. The rest of xCAT uses system name resolution to resolve node names to IP addresses.

#### **hostnames**

Hostname aliases added to /etc/hosts for this node. Comma or blank separated list.

#### **otherinterfaces**

Other IP addresses to add for this node. Format: -<ext>:<ip>,<intfhostname>:<ip>,...

#### **comments**

Any user-written notes.

#### **disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## hwinv.5

## NAME

**hwinv** - a table in the xCAT database.

## SYNOPSIS

**hwinv Attributes:** *node, cputype, cpucount, memory, disksize, comments, disable*

## DESCRIPTION

The hardware inventory for the node.

### hwinv Attributes:

#### **node**

The node name or group name.

#### **cputype**

The cpu model name for the node.

#### **cpucount**

The number of cpus for the node.

#### **memory**

The size of the memory for the node in MB.

#### **disksize**

The size of the disks for the node in GB.

#### **comments**

Any user-provided notes.

#### **disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## hypervisor.5

### NAME

**hypervisor** - a table in the xCAT database.

### SYNOPSIS

**hypervisor Attributes:** *node, type, mgr, interface, netmap, defaultnet, cluster, datacenter, preferdirect, comments, disable*

### DESCRIPTION

Hypervisor parameters

#### hypervisor Attributes:

##### **node**

The node or static group name

##### **type**

The plugin associated with hypervisor specific commands such as revacuate

##### **mgr**

The virtualization specific manager of this hypervisor when applicable

##### **interface**

The definition of interfaces for the hypervisor. The format is [network-name:interfacename:bootprotocol:IP:netmask:gateway] that split with | for each interface

##### **netmap**

Optional mapping of useful names to relevant physical ports. For example, 10ge=vmnic\_16.0&vmnic\_16.1,ge=vmnic1 would be requesting two virtual switches to be created, one called 10ge with vmnic\_16.0 and vmnic\_16.1 bonded, and another simply connected to vmnic1. Use of this allows abstracting guests from network differences amongst hypervisors

##### **defaultnet**

Optionally specify a default network entity for guests to join to if they do not specify.

##### **cluster**

Specify to the underlying virtualization infrastructure a cluster membership for the hypervisor.

##### **datacenter**

Optionally specify a datacenter for the hypervisor to exist in (only applicable to VMWare)



## preferdirect

If a mgr is declared for a hypervisor, xCAT will default to using the mgr for all operations. If this is field is set to yes or 1, xCAT will prefer to directly communicate with the hypervisor if possible

## comments

## disable

## SEE ALSO

**nodels(1), chtag(8), tabdump(8), tabedit(8)**

## ipmi.5

## NAME

**ipmi** - a table in the xCAT database.

## SYNOPSIS

**ipmi Attributes:** *node, bmc, bmcport, taggedvlan, bmcid, username, password, comments, disable*

## DESCRIPTION

Settings for nodes that are controlled by an on-board BMC via IPMI.

## ipmi Attributes:

### node

The node name or group name.

### bmc

The hostname of the BMC adapter.

### bmcport

In systems with selectable shared/dedicated ethernet ports, this parameter can be used to specify the preferred port. 0 means use the shared port, 1 means dedicated, blank is to not assign.

The following special cases exist **for** IBM System x servers:

For x3755 M3 systems, **0** means **use the** dedicated port, **1** means shared, blank is to **not** assign.

For certain systems which have a mezzaine **or** ML2 adapter, there is a second value to include:

For x3750 M4 (Model **8722**):

(continues on next page)

(continued from previous page)

```

0 2    1st 1Gbps interface for LOM
0 0    1st 10Gbps interface for LOM
0 3    2nd 1Gbps interface for LOM
0 1    2nd 10Gbps interface for LOM

```

For x3750 M4 (Model 8752), x3850/3950 X6, dx360 M4, x3550 M4, and x3650 M4:

```

0      Shared (1st onboard interface)
1      Dedicated
2 0    First interface on ML2 or mezzanine adapter
2 1    Second interface on ML2 or mezzanine adapter
2 2    Third interface on ML2 or mezzanine adapter
2 3    Fourth interface on ML2 or mezzanine adapter

```

#### taggedvlan

bmcsetup script will configure the network interface of the BMC to be tagged to the VLAN specified.

#### bmcid

Unique identified data used by discovery processes to distinguish known BMCs from unrecognized BMCs

#### username

The BMC userid. If not specified, the key=ipmi row in the passwd table is used as the default.

#### password

The BMC password. If not specified, the key=ipmi row in the passwd table is used as the default.

#### comments

Any user-written notes.

#### disable

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## iscsi.5

## NAME

**iscsi** - a table in the xCAT database.

## SYNOPSIS

**iscsi Attributes:** *node, server, target, lun, iname, file, userid, passwd, kernel, kcmdline, initrd, comments, disable*

## DESCRIPTION

Contains settings that control how to boot a node from an iSCSI target

### iscsi Attributes:

#### **node**

The node name or group name.

#### **server**

The server containing the iscsi boot device for this node.

#### **target**

The iscsi disk used for the boot device for this node. Filled in by xCAT.

#### **lun**

LUN of boot device. Per RFC-4173, this is presumed to be 0 if unset. tgtd often requires this to be 1

#### **iname**

Initiator name. Currently unused.

#### **file**

The path on the server of the OS image the node should boot from.

#### **userid**

The userid of the iscsi server containing the boot device for this node.

#### **passwd**

The password for the iscsi server containing the boot device for this node.

#### **kernel**

The path of the linux kernel to boot from.

#### **kcmdline**

The kernel command line to use with iSCSI for this node.

### **initrd**

The initial ramdisk to use when network booting this node.

### **comments**

Any user-written notes.

### **disable**

Set to 'yes' or '1' to comment out this row.

## **SEE ALSO**

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

## **kit.5**

## **NAME**

**kit** - a table in the xCAT database.

## **SYNOPSIS**

**kit Attributes:** *kitname, basename, description, version, release, ostype, isinternal, kitdeployparams, kitdir, comments, disable*

## **DESCRIPTION**

This table stores all kits added to the xCAT cluster.

### **kit Attributes:**

#### **kitname**

The unique generated kit name, when kit is added to the cluster.

#### **basename**

The kit base name

#### **description**

The Kit description.

#### **version**

The kit version

#### **release**

The kit release

#### **ostype**

The kit OS type. Linux or AIX.

### **isinternal**

A flag to indicated if the Kit is internally used. When set to 1, the Kit is internal. If 0 or undefined, the kit is not internal.

### **kitdeployparams**

The file containing the default deployment parameters for this Kit. These parameters are added to the OS Image definition.s list of deployment parameters when one or more Kit Components from this Kit are added to the OS Image.

### **kitdir**

The path to Kit Installation directory on the Mgt Node.

### **comments**

Any user-written notes.

### **disable**

Set to 'yes' or '1' to comment out this row.

## **SEE ALSO**

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

## **kitcomponent.5**

### **NAME**

**kitcomponent** - a table in the xCAT database.

### **SYNOPSIS**

**kitcomponent Attributes:** *kitcompname, description, kitname, kitreponame, basename, version, release, serverroles, kitpkgdeps, prerequisite, driverpacks, kitcompdeps, postbootscripts, genimage\_postinstall, exlist, comments, disable*

### **DESCRIPTION**

This table stores all kit components added to the xCAT cluster.

### **kitcomponent Attributes:**

#### **kitcompname**

The unique Kit Component name. It is auto-generated when the parent Kit is added to the cluster.

#### **description**

The Kit component description.

#### **kitname**

The Kit name which this Kit Component belongs to.

**kitreponame**

The Kit Package Repository name which this Kit Component belongs to.

**basename**

Kit Component basename.

**version**

Kit Component version.

**release**

Kit Component release.

**serverroles**

The types of servers that this Kit Component can install on. Valid types are: mgtnode, servicenode, compute

**kitpkgdeps**

Comma-separated list of packages that this kit component depends on.

**prerequisite**

Prerequisite for this kit component, the prerequisite includes ospkgdeps,preinstall,preupgrade,preuninstall scripts

**driverpacks**

Comma-separated List of driver package names. These must be full names like: pkg1-1.0-1.x86\_64.rpm.

**kitcompdeps**

Comma-separated list of kit components that this kit component depends on.

**postbootscripts**

Comma-separated list of postbootscripts that will run during the node boot.

**genimage\_postinstall**

Comma-separated list of postinstall scripts that will run during the genimage.

**exlist**

Exclude list file containing the files/directories to exclude when building a diskless image.

**comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## kitrepo.5

## NAME

**kitrepo** - a table in the xCAT database.

## SYNOPSIS

**kitrepo Attributes:** *kitreponame, kitname, osbasename, osmajorversion, osminorversion, osarch, compat\_osbasenames, kitrepodir, comments, disable*

## DESCRIPTION

This table stores all kits added to the xCAT cluster.

### kitrepo Attributes:

#### **kitreponame**

The unique generated kit repo package name, when kit is added to the cluster.

#### **kitname**

The Kit name which this Kit Package Repository belongs to.

#### **osbasename**

The OS distro name which this repository is based on.

#### **osmajorversion**

The OS distro major version which this repository is based on.

#### **osminorversion**

The OS distro minor version which this repository is based on. If this attribute is not set, it means that this repo applies to all minor versions.

#### **osarch**

The OS distro arch which this repository is based on.

#### **compat\_osbasenames**

List of compatible OS base names.

#### **kitrepodir**

The path to Kit Repository directory on the Mgt Node.

#### **comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

**SEE ALSO**

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

**kvm\_masterdata.5**

**NAME**

**kvm\_masterdata** - a table in the xCAT database.

**SYNOPSIS**

**kvm\_masterdata Attributes:** *name, xml, comments, disable*

**DESCRIPTION**

Persistent store for KVM plugin for masters

**kvm\_masterdata Attributes:**

**name**

The name of the relevant master

**xml**

The XML description to be customized for clones of this master

**comments**

**disable**

Set to 'yes' or '1' to comment out this row.

**SEE ALSO**

**nodels(1), chtab(8), tabdump(8), tabedit(8)**



## kvm\_nodedata.5

### NAME

**kvm\_nodedata** - a table in the xCAT database.

### SYNOPSIS

**kvm\_nodedata Attributes:** *node, xml, comments, disable*

### DESCRIPTION

Persistent store for KVM plugin, not intended for manual modification.

#### **kvm\_nodedata Attributes:**

##### **node**

The node corresponding to the virtual machine

##### **xml**

The XML description generated by xCAT, fleshed out by libvirt, and stored for reuse

##### **comments**

Any user-written notes.

##### **disable**

Set to 'yes' or '1' to comment out this row.

### SEE ALSO

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

## linuximage.5

### NAME

**linuximage** - a table in the xCAT database.

## SYNOPSIS

**linuximage Attributes:** *imagename, template, boottarget, addkcmdline, pkglist, pkgdir, otherpkglist, otherpkgdir, exlist, postinstall, rootimgdir, kerneldir, nodebootif, otherifce, netdrivers, kernelver, krpmmver, permission, dump, crashkernelsize, partitionfile, driverupdatesrc, comments, disable*

## DESCRIPTION

Information about a Linux operating system image that can be used to deploy cluster nodes.

### linuximage Attributes:

#### **imagename**

The name of this xCAT OS image definition.

#### **template**

The fully qualified name of the template file that will be used to create the OS installer configuration file for stateful installations (e.g. kickstart for RedHat, autoyast for SLES).

#### **boottarget**

The name of the boottarget definition. When this attribute is set, xCAT will use the kernel, initrd and kernel params defined in the boottarget definition instead of the default.

#### **addkcmdline**

User specified kernel options for os provision process(no prefix) or the provisioned os(with prefix "R::"). The options should be delimited with spaces(" "). This attribute is ignored if linuximage.boottarget is set.

#### **pkglist**

The fully qualified name of the file that stores the distro packages list that will be included in the image. Make sure that if the pkgs in the pkglist have dependency pkgs, the dependency pkgs should be found in one of the pkgdir

#### **pkgdir**

The name of the directory where the distro packages are stored. It could be set to multiple paths. The multiple paths must be separated by ",". The first path in the value of osimage.pkgdir must be the OS base pkg dir path, such as pkgdir=/install/rhels6.2/x86\_64/install/updates . In the os base pkg path, there are default repository data. And in the other pkg path(s), the users should make sure there are repository data. If not, use "createrepo" command to create them. For ubuntu, multiple mirrors can be specified in the pkgdir attribute, the mirrors must be prefixed by the protocol(http/ssh) and delimited with "," between each other.

#### **otherpkglist**

The fully qualified name of the file that stores non-distro package lists that will be included in the image. It could be set to multiple paths. The multiple paths must be separated by ",".

#### **otherpkgdir**

The base directory and urls of internet repos from which the non-distro packages are retrived. Only 1 local directory is supported at present. The entries should be delimited with comma ",". Currently, the internet repos are only supported on Ubuntu and Redhat.

#### **exlist**

The fully qualified name of the file that stores the file names and directory names that will be excluded from the image during packimage command. It is used for diskless image only.

### postinstall

Supported in diskless image only. The fully qualified name of the scripts and the user-specified arguments running in non-chroot mode after the package installation but before initrd generation during genimage. If multiple scripts are specified, they should be separated with comma “,”. The arguments passed to each postinstall script include 4 implicit arguments(<rootimage path>,<os version>,<os arch>,<profile>) and the user-specified arguments. A set of osimage attributes are exported as the environment variables to be used in the postinstall scripts:

```
IMG_ARCH(The architecture of the osimage, such as "ppc64le","x86_64"),
IMG_NAME(The name of the osimage, such as "rhels7.3-ppc64le-netboot-compute"),
IMG_OSVER(The os release of the osimage, such as "rhels7.3","sles11.4"),
IMG_KERNELVERSION(the "kernelver" attribute of the osimage),
IMG_PROFILE(the profile of the osimage, such as "service","compute"),
IMG_PKGLIST(the "pkglist" attribute of the osimage),
IMG_PKGDIR(the "pkgdir" attribute of the osimage),
IMG_OTHERPKGLIST(the "otherpkglist" attribute of the osimage),
IMG_OTHERPKGDIR(the "otherpkgdir" attribute of the osimage),
IMG_ROOTIMGDIR(the "rootimgdir" attribute of the osimage)
```

### rootimgdir

The directory name where the image is stored. It is generally used for diskless image. it also can be used in sysclone environment to specify where the image captured from golden client is stored. in sysclone environment, rootimgdir is generally assigned to some default value by xcat, but you can specify your own store directory. just one thing need to be noticed, wherever you save the image, the name of last level directory must be the name of image. for example, if your image name is testimage and you want to save this image under home directoy, rootimgdir should be assigned to value /home/testimage/

### kerneldir

The directory name where the 3rd-party kernel is stored. It is used for diskless image only.

### nodebootif

The network interface the stateless/statelite node will boot over (e.g. eth0)

### otherifce

Other network interfaces (e.g. eth1) in the image that should be configured via DHCP

### netdrivers

The ethernet device drivers of the nodes which will use this linux image, at least the device driver for the nodes' installnic should be included

### kernelver

The version of linux kernel used in the linux image. If the kernel version is not set, the default kernel in rootimgdir will be used

### krpmver

The rpm version of kernel packages (for SLES only). If it is not set, the default rpm version of kernel packages will be used.

### permission

The mount permission of /.statelite directory is used, its default value is 755

**dump**

The NFS directory to hold the Linux kernel dump file (vmcore) when the node with this image crashes, its format is “nfs://<nfs\_server\_ip>/<kdump\_path>”. If you want to use the node’s “xcatmaster” (its SN or MN), <nfs\_server\_ip> can be left blank. For example, “nfs:///<kdump\_path>” means the NFS directory to hold the kernel dump file is on the node’s SN, or MN if there’s no SN.

**crashkernelsize**

the size that assigned to the kdump kernel. If the kernel size is not set, 256M will be the default value.

**partitionfile**

**Only available for diskful osimages and statelite osimages(localdisk enabled). The full path of the partition file or the script to generate the partition file. The valid value includes:**

“<the absolute path of the partition file>”: For diskful osimages, the partition file contains the partition definition that will be inserted directly into the template file for os installation. The syntax and format of the partition file should confirm to the corresponding OS installer of the Linux distributions(e.g. kickstart for RedHat, autoyast for SLES, pressed for Ubuntu). For statelite osimages, when the localdisk is enabled, the partition file with specific syntax and format includes the partition scheme of the local disk, please refer to the statelite documentation for details. “s:<the absolute path of the partition script>”: a shell script to generate the partition file “/tmp/partitionfile” inside the installer before the installation start. “d:<the absolute path of the disk name file>”: only available for ubuntu osimages, includes the name(s) of the disks to partition in traditional, non-devfs format(e.g. /dev/sdx, not e.g. /dev/discs/disc0/disc), and be delimited with space. All the disks involved in the partition file should be specified. “s:d:<the absolute path of the disk script>”: only available for ubuntu osimages, a script to generate the disk name file “/tmp/xcat.install\_disk” inside the debian installer. This script is run in the “pressed/early\_command” section. “c:<the absolute path of the additional pressed config file>”: only available for ubuntu osimages, contains the additional pressed entries in “d-i ...” form. This can be used to specify some additional preseed options to support RAID or LVM in Ubuntu. “s:c:<the absolute path of the additional pressed config script>”: only available for ubuntu osimages, runs in pressed/early\_command and set the preseed values with “debconf-set”. The multiple values should be delimited with comma “,”

**driverupdatesrc**

The source of the drivers which need to be loaded during the boot. Two types of driver update source are supported: Driver update disk and Driver rpm package. The value for this attribute should be comma separated sources. Each source should be the format tab:full\_path\_of\_source\_file. The tab keyword can be: dud (for Driver update disk) and rpm (for driver rpm). If missing the tab, the rpm format is the default. e.g. dud:/install/dud/dd.img,rpm:/install/rpm/d.rpm

**comments**

Any user-written notes.

**disable**

Set to ‘yes’ or ‘1’ to comment out this row.

## SEE ALSO

`models(1)`, `chtab(8)`, `tabdump(8)`, `tabedit(8)`

## litefile.5

## NAME

**litefile** - a table in the xCAT database.

## SYNOPSIS

**litefile Attributes:** *image, file, options, comments, disable*

## DESCRIPTION

The litefile table specifies the directories and files on the statelite nodes that should be readwrite, persistent, or readonly overlay. All other files in the statelite nodes come from the readonly statelite image.

### litefile Attributes:

#### image

The name of the image (as specified in the osimage table) that will use these options on this dir/file. You can also specify an image group name that is listed in the osimage.groups attribute of some osimages. 'ALL' means use this row for all images.

#### file

The full pathname of the file. e.g: /etc/hosts. If the path is a directory, then it should be terminated with a '/'.

#### options

Options for the file:

`tmpfs` - It is the default option **if** you leave the options column blank. It provides a file **or** directory **for** the node to **use when** booting, its permission will be the same as the original version on the server. In most cases, it is **read-write**; however, on the **next** statelite boot, the original version of the file **or** directory on the server will be used, it means it is **non-persistent**. This option can be performed on files **and** directories..

`rw` - Same as above. Its name "**rw**" does NOT mean it always be **read-write**, even in most cases it is **read-write**. Do **not** confuse it with the "**rw**" permission in the file **system**.

`persistent` - It provides a mounted file **or** directory that is copied to the xCAT persistent location **and then** over-mounted on the **local** file **or** directory. Anything written to that file **or** directory is preserved. It means, **if** the file/directory does not exist at first, it will be copied to the

(continues on next page)

(continued from previous page)

→persistent location. Next time the file/directory in the persistent location.  
→will be used. The file/directory will be persistent across reboots. Its.  
→permission will be the same as the original one in the statelite location.  
→It requires the statelite table to be filled out with a spot **for** persistent.  
→statelite. This option can be performed on files **and** directories.

con - The contents of the pathname are concatenated to the contents of the.  
→existing file. For this directive the searching in the litetree hierarchy.  
→does **not** stop when the first match is found. All files found in the.  
→hierarchy will be concatenated to the file when found. The permission of the.  
→file will be "-rw-r--r--", which means it is **read-write for** the root user,  
→but readonly **for** the others. It is non-persistent, when the node reboots,  
→all changes to the file will be lost. It can only be performed on files. Do  
→**not use it for** one directory.

ro - The file/directory will be overmounted read-only on the local file/  
→directory. It will be located in the directory hierarchy specified in the.  
→litetree table. Changes made to this file **or** directory on the server will be.  
→immediately seen in this file/directory on the node. This option requires  
→that the file/directory to be mounted must be available in one of the.  
→entries in the litetree table. This option can be performed on files **and**  
→directories.

link - It provides one file/directory for the node to use when booting, it is.  
→copied from the server, and will be placed in tmpfs on the booted node. In.  
→the local file system of the booted node, it is one symbolic link to one.  
→file/directory in tmpfs. And the permission of the symbolic link is  
→"lrwxrwxrwx", which is **not** the real permission of the file/directory on the.  
→node. So for some application sensitive to file permissions, it will be one.  
→issue to use "link" as its option, for example, "/root/.ssh/", which is used  
→for SSH, should NOT use "link" as its option. It is non-persistent, when the.  
→node is rebooted, all changes to the file/directory will be lost. This  
→option can be performed on files and directories.

link,con - It works similar to the "con" option. All the files found in the.  
→litetree hierarchy will be concatenated to the file when found. The final.  
→file will be put to the tmpfs on the booted node. In the **local file system**  
→of the booted node, it is one symbolic link to the file/directory in tmpfs.  
→It is non-persistent, when the node is rebooted, all changes to the file  
→will be lost. The option can only be performed on files.

link,persistent - It provides a mounted file **or** directory that is copied to.  
→the xCAT persistent location **and then** over-mounted to the tmpfs on the.  
→booted node, **and** finally the symbolic link in the **local file system** will be.  
→linked to the over-mounted tmpfs file/directory on the booted node. The file/  
→directory will be persistent across reboots. The permission of the file/  
→directory where the symbolic link points to will be the same as the original.  
→one in the statelite location. It requires the statelite table to be filled  
→out with a spot **for** persistent statelite. The option can be performed on  
→files **and** directories.

link,ro - The file is readonly, **and** will be placed in tmpfs on the booted node.

(continues on next page)

(continued from previous page)

```
→ In the local file system of the booted node, it is one symbolic link to the
→ tmpfs. It is non-persistent, when the node is rebooted, all changes to the
→ file/directory will be lost. This option requires that the file/directory to
→ be mounted must be available in one of the entries in the litetree table.
→ The option can be performed on files and directories.
```

#### comments

Any user-written notes.

#### disable

Set to 'yes' or '1' to comment out this row.

### SEE ALSO

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

#### litetree.5

#### NAME

**litetree** - a table in the xCAT database.

#### SYNOPSIS

**litetree Attributes:** *priority, image, directory, mntopts, comments, disable*

#### DESCRIPTION

Directory hierarchy to traverse to get the initial contents of node files. The files that are specified in the litefile table are searched for in the directories specified in this table.

#### litetree Attributes:

##### priority

This number controls what order the directories are searched. Directories are searched from smallest priority number to largest.

##### image

The name of the image (as specified in the osimage table) that will use this directory. You can also specify an image group name that is listed in the osimage.groups attribute of some osimages. 'ALL' means use this row for all images.

##### directory

The location (hostname:path) of a directory that contains files specified in the litefile table. Variables are allowed. E.g: \$noderes.nfsserver://xcatmasternode/install/\$node/#CMD=uname-r#/

##### mntopts

A comma-separated list of options to use when mounting the litetree directory. (Ex. 'soft') The default is to do a 'hard' mount.

**comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

**SEE ALSO**

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

**mac.5**

**NAME**

**mac** - a table in the xCAT database.

**SYNOPSIS**

**mac Attributes:** *node, interface, mac, comments, disable*

**DESCRIPTION**

The MAC address of the node's install adapter. Normally this table is populated by getmacs or node discovery, but you can also add entries to it manually.

**mac Attributes:**

**node**

The node name or group name.

**interface**

The adapter interface name that will be used to install and manage the node. E.g. eth0 (for linux) or en0 (for AIX).)

**mac**

The mac address or addresses for which xCAT will manage static bindings for this node. This may be simply a mac address, which would be bound to the node name (such as "01:02:03:04:05:0E"). This may also be a "|" delimited string of "mac address!hostname" format (such as "01:02:03:04:05:0E!node5|01:02:03:04:05:0F!node6-eth1"). If there are multiple nics connected to Management Network(usually for bond), in order to make sure the OS deployment finished successfully, the macs of those nics must be able to resolve to same IP address. First, users have to create alias of the node for each mac in the Management Network through either: 1. adding the alias into /etc/hosts for the node directly or: 2. setting the alias to the "hostnames" attribute and then run "makehost" against the node.



Then, configure the “mac” attribute of the node like “mac1!node|mac2!node-alias”. For the first mac address (mac1 in the example) set in “mac” attribute, do not need to set a “node name” string for it since the nodename of the node will be used for it by default.

#### comments

Any user-written notes.

#### disable

Set to ‘yes’ or ‘1’ to comment out this row.

### SEE ALSO

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

#### mic.5

### NAME

**mic** - a table in the xCAT database.

### SYNOPSIS

**mic Attributes:** *node, host, id, nodetype, bridge, onboot, vlog, powermgt, comments, disable*

### DESCRIPTION

The host, slot id and configuration of the mic (Many Integrated Core).

#### mic Attributes:

##### node

The node name or group name.

##### host

The host node which the mic card installed on.

##### id

The device id of the mic node.

##### nodetype

The hardware type of the mic node. Generally, it is mic.

##### bridge

The virtual bridge on the host node which the mic connected to.

##### onboot

Set mic to autoboot when mpss start. Valid values: yes|no. Default is yes.

##### vlog

Set the Verbose Log to console. Valid values: yes|no. Default is no.

**powermgt**

Set the Power Management for mic node. This attribute is used to set the power management state that mic may get into when it is idle. Four states can be set: cpufreq, corec6, pc3 and pc6. The valid value for powermgt attribute should be [cpufreq=<on|off>]![corec6=<on|off>]![pc3=<on|off>]![pc6=<on|off>]. e.g. cpufreq=on!corec6=off!pc3=on!pc6=off. Refer to the doc of mic to get more information for power management.

**comments**

Any user-provided notes.

**disable**

Do not use. tabprune will not work if set to yes or 1

**SEE ALSO**

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

**monitoring.5****NAME**

**monitoring** - a table in the xCAT database.

**SYNOPSIS**

**monitoring Attributes:** *name, nodestatmon, comments, disable*

**DESCRIPTION**

Controls what external monitoring tools xCAT sets up and uses. Entries should be added and removed from this table using the provided xCAT commands monstart and monstop.

**monitoring Attributes:****name**

The name of the monitoring plug-in module. The plug-in must be put in /lib/perl/xCAT\_monitoring/. See the man page for monstart for details.

**nodestatmon**

Specifies if the monitoring plug-in is used to feed the node status to the xCAT cluster. Any one of the following values indicates “yes”: y, Y, yes, Yes, YES, 1. Any other value or blank (default), indicates “no”.

**comments**

Any user-written notes.

**disable**

Set to ‘yes’ or ‘1’ to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

**monsetting.5**

## NAME

**monsetting** - a table in the xCAT database.

## SYNOPSIS

**monsetting Attributes:** *name, key, value, comments, disable*

## DESCRIPTION

Specifies the monitoring plug-in specific settings. These settings will be used by the monitoring plug-in to customize the behavior such as event filter, sample interval, responses etc. Entries should be added, removed or modified by **chtab** command. Entries can also be added or modified by the **monstart** command when a monitoring plug-in is brought up.

### monsetting Attributes:

#### name

The name of the monitoring plug-in module. The plug-in must be put in `/lib/perl/xCAT_monitoring/`. See the man page for **monstart** for details.

#### key

Specifies the name of the attribute. The valid values are specified by each monitoring plug-in. Use “**monls name -d**” to get a list of valid keys.

#### value

Specifies the value of the attribute.

#### comments

Any user-written notes.

#### disable

Set to ‘yes’ or ‘1’ to comment out this row.

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## mp.5

## NAME

**mp** - a table in the xCAT database.

## SYNOPSIS

**mp Attributes:** *node, mpa, id, nodetype, comments, disable*

## DESCRIPTION

Contains the hardware control info specific to blades. This table also refers to the mpa table, which contains info about each Management Module.

### mp Attributes:

#### **node**

The blade node name or group name.

#### **mpa**

The management module used to control this blade.

#### **id**

The slot number of this blade in the BladeCenter chassis.

#### **nodetype**

The hardware type for mp node. Valid values: mm, cmm, blade.

#### **comments**

Any user-written notes.

#### **disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## mpa.5

## NAME

**mpa** - a table in the xCAT database.

## SYNOPSIS

**mpa Attributes:** *mpa, username, password, displayname, slots, urlpath, comments, disable*

## DESCRIPTION

Contains info about each Management Module and how to access it.

### mpa Attributes:

#### mpa

Hostname of the management module.

#### username

Userid to use to access the management module.

#### password

Password to use to access the management module. If not specified, the key=blade row in the passwd table is used as the default.

#### displayname

Alternative name for BladeCenter chassis. Only used by PCM.

#### slots

The number of available slots in the chassis. For PCM, this attribute is used to store the number of slots in the following format: <slot rows>,<slot columns>,<slot orientation> Where:

```
<slot rows> = number of rows of slots in chassis
<slot columns> = number of columns of slots in chassis
<slot orientation> = set to 0 if slots are vertical, and set to 1 if slots of
↳horizontal
```

#### urlpath

URL path for the Chassis web interface. The full URL is built as follows: <hostname>/<urlpath>

#### comments

Any user-written notes.

#### disable

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## networks.5

### NAME

**networks** - a table in the xCAT database.

### SYNOPSIS

**networks Attributes:** *netname, net, mask, mgtifname, gateway, dhcpserver, tftpserver, nameservers, ntpservers, logservers, dynamicrange, staticrange, staticrangeincrement, nodehostname, ddnsdomain, vlanid, domain, mtu, comments, disable*

### DESCRIPTION

Describes the networks in the cluster and info necessary to set up nodes on that network.

#### networks Attributes:

##### **netname**

Name used to identify this network definition.

##### **net**

The network address.

##### **mask**

The network mask.

##### **mgtifname**

The interface name of the management/service node facing this network. !remote!<nicname> indicates a non-local network on a specific nic for relay DHCP.

##### **gateway**

The network gateway. It can be set to an ip address or the keyword <xcatmaster>, the keyword <xcatmaster> indicates the cluster-facing ip address configured on this management node or service node. Leaving this field blank means that there is no gateway for this network.

##### **dhcpserver**

The DHCP server that is servicing this network. Required to be explicitly set for pooled service node operation.

##### **tftpserver**

The TFTP server that is servicing this network. If not set, the DHCP server is assumed.

**nameservers**

A comma delimited list of DNS servers that each node in this network should use. This value will end up in the nameserver settings of the `/etc/resolv.conf` on each node in this network. If this attribute value is set to the IP address of an xCAT node, make sure DNS is running on it. In a hierarchical cluster, you can also set this attribute to “<xcatmaster>” to mean the DNS server for each node in this network should be the node that is managing it (either its service node or the management node). Used in creating the DHCP network definition, and DNS configuration.

**ntpservers**

The ntp servers for this network. Used in creating the DHCP network definition. Assumed to be the DHCP server if not set.

**logservers**

The log servers for this network. Used in creating the DHCP network definition. Assumed to be the DHCP server if not set.

**dynamicrange**

The IP address range used by DHCP to assign dynamic IP addresses for requests on this network. This should not overlap with entities expected to be configured with static host declarations, i.e. anything ever expected to be a node with an address registered in the mac table.

**staticrange**

The IP address range used to dynamically assign static IPs to newly discovered nodes. This should not overlap with the dynamicrange nor overlap with entities that were manually assigned static IPs. The format for the attribute value is: <startip>-<endip>.

**staticrangeincrement****nodehostname**

A regular expression used to specify node name to network-specific hostname. i.e. “/z/-secondary/” would mean that the hostname of “n1” would be n1-secondary on this network. By default, the nodename is assumed to equal the hostname, followed by nodename-interfacesname.

**ddnsdomain**

A domain to be combined with nodename to construct FQDN for DDNS updates induced by DHCP. This is not passed down to the client as “domain”

**vlanid**

The vlan ID if this network is within a vlan.

**domain**

The DNS domain name (ex. cluster.com).

**mtu**

The default MTU for the network, If multiple networks are applied to the same nic on the SN and/or CN, the MTU shall be the same for those networks.

**comments**

Any user-written notes.

**disable**

Set to ‘yes’ or ‘1’ to comment out this row.

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## nics.5

## NAME

**nics** - a table in the xCAT database.

## SYNOPSIS

**nics Attributes:** *node*, *nicips*, *nichostnamesuffixes*, *nichostnameprefixes*, *nictypes*, *niccustomscripts*, *nicnetworks*, *nicaliases*, *nicextraparams*, *nicdevices*, *nicsadapter*, *comments*, *disable*

## DESCRIPTION

Stores NIC details.

### nics Attributes:

#### node

The node or group name.

#### nicips

Comma-separated list of IP addresses per NIC.

**To specify one ip address per NIC:**

<nic1>!<ip1>,<nic2>!<ip2>,..., for example, eth0!10.0.0.100,ib0!11.0.0.100

**To specify multiple ip addresses per NIC:**

<nic1>!<ip1>|<ip2>,<nic2>!<ip1>|<ip2>,..., for example, eth0!10.0.0.100|fd55::214:5eff:fe15:849b,ib0!11.0.0.100|2001::1

The xCAT object definition commands support to use `nicips.<nicname>` as the sub attributes.

Note: The primary IP address must also be stored in the `hosts.ip` attribute. The `nichostnamesuffixes` should specify one hostname suffix for each ip address.

#### nichostnamesuffixes

Comma-separated list of hostname suffixes per NIC.

**If only one ip address is associated with each NIC:**

<nic1>!<ext1>,<nic2>!<ext2>,..., for example, eth0!-eth0,ib0!-ib0

**If multiple ip addresses are associated with each NIC:**

<nic1>!<ext1>|<ext2>,<nic2>!<ext1>|<ext2>,..., for example, eth0!-eth0|-eth0-ipv6,ib0!-ib0|-ib0-ipv6.

The xCAT object definition commands support to use `nichostnamesuffixes.<nicname>` as the sub attributes.



Note: According to DNS rules a hostname must be a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9) and minus sign (-). When you are specifying "nichostnamesuffixes" or "nicaliases" make sure the resulting hostnames will conform to this naming convention

## nichostnameprefixes

Comma-separated list of hostname prefixes per NIC.

**If only one ip address is associated with each NIC:**

<nic1>|<ext1>,<nic2>|<ext2>,..., for example, eth0!eth0-,ib0!ib-

**If multiple ip addresses are associated with each NIC:**

<nic1>|<ext1>|<ext2>,<nic2>|<ext1>|<ext2>,..., for example, eth0!eth0-|eth0-ipv6i-,ib0!ib-|ib-ipv6-.

The xCAT object definition commands support to use nichostnameprefixes.<nicname> as the sub attributes. Note: According to DNS rules a hostname must be a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9) and minus sign (-). When you are specifying "nichostnameprefixes" or "nicaliases" make sure the resulting hostnames will conform to this naming convention

## nictypes

Comma-separated list of NIC types per NIC. <nic1>|<type1>,<nic2>|<type2>, e.g. eth0!Ethernet,ib0!Infiniband. The xCAT object definition commands support to use nictypes.<nicname> as the sub attributes.

## niccustomscripts

Comma-separated list of custom scripts per NIC. <nic1>|<script1>,<nic2>|<script2>, e.g. eth0!configeth eth0, ib0!configib ib0. The xCAT object definition commands support to use niccustomscripts.<nicname> as the sub attribute .

## nicnetworks

Comma-separated list of networks connected to each NIC.

**If only one ip address is associated with each NIC:**

<nic1>|<network1>,<nic2>|<network2>, for example, eth0!10\_0\_0\_0-255\_255\_0\_0, ib0!11\_0\_0\_0-255\_255\_0\_0

**If multiple ip addresses are associated with each NIC:**

<nic1>|<network1>|<network2>,<nic2>|<network1>|<network2>, for example, eth0!10\_0\_0\_0-255\_255\_0\_0|fd55:faaf:e1ab:336::/64,ib0!11\_0\_0\_0-255\_255\_0\_0|2001:db8:1:0::/64. The xCAT object definition commands support to use nicnetworks.<nicname> as the sub attributes.

## nicaliases

Comma-separated list of hostname aliases for each NIC.

**Format:** eth0!<alias list>,eth1!<alias1 list>|<alias2 list>

For multiple aliases per nic use a space-separated list.

For example: eth0!moe larry curly,eth1!tomljerry

## nicextraparams

Comma-separated list of extra parameters that will be used for each NIC configuration.

**If only one ip address is associated with each NIC:**

<nic1>|<param1=value1 param2=value2>,<nic2>|<param3=value3>, for example, eth0!MTU=1500,ib0!MTU=65520 CONNECTED\_MODE=yes.

**If multiple ip addresses are associated with each NIC:**

<nic1>!<param1=value1 param2=value2>|<param3=value3>,<nic2>!<param4=value4  
param5=value5>|<param6=value6>, for example, eth0!MTU=1500|MTU=1460,ib0!MTU=65520  
CONNECTED\_MODE=yes.

**The semicolon separator is needed if there are multiple values for extra parameters:**

bond0!BONDING\_OPTS=lacp\_rate=1;miimon=100;mode=802.3ad

The xCAT object definition commands support to use nicextraparams.<nicname> as the sub attributes.

**nicdevices**

Comma-separated list of NIC device per NIC, multiple ethernet devices can be bonded as bond device, these ethernet devices are separated by | . <nic1>!<dev1>|<dev3>,<nic2>!<dev2>, e.g. bond0!eth0|eth2,br0!bond0. The xCAT object definition commands support to use nicdevices.<nicname> as the sub attributes.

**nicsadapter**

Comma-separated list of NIC information collected by getadapter. <nic1>!<param1=value1  
param2=value2>,<nic2>!<param4=value4 param5=value5>, for example,  
enP3p3s0f1!mac=98:be:94:59:fa:cd linkstate=DOWN,enP3p3s0f2!mac=98:be:94:59:fa:ce candidate-  
name=enP3p3s0f2/enx98be9459face

**comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

**SEE ALSO**

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

**nimimage.5**

**NAME**

**nimimage** - a table in the xCAT database.

**SYNOPSIS**

**nimimage Attributes:** *imagename, nimtype, lpp\_source, spot, root, dump, paging, resolv\_conf, tmp, home, shared\_home, res\_group, nimmethod, script, bosinst\_data, installp\_bundle, mkysyb, fb\_script, shared\_root, otherpkgs, image\_data, configdump, comments, disable*

## DESCRIPTION

All the info that specifies a particular AIX operating system image that can be used to deploy AIX nodes.

### **nimimage Attributes:**

#### **imagename**

User provided name of this xCAT OS image definition.

#### **nimtype**

The NIM client type- standalone, diskless, or dataless.

#### **lpp\_source**

The name of the NIM lpp\_source resource.

#### **spot**

The name of the NIM SPOT resource.

#### **root**

The name of the NIM root resource.

#### **dump**

The name of the NIM dump resource.

#### **paging**

The name of the NIM paging resource.

#### **resolv\_conf**

The name of the NIM resolv\_conf resource.

#### **tmp**

The name of the NIM tmp resource.

#### **home**

The name of the NIM home resource.

#### **shared\_home**

The name of the NIM shared\_home resource.

#### **res\_group**

The name of a NIM resource group.

#### **nimmethod**

The NIM install method to use, (ex. rte, mkysb).

#### **script**

The name of a NIM script resource.

#### **bosinst\_data**

The name of a NIM bosinst\_data resource.

#### **installp\_bundle**

One or more comma separated NIM installp\_bundle resources.

**mksysb**

The name of a NIM mksysb resource.

**fb\_script**

The name of a NIM fb\_script resource.

**shared\_root**

A shared\_root resource represents a directory that can be used as a / (root) directory by one or more diskless clients.

**otherpkgs**

One or more comma separated installp or rpm packages. The rpm packages must have a prefix of 'R:', (ex. R:foo.rpm)

**image\_data**

The name of a NIM image\_data resource.

**configdump**

Specifies the type of system dump to be collected. The values are selective, full, and none. The default is selective.

**comments**

Any user-provided notes.

**disable**

Set to 'yes' or '1' to comment out this row.

**SEE ALSO**

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

**nodegroup.5**

**NAME**

**nodegroup** - a table in the xCAT database.

**SYNOPSIS**

**nodegroup Attributes:** *groupname, grouptype, members, membergroups, wherevals, comments, disable*

## DESCRIPTION

Contains group definitions, whose membership is dynamic depending on characteristics of the node.

### **nodegroup Attributes:**

#### **groupname**

Name of the group.

#### **groupype**

Static or Dynamic. A static group is defined to contain a specific set of cluster nodes. A dynamic node group is one that has its members determined by specifying a selection criteria for node attributes.

#### **members**

The value of the attribute is not used, but the attribute is necessary as a place holder for the object def commands. (The membership for static groups is stored in the nodelist table.)

#### **membergroups**

This attribute stores a comma-separated list of nodegroups that this nodegroup refers to. This attribute is only used by PCM.

#### **wherevals**

A list of “attr\*val” pairs that can be used to determine the members of a dynamic group, the delimiter is “::” and the operator \* can be ==, =~, != or !~.

#### **comments**

Any user-written notes.

#### **disable**

Set to ‘yes’ or ‘1’ to comment out this row.

## SEE ALSO

**nodels(1), chtag(8), tabdump(8), tabedit(8)**

### **nodehm.5**

## NAME

**nodehm** - a table in the xCAT database.

## SYNOPSIS

**nodehm Attributes:** *node, power, mgt, cons, termserver, termport, conserver, serialport, serialspeed, serialflow, getmac, cmdmapping, consoleondemand, consoleenabled, comments, disable*

## DESCRIPTION

Settings that control how each node's hardware is managed. Typically, an additional table that is specific to the hardware type of the node contains additional info. E.g. the ipmi, mp, and ppc tables.

### nodehm Attributes:

#### node

The node name or group name.

#### power

The method to use to control the power of the node. If not set, the mgt attribute will be used. Valid values: ipmi, blade, hmc, ivm, fsp, kvm, esx, rhevm. If "ipmi", xCAT will search for this node in the ipmi table for more info. If "blade", xCAT will search for this node in the mp table. If "hmc", "ivm", or "fsp", xCAT will search for this node in the ppc table.

#### mgt

The method to use to do general hardware management of the node. This attribute is used as the default if power or getmac is not set. Valid values: openbmc, ipmi, blade, hmc, ivm, fsp, bpa, kvm, esx, rhevm. See the power attribute for more details.

#### cons

The console method. If nodehm.serialport is set, this will default to the nodehm.mgt setting, otherwise it defaults to unused. Valid values: cyclades, mrv, or the values valid for the mgt attribute.

#### termserver

The hostname of the terminal server.

#### termport

The port number on the terminal server that this node is connected to.

#### conserver

The hostname of the machine where the conserver daemon is running. If not set, the default is the xCAT management node.

#### serialport

The serial port for this node, in the linux numbering style (0=COM1/ttyS0, 1=COM2/ttyS1). For SOL on IBM blades, this is typically 1. For rackmount IBM servers, this is typically 0.

#### serialspeed

The speed of the serial port for this node. For SOL this is typically 19200.

#### serialflow

The flow control value of the serial port for this node. For SOL this is typically 'hard'.

#### getmac

The method to use to get MAC address of the node with the `getmac` command. If not set, the `mgt` attribute will be used. Valid values: same as values for `mgmt` attribute.

**cmdmapping**

The fully qualified name of the file that stores the mapping between PCM hardware management commands and xCAT/third-party hardware management commands for a particular type of hardware device. Only used by PCM.

**consoleondemand**

This overrides the value from `site.consoleondemand`. Set to 'yes', 'no', '1' (equivalent to 'yes'), or '0' (equivalent to 'no'). If not set, the default is the value from `site.consoleondemand`.

**consoleenabled**

A flag field to indicate whether the node is registered in the console server. If '1', console is enabled, if not set, console is not enabled.

**comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

**SEE ALSO**

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

**nodelist.5****NAME**

**nodelist** - a table in the xCAT database.

**SYNOPSIS**

**nodelist Attributes:** *node, groups, status, statustime, appstatus, appstatustime, primarysn, hidden, updatestatus, updatestatustime, zonename, comments, disable*

**DESCRIPTION**

The list of all the nodes in the cluster, including each node's current status and what groups it is in.

**nodelist Attributes:****node**

The hostname of a node in the cluster.

**groups**

A comma-delimited list of groups this node is a member of. Group names are arbitrary, except all nodes should be part of the 'all' group. Internal group names are designated by using \_\_<groupname>. For example, \_\_Unmanaged, could be the internal name for a group of nodes that is not managed by xCAT. Admins should avoid using the \_\_ characters when defining their groups.

**status**

The current status of this node. This attribute will be set by xCAT software. Valid values: defined, booting, netbooting, booted, discovering, configuring, installing, alive, standingby, powering-off, unreachable. If blank, defined is assumed. The possible status change sequences are: For installation: defined->[discovering]->[configuring]->[standingby]->installing->booting->[postbooting]->booted->[alive], For diskless deployment: defined->[discovering]->[configuring]->[standingby]->netbooting->[postbooting]->booted->[alive], For booting: [alive/unreachable]->booting->[postbooting]->booted->[alive], For powering off: [alive]->powering-off->[unreachable], For monitoring: alive->unreachable. Discovering and configuring are for x Series discovery process. Alive and unreachable are set only when there is a monitoring plug-in start monitor the node status for xCAT. Note that the status values will not reflect the real node status if you change the state of the node from outside of xCAT (i.e. power off the node using HMC GUI).

**statustime**

The data and time when the status was updated.

**appstatus**

A comma-delimited list of application status. For example: 'sshd=up,ftp=down,ll=down'

**appstatustime**

The date and time when appstatus was updated.

**primarysn**

Not used currently. The primary servicenode, used by this node.

**hidden**

Used to hide fsp and bpa definitions, 1 means not show them when running lsdef and nodels

**updatestatus**

The current node update status. Valid states are synced, out-of-sync, syncing, failed.

**updatestatustime**

The date and time when the updatestatus was updated.

**zonename**

The name of the zone to which the node is currently assigned. If undefined, then it is not assigned to any zone.

**comments**

Any user-written notes.

**disable**



Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## nodepos.5

## NAME

**nodepos** - a table in the xCAT database.

## SYNOPSIS

**nodepos Attributes:** *node, rack, u, chassis, slot, room, height, comments, disable*

## DESCRIPTION

Contains info about the physical location of each node. Currently, this info is not used by xCAT, and therefore can be in whatever format you want. It will likely be used in xCAT in the future.

## nodepos Attributes:

### node

The node name or group name.

### rack

The frame the node is in.

### u

The vertical position of the node in the frame

### chassis

The BladeCenter chassis the blade is in.

### slot

The slot number of the blade in the chassis. For PCM, a comma-separated list of slot numbers is stored

### room

The room where the node is located.

### height

The server height in U(s).

### comments

Any user-written notes.

### disable

Set to 'yes' or '1' to comment out this row.

SEE ALSO

nodels(1), chtab(8), tabdump(8), tabedit(8)

noderes.5

NAME

**noderes** - a table in the xCAT database.

SYNOPSIS

**noderes Attributes:** *node, servicenode, netboot, tftpserver, tftpdn, nfsserver, monserver, nfsdir, installnic, primar-  
ynic, discoverynics, cmdinterface, xcatmaster, current\_osimage, next\_osimage, nimserver, routenames, nameservers,  
proxydhcp, syslog, comments, disable*

DESCRIPTION

Resources and settings to use when installing nodes.

noderes Attributes:

node

The node name or group name.

servicenode

A comma separated list of node names (as known by the management node) that provides most services for this node. The first service node on the list that is accessible will be used. The 2nd node on the list is generally considered to be the backup service node for this node when running commands like snmove.

netboot

The type of network booting to use for this node. Valid values:

Arch	OS	valid netboot options
x86, x86_64	ALL	pxe, xnba
ppc64	<=rhel6, <=sles11.3	yaboot
ppc64	>=rhels7, >=sles11.4	grub2,grub2-http,grub2-
↪ tftp		
ppc64le NonVirtualize	ALL	petitboot
ppc64le PowerKVM Guest	ALL	grub2,grub2-http,grub2-
↪ tftp		

tftpserver

The TFTP server for this node (as known by this node). If not set, it defaults to networks.tftpserver.

tftpdn

The directory that roots this nodes contents from a tftp and related perspective. Used for NAS offload by using different mountpoints.

**nfsserver**

The NFS or HTTP server for this node (as known by this node).

**monserver**

The monitoring aggregation point for this node. The format is “x,y” where x is the ip address as known by the management node and y is the ip address as known by the node.

**nfssdir**

The path that should be mounted from the NFS server.

**installnic**

The network adapter on the node that will be used for OS deployment, the installnic can be set to the network adapter name or the mac address or the keyword “mac” which means that the network interface specified by the mac address in the mac table will be used. If not set, primarynic will be used. If primarynic is not set too, the keyword “mac” will be used as default.

**primarynic**

This attribute will be deprecated. All the used network interface will be determined by installnic. The network adapter on the node that will be used for xCAT management, the primarynic can be set to the network adapter name or the mac address or the keyword “mac” which means that the network interface specified by the mac address in the mac table will be used. Default is eth0.

**discoverynics**

If specified, force discovery to occur on specific network adapters only, regardless of detected connectivity. Syntax can be simply “eth2,eth3” to restrict discovery to whatever happens to come up as eth2 and eth3, or by driver name such as “bnx2:0,bnx2:1” to specify the first two adapters managed by the bnx2 driver

**cmdinterface**

Not currently used.

**xcatmaster**

The hostname of the xCAT service node (as known by this node). This acts as the default value for nfsserver and tftpserver, if they are not set. If xcatmaster is not set, the node will use whoever responds to its boot request as its master. For the directed bootp case for POWER, it will use the management node if xcatmaster is not set.

**current\_osimage**

Not currently used. The name of the osimage data object that represents the OS image currently deployed on this node.

**next\_osimage**

Not currently used. The name of the osimage data object that represents the OS image that will be installed on the node the next time it is deployed.

**nimserver**

Not used for now. The NIM server for this node (as known by this node).

**routenames**

A comma separated list of route names that refer to rows in the routes table. These are the routes that should be defined on this node when it is deployed.

#### **nameservers**

An optional node/group specific override for name server list. Most people want to stick to site or network defined nameserver configuration.

#### **proxydhcp**

To specify whether the node supports proxydhcp protocol. Valid values: yes or 1, no or 0. Default value is yes.

#### **syslog**

To configure how to configure syslog for compute node. Valid values:blank(not set), ignore. blank - run postscript syslog; ignore - do NOT run postscript syslog

#### **comments**

Any user-written notes.

#### **disable**

Set to 'yes' or '1' to comment out this row.

### **SEE ALSO**

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

### **nodetype.5**

#### **NAME**

**nodetype** - a table in the xCAT database.

#### **SYNOPSIS**

**nodetype Attributes:** *node, os, arch, profile, provmethod, supportedarchs, nodetype, comments, disable*

#### **DESCRIPTION**

A few hardware and software characteristics of the nodes.

#### **nodetype Attributes:**

##### **node**

The node name or group name.

##### **os**

The operating system deployed on this node. Valid values: AIX, rhels\*,rhelc\*, rhas\*,centos\*, alma\*, rocky\*,SL\*, fedora\*, sles\* (where \* is the version #). As a special case, if this is set to "boottarget", then it will use the initrd/kernel/parameters specified in the row in the boottarget table in which boottarget.bprofile equals nodetype.profile.

##### **arch**

The hardware architecture of this node. Valid values: x86\_64, ppc64, x86, ia64.

#### **profile**

The string to use to locate a kickstart or autoyast template to use for OS deployment of this node. If the provmethod attribute is set to an osimage name, that takes precedence, and profile need not be defined. Otherwise, the os, profile, and arch are used to search for the files in /install/custom first, and then in /opt/xcat/share/xcat.

#### **provmethod**

The provisioning method for node deployment. The valid values are install, netboot, statelite or an os image name from the osimage table. If an image name is specified, the osimage definition stored in the osimage table and the linuximage table (for Linux) or nimimage table (for AIX) are used to locate the files for templates, pkglists, syncfiles, etc. On Linux, if install, netboot or statelite is specified, the os, profile, and arch are used to search for the files in /install/custom first, and then in /opt/xcat/share/xcat.

#### **supportedarchs**

Comma delimited list of architectures this node can execute.

#### **nodetype**

A comma-delimited list of characteristics of this node. Valid values: ppc, blade, vm (virtual machine), osi (OS image), mm, mn, rsa, switch.

#### **comments**

Any user-written notes.

#### **disable**

Set to 'yes' or '1' to comment out this row.

### **SEE ALSO**

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

### **notification.5**

#### **NAME**

**notification** - a table in the xCAT database.

#### **SYNOPSIS**

**notification Attributes:** *filename, tables, tableops, comments, disable*

## DESCRIPTION

Contains registrations to be notified when a table in the xCAT database changes. Users can add entries to have additional software notified of changes. Add and remove entries using the provided xCAT commands `regnotif` and `unregnotif`.

### notification Attributes:

#### **filename**

The path name of a file that implements the callback routine when the monitored table changes. Can be a perl module or a command. See the `regnotif` man page for details.

#### **tables**

Comma-separated list of xCAT database tables to monitor.

#### **tableops**

Specifies the table operation to monitor for. Valid values: “d” (rows deleted), “a” (rows added), “u” (rows updated).

#### **comments**

Any user-written notes.

#### **disable**

Set to ‘yes’ or ‘1’ to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## openbmc.5

## NAME

**openbmc** - a table in the xCAT database.

## SYNOPSIS

**openbmc Attributes:** *node, bmc, consport, taggedvlan, username, password, comments, disable*

## DESCRIPTION

Setting for nodes that are controlled by an on-board OpenBMC.

## openbmc Attributes:

### node

The node name or group name.

### bmc

The hostname of the BMC adapter.

### conspport

The console port for OpenBMC.

### taggedvlan

bmcsetup script will configure the network interface of the BMC to be tagged to the VLAN specified.

### username

The BMC userid. If not specified, the key=openbmc row in the passwd table is used as the default.

### password

The BMC password. If not specified, the key=openbmc row in the passwd table is used as the default.

### comments

Any user-written notes.

### disable

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## osdistro.5

## NAME

**osdistro** - a table in the xCAT database.

## SYNOPSIS

**osdistro Attributes:** *osdistraname, basename, majorversion, minorversion, arch, type, dirpaths, comments, disable*

## DESCRIPTION

Information about all the OS distros in the xCAT cluster

### **osdistro Attributes:**

#### **osdistraname**

Unique name (e.g. rhels6.2-x86\_64)

#### **basename**

The OS base name (e.g. rhels)

#### **majorversion**

The OS distro major version.(e.g. 6)

#### **minorversion**

The OS distro minor version. (e.g. 2)

#### **arch**

The OS distro arch (e.g. x86\_64)

#### **type**

Linux or AIX

#### **dirpaths**

Directory paths where OS distro is store. There could be multiple paths if OS distro has more than one ISO image. (e.g. /install/rhels6.2/x86\_64,...)

#### **comments**

Any user-written notes.

#### **disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1), chtag(8), tabdump(8), tabedit(8)**

### **osdistroupdate.5**

## NAME

**osdistroupdate** - a table in the xCAT database.



## SYNOPSIS

**osdistrouupdate Attributes:** *osupdatename, osdistrname, dirpath, downloadtime, comments, disable*

## DESCRIPTION

Information about the OS distro updates in the xCAT cluster

### osdistrouupdate Attributes:

#### osupdatename

Name of OS update. (e.g. rhn-update1)

#### osdistrname

The OS distro name to update. (e.g. rhels)

#### dirpath

Path to where OS distro update is stored. (e.g. /install/osdistrouupdates/rhels6.2-x86\_64-20120716-update)

#### downloadtime

The timestamp when OS distro update was downloaded..

#### comments

Any user-written notes.

#### disable

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**models(1), chtab(8), tabdump(8), tabedit(8)**

## osimage.5

## NAME

**osimage** - a table in the xCAT database.

## SYNOPSIS

**osimage Attributes:** *imagename, groups, profile, imagetype, description, provmethod, rootfstype, osdistrname, osupdatename, cfndir, osname, osvers, osarch, synclists, postscripts, postbootscripts, serverrole, isdeletable, kitcomponents, environvar, comments, disable*

## **DESCRIPTION**

Basic information about an operating system image that can be used to deploy cluster nodes.

### **osimage Attributes:**

#### **imagename**

The name of this xCAT OS image definition.

#### **groups**

A comma-delimited list of image groups of which this image is a member. Image groups can be used in the litelfile and litetree table instead of a single image name. Group names are arbitrary.

#### **profile**

The node usage category. For example compute, service.

#### **imagetype**

The type of operating system image this definition represents (linux,AIX).

#### **description**

OS Image Description

#### **provmethod**

The provisioning method for node deployment. The valid values are install, net-boot,statelite,boottarget,dualboot,sysclone. If boottarget is set, you must set linuximage.boottarget to the name of the boottarget definition. It is not used by AIX.

#### **rootfstype**

The filesystem type for the rootfs is used when the provmethod is statelite. The valid values are nfs or ramdisk. The default value is nfs

#### **osdistriname**

The name of the OS distro definition. This attribute can be used to specify which OS distro to use, instead of using the osname,osvers,and osarch attributes. For \*kit commands, the attribute will be used to read the osdistro table for the osname, osvers, and osarch attributes. If defined, the osname, osvers, and osarch attributes defined in the osimage table will be ignored.

#### **osupdatename**

A comma-separated list of OS distro updates to apply to this osimage.

#### **cfmdir**

CFM directory name for PCM. Set to /install/osimages/<osimage name>/cfmdir by PCM.

#### **osname**

Operating system name- AIX or Linux.

#### **osvers**

The Linux operating system deployed on this node. Valid values: rhels\*,rhelc\*, rhas\*,centos\*,alma\*, rocky\*,SL\*, fedora\*, sles\* (where \* is the version #).

#### **osarch**

The hardware architecture of this node. Valid values: x86\_64, ppc64, x86, ia64.

### **synclists**

The fully qualified name of a file containing a list of files to synchronize on the nodes. Can be a comma separated list of multiple synclist files. The synclist generated by PCM named `/install/osimages/<imagename>/synclist.cfm` is reserved for use only by PCM and should not be edited by the admin.

### **postscripts**

Comma separated list of scripts that should be run on this image after diskful installation or diskless boot. For installation of RedHat, CentOS, Fedora, the scripts will be run before the reboot. For installation of SLES, the scripts will be run after the reboot but before the `init.d` process. For diskless deployment, the scripts will be run at the `init.d` time, and xCAT will automatically add the list of scripts from the `postbootscripts` attribute to run after `postscripts` list. For installation of AIX, the scripts will run after the reboot and acts the same as the `postbootscripts` attribute. For AIX, use the `postbootscripts` attribute. See the site table `runbootscripts` attribute.

### **postbootscripts**

Comma separated list of scripts that should be run on this after diskful installation or diskless boot. On AIX these scripts are run during the processing of `/etc/inittab`. On Linux they are run at the `init.d` time. xCAT automatically adds the scripts in the `xcatdefaults.postbootscripts` attribute to run first in the list. See the site table `runbootscripts` attribute.

### **serverrole**

The role of the server created by this osimage. Default roles: `mgtnode`, `servicenode`, `compute`, `login`, `storage`, `utility`.

### **isdeletable**

A flag to indicate whether this image profile can be deleted. This attribute is only used by PCM.

### **kitcomponents**

List of Kit Component IDs assigned to this OS Image definition.

### **environvar**

Comma delimited environment variables for the osimage

### **comments**

Any user-written notes.

### **disable**

Set to 'yes' or '1' to comment out this row.

## **SEE ALSO**

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## passwd.5

### NAME

**passwd** - a table in the xCAT database.

### SYNOPSIS

**passwd Attributes:** *key, username, password, cryptmethod, authdomain, comments, disable*

### DESCRIPTION

Contains default userids and passwords for xCAT to access cluster components. In most cases, xCAT will also actually set the userid/password in the relevant component when it is being configured or installed. Userids/passwords for specific cluster components can be overridden in other tables, e.g. mpa, ipmi, ppchcp, etc.

#### passwd Attributes:

##### key

The type of component this user/pw is for. Valid values: blade (management module), ipmi (BMC), system (nodes), omapi (DHCP), hmc, ivm, cec, frame, switch.

##### username

The default userid for this type of component

##### password

The default password for this type of component. On Linux, a crypted form could be provided for the “system” component, which will be used during initial node provisioning. Hashes starting with \$1\$, \$5\$ and \$6\$ (md5, sha256 and sha512 respectively) are supported.

##### cryptmethod

Indicates the method to use to encrypt the password attribute. On AIX systems, if a value is provided for this attribute it indicates that the password attribute is encrypted. If the cryptmethod value is not set it indicates the password is a simple string value. On Linux systems, the cryptmethod can be set to md5, sha256 or sha512. If not set, sha256 will be used as default to encrypt plain-text passwords.

##### authdomain

The domain in which this entry has meaning, e.g. specifying different domain administrators per active directory domain

##### comments

Any user-written notes.

##### disable

Set to ‘yes’ or ‘1’ to comment out this row.

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## pdu.5

## NAME

**pdu** - a table in the xCAT database.

## SYNOPSIS

**pdu Attributes:** *node, nodetype, pdutype, outlet, username, password, snmpversion, community, snmpuser, authtype, authkey, privtype, privkey, seclevel, comments, disable*

## DESCRIPTION

Parameters to use when interrogating pdus

### pdu Attributes:

#### **node**

The hostname/address of the pdu to which the settings apply

#### **nodetype**

The node type should be pdu

#### **pdutype**

The type of pdu

#### **outlet**

The pdu outlet count

#### **username**

The remote login user name

#### **password**

The remote login password

#### **snmpversion**

The version to use to communicate with switch. SNMPv1 is assumed by default.

#### **community**

The community string to use for SNMPv1/v2

#### **snmpuser**

The username to use for SNMPv3 communication, ignored for SNMPv1

#### **authtype**

The authentication protocol(MD5|SHA) to use for SNMPv3.

**authkey**

The authentication passphrase for SNMPv3

**privtype**

The privacy protocol(AES|DES) to use for SNMPv3.

**privkey**

The privacy passphrase to use for SNMPv3.

**seclevel**

The Security Level(noAuthNoPriv|authNoPriv|authPriv) to use for SNMPv3.

**comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

**SEE ALSO**

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

**pduoutlet.5**

**NAME**

**pduoutlet** - a table in the xCAT database.

**SYNOPSIS**

**pduoutlet Attributes:** *node, pdu, comments, disable*

**DESCRIPTION**

Contains list of outlet numbers on the pdu each node is connected to.

**pduoutlet Attributes:**

**node**

The node name or group name.

**pdu**

a comma-separated list of outlet number for each PDU, ex: pdu1:outlet1,pdu2:outlet1

**comments**

Any user-written notes.

## disable

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## performance.5

## NAME

**performance** - a table in the xCAT database.

## SYNOPSIS

**performance Attributes:** *timestamp, node, attrname, attrvalue, comments, disable*

## DESCRIPTION

Describes the system performance every interval unit of time.

## performance Attributes:

### timestamp

The time at which the metric was captured.

### node

The node name.

### attrname

The metric name.

### attrvalue

The metric value.

### comments

Any user-provided notes.

### disable

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## policy.5

### NAME

**policy** - a table in the xCAT database.

### SYNOPSIS

**policy Attributes:** *priority, name, host, commands, noderange, parameters, time, rule, comments, disable*

### DESCRIPTION

The policy table in the xCAT database controls who has authority to run specific xCAT operations. It is basically the Access Control List (ACL) for xCAT. It is sorted on the priority field before evaluating.

#### policy Attributes:

##### priority

The priority value for this rule. This value is used to identify this policy data object (i.e. this rule) The table is sorted on this field with the lower the number the higher the priority. For example 1.0 is higher priority than 4.1 is higher than 4.9.

##### name

The username that is allowed to perform the commands specified by this rule. Default is "\*" (all users).

##### host

The host from which users may issue the commands specified by this rule. Default is "\*" (all hosts). Only all or one host is supported

##### commands

The list of commands that this rule applies to. Default is "\*" (all commands).

##### noderange

The Noderange that this rule applies to. Default is "\*" (all nodes). Not supported with the \*def commands.

##### parameters

A regular expression that matches the command parameters (everything except the noderange) that this rule applies to. Default is "\*" (all parameters). Not supported with the \*def commands.

##### time

Time ranges that this command may be executed in. This is not supported.

##### rule



Specifies how this rule should be applied. Valid values are: allow, trusted. Allow will allow the user to run the commands. Any other value will deny the user access to the commands. Trusted means that once this client has been authenticated via the certificate, all other information that is sent (e.g. the username) is believed without question. This authorization should only be given to the xcatd on the management node at this time.

#### comments

Any user-written notes.

#### disable

Set to 'yes' or '1' to comment out this row.

### SEE ALSO

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

### postscripts.5

#### NAME

**postscripts** - a table in the xCAT database.

### SYNOPSIS

**postscripts Attributes:** *node, postscripts, postbootscripts, comments, disable*

### DESCRIPTION

The scripts that should be run on each node after installation or diskless boot.

#### postscripts Attributes:

##### node

The node name or group name.

##### postscripts

Comma separated list of scripts that should be run on this node after diskful installation or diskless boot. Each script can take zero or more parameters. For example: "script1 p1 p2,script2,...". xCAT automatically adds the postscripts from the xcatdefaults.postscripts attribute of the table to run first on the nodes after install or diskless boot. For installation of RedHat, CentOS, Fedora, the scripts will be run before the reboot. For installation of SLES, the scripts will be run after the reboot but before the init.d process. For diskless deployment, the scripts will be run at the init.d time, and xCAT will automatically add the list of scripts from the postbootscripts attribute to run after postscripts list. For installation of AIX, the scripts will run after the reboot and acts the same as the postbootscripts attribute. For AIX, use the postbootscripts attribute. Please note that the postscripts specified for "xcatdefaults" will be assigned to node automatically, they can not be removed from "postscripts" attribute of a node with "chdef -m" command

##### postbootscripts

Comma separated list of scripts that should be run on this node after diskful installation or diskless boot. Each script can take zero or more parameters. For example: “script1 p1 p2,script2,...”. On AIX these scripts are run during the processing of /etc/inittab. On Linux they are run at the init.d time. xCAT automatically adds the scripts in the xcatdefaults.postbootscripts attribute to run first in the list. Please note that the postbootscripts specified for “xcatdefaults” will be assigned to node automatically, they can not be removed from “postbootscripts” attribute of a node with “chdef -m” command

**comments**

Any user-written notes.

**disable**

Set to ‘yes’ or ‘1’ to comment out this row.

**SEE ALSO**

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

**ppc.5****NAME**

**ppc** - a table in the xCAT database.

**SYNOPSIS**

**ppc Attributes:** *node, hcp, id, pprofile, parent, nodetype, supernode, sfp, comments, disable*

**DESCRIPTION**

List of system p hardware: HMCs, IVMs, FSPs, BPCs, CECs, Frames.

**ppc Attributes:****node**

The node name or group name.

**hcp**

The hardware control point for this node (HMC, IVM, Frame or CEC). Do not need to set for BPAs and FSPs.

**id**

For LPARs: the LPAR numeric id; for CECs: the cage number; for Frames: the frame number.

**pprofile**

The LPAR profile that will be used the next time the LPAR is powered on with rpower. For DFM, the pprofile attribute should be set to blank

**parent**

For LPARs: the CEC; for FSPs: the CEC; for CEC: the frame (if one exists); for BPA: the frame; for frame: the building block number (which consists 1 or more service nodes and compute/storage nodes that are serviced by them - optional).

#### **nodetype**

The hardware type of the node. Only can be one of fsp, bpa, cec, frame, ivm, hmc and lpar

#### **supernode**

Indicates the connectivity of this CEC in the HFI network. A comma separated list of 2 ids. The first one is the supernode number the CEC is part of. The second one is the logical location number (0-3) of this CEC within the supernode.

#### **sfp**

The Service Focal Point of this Frame. This is the name of the HMC that is responsible for collecting hardware service events for this frame and all of the CECs within this frame.

#### **comments**

Any user-written notes.

#### **disable**

Set to 'yes' or '1' to comment out this row.

### **SEE ALSO**

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

### **ppcdirect.5**

#### **NAME**

**ppcdirect** - a table in the xCAT database.

#### **SYNOPSIS**

**ppcdirect Attributes:** *hcp, username, password, comments, disable*

#### **DESCRIPTION**

Info necessary to use FSPs/BPAs to control system p CECs/Frames.

### ppcdirect Attributes:

#### hcp

Hostname of the FSPs/BPAs(for ASMI) and CECs/Frames(for DFM).

#### username

Userid of the FSP/BPA(for ASMI) and CEC/Frame(for DFM). If not filled in, xCAT will look in the passwd table for key=fsp. If not in the passwd table, the default used is admin.

#### password

Password of the FSP/BPA(for ASMI) and CEC/Frame(for DFM). If not filled in, xCAT will look in the passwd table for key=fsp. If not in the passwd table, the default used is admin.

#### comments

Any user-written notes.

#### disable

Set to 'yes' or '1' to comment out this row.

### SEE ALSO

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

### ppchcp.5

#### NAME

**ppchcp** - a table in the xCAT database.

#### SYNOPSIS

**ppchcp Attributes:** *hcp, username, password, comments, disable*

#### DESCRIPTION

Info necessary to use HMCs and IVMs as hardware control points for LPARs.

### ppchcp Attributes:

#### hcp

Hostname of the HMC or IVM.

#### username

Userid of the HMC or IVM. If not filled in, xCAT will look in the passwd table for key=hmc or key=ivm. If not in the passwd table, the default used is hscroot for HMCs and padmin for IVMs.

#### password

Password of the HMC or IVM. If not filled in, xCAT will look in the passwd table for key=hmc or key=ivm. If not in the passwd table, the default used is abc123 for HMCs and padmin for IVMs.

#### comments

Any user-written notes.

#### disable

Set to 'yes' or '1' to comment out this row.

### SEE ALSO

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

#### prescripts.5

### NAME

**prescripts** - a table in the xCAT database.

### SYNOPSIS

**prescripts Attributes:** *node, begin, end, comments, disable*

### DESCRIPTION

The scripts that will be run at the beginning and the end of the nodeset(Linux), nimnodeset(AIX) or mkdsklnode(AIX) command.

#### prescripts Attributes:

##### node

The node name or group name.

##### begin

**The scripts to be run at the beginning of the nodeset(Linux), nimnodeset(AIX) or mkdsklnode(AIX) command.**

##### The format is:

```
[action1:]s1,s2...[| action2:s3,s4,s5...]
```

##### where:

- action1 and action2 for Linux are the nodeset actions specified in the command. For AIX, action1 and action1 can be 'diskless' for mkdsklnode command' and 'standalone for nimnodeset command.
- s1 and s2 are the scripts to run for action1 in order.
- s3, s4, and s5 are the scripts to run for actions2.

If actions are omitted, the scripts apply to all actions. Examples:

```
myscript1,myscript2 (all actions) diskless:myscript1,myscript2 (AIX) in-  
stall:myscript1,myscript2|netboot:myscript3 (Linux)
```

All the scripts should be copied to /install/prescripts directory. The following two environment variables will be passed to each script:

**NODES** a coma separated list of node names that need to run the script for ACTION current nodeset action.

If '#xCAT setting:MAX\_INSTANCE=number' is specified in the script, the script will get invoked for each node in parallel, but no more than number of instances will be invoked at a time. If it is not specified, the script will be invoked once for all the nodes.

#### **end**

The scripts to be run at the end of the nodeset(Linux), nimnodeset(AIX),or mkdsklsnode(AIX) command. The format is the same as the 'begin' column.

#### **comments**

Any user-written notes.

#### **disable**

Set to 'yes' or '1' to comment out this row.

### **SEE ALSO**

**nodes(1), chtab(8), tabdump(8), tabedit(8)**

#### **prodkey.5**

#### **NAME**

**prodkey** - a table in the xCAT database.

#### **SYNOPSIS**

**prodkey Attributes:** *node, product, key, comments, disable*

#### **DESCRIPTION**

Specify product keys for products that require them

### prodkey Attributes:

#### node

The node name or group name.

#### product

A string to identify the product (for OSes, the osname would be used, i.e. wink28

#### key

The product key relevant to the aforementioned node/group and product combination

#### comments

Any user-written notes.

#### disable

Set to 'yes' or '1' to comment out this row.

### SEE ALSO

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

### rack.5

### NAME

**rack** - a table in the xCAT database.

### SYNOPSIS

**rack Attributes:** *rackname, displayname, num, height, room, comments, disable*

### DESCRIPTION

Rack information.

### rack Attributes:

#### rackname

The rack name.

#### displayname

Alternative name for rack. Only used by PCM.

#### num

The rack number.

#### height

Number of units which can be stored in the rack.

**room**

The room in which the rack is located.

**comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

**SEE ALSO**

**nodels(1), chtag(8), tabdump(8), tabedit(8)**

**routes.5**

**NAME**

**routes** - a table in the xCAT database.

**SYNOPSIS**

**routes Attributes:** *routename, net, mask, gateway, ifname, comments, disable*

**DESCRIPTION**

Describes the additional routes needed to be setup in the os routing table. These routes usually are used to connect the management node to the compute node using the service node as gateway.

**routes Attributes:**

**routename**

Name used to identify this route.

**net**

The network address.

**mask**

The network mask.

**gateway**

The gateway that routes the ip traffic from the mn to the nodes. It is usually a service node.

**ifname**

The interface name that facing the gateway. It is optional for IPv4 routes, but it is required for IPv6 routes.

**comments**



Any user-written notes.

## **disable**

Set to 'yes' or '1' to comment out this row.

## **SEE ALSO**

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## **servicenode.5**

## **NAME**

**servicenode** - a table in the xCAT database.

## **SYNOPSIS**

**servicenode Attributes:** *node, nameserver, dhcpserver, tftpserver, nfsserver, conserver, monserver, ldapserver, ntpserver, fipsserver, nimserver, ipforward, dhcpinterfaces, proxydhcp, comments, disable*

## **DESCRIPTION**

List of all Service Nodes and services that will be set up on the Service Node.

### **servicenode Attributes:**

#### **node**

The hostname of the service node as known by the Management Node.

#### **nameserver**

Do we set up DNS on this service node? Valid values: 2, 1, or 0. If 2, creates named.conf as dns slave, using the management node as dns master, and starts named. If 1, creates named.conf file with forwarding to the management node and starts named. If 0, it does not change the current state of the service.

#### **dhcpserver**

Do we set up DHCP on this service node? Not supported on AIX. Valid values:1 or 0. If 1, runs makedhcp -n. If 0, it does not change the current state of the service.

#### **tftpserver**

Do we set up TFTP on this service node? Not supported on AIX. Valid values:1 or 0. If 1, configures and starts atftp. If 0, it does not change the current state of the service.

#### **nfsserver**

Do we set up file services (HTTP,FTP,or NFS) on this service node? For AIX will only setup NFS, not HTTP or FTP. Valid values:1 or 0.If 0, it does not change the current state of the service.

#### **conserver**

Do we set up console service on this service node? Valid values: 0, 1, or 2. If 0, it does not change the current state of the service. If 1, configures and starts conserver daemon. If 2, configures and starts goconserver daemon.

**monserver**

Is this a monitoring event collection point? Valid values: 1 or 0. If 0, it does not change the current state of the service.

**ldapservice**

Do we set up ldap caching proxy on this service node? Not supported on AIX. Valid values:1 or 0. If 0, it does not change the current state of the service.

**ntpserver**

Not used. Use setupntp postscript to setup a ntp server on this service node? Valid values:1 or 0. If 0, it does not change the current state of the service.

**ftpservice**

Do we set up a ftp server on this service node? Not supported on AIX Valid values:1 or 0. If 1, configure and start vsftpd. (You must manually install vsftpd on the service nodes before this.) If 0, it does not change the current state of the service. xCAT is not using ftp for compute nodes provisioning or any other xCAT features, so this attribute can be set to 0 if the ftp service will not be used for other purposes

**nimserver**

Not used. Do we set up a NIM server on this service node? Valid values:1 or 0. If 0, it does not change the current state of the service.

**ipforward**

Do we set up ip forwarding on this service node? Valid values:1 or 0. If 0, it does not change the current state of the service.

**dhcpinterfaces**

The network interfaces DHCP server should listen on for the target node. This attribute can be used for management node and service nodes. If defined, it will override the values defined in site.dhcpinterfaces. This is a comma separated list of device names. !remote! indicates a non-local network for relay DHCP. For example: !remote!,eth0,eth1

**proxydhcp**

Do we set up proxydhcp service on this node? valid values: 1 or 0. If 1, the proxydhcp daemon will be enabled on this node.

**comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

**site.5**

## NAME

**site** - a table in the xCAT database.

## SYNOPSIS

**site Attributes:** *key, value, comments, disable*

## DESCRIPTION

Global settings for the whole cluster. This table is different from the other tables in that each attribute is just named in the key column, rather than having a separate column for each attribute. The following is a list of attributes currently used by xCAT organized into categories.

### site Attributes:

#### key

Attribute Name: Description

#### AIX ATTRIBUTES

```
nimprime :   The name of NIM server, if not set default is the AIX MN.
              If Linux MN, then must be set for support of mixed cluster (TBD).

useSSHonAIX: (yes/1 or no/0). Default is yes. The support for rsh/rcp is
↳ deprecated.

useNFSv4onAIX: (yes/1 or no/0). If yes, NFSv4 will be used with NIM. If no,
              NFSv3 will be used with NIM. Default is no.
```

#### DATABASE ATTRIBUTES

```
auditnosyslog: If set to 1, then commands will only be written to the
↳ auditlog table.
              If this attribute is set to 1 and auditskipcmds=ALL means no
↳ logging of commands.
              Default is to write to both the auditlog table and syslog.

auditskipcmds: List of commands and/or client types that will not be
              written to the auditlog table and syslog. See auditnosyslog.
              'ALL' means all cmds will be skipped. If attribute is null, all
              commands will be written.
```

(continues on next page)

(continued from previous page)

```

clienttype:web would skip all commands from the web client
For example: tabdump,nodeids,clienttype:web
will not log tabdump, nodeids or any web client commands.

databaseloc:  Directory where we create the db instance directory.
               Default is /var/lib. Only DB2 is currently supported.
               Do not use the directory in the site.installloc or
               installldir attribute. This attribute must not be changed
               once db2sqlsetup script has been run and DB2 has been setup.

excludenodes: A set of comma separated nodes and/or groups that would
↳automatically
               be subtracted from any noderange, it can be used for excluding
↳some
               failed nodes for any xCAT commands. See the 'noderange'
↳manpage for
               details on supported formats.

nodestatus:  If set to 'n', the nodelist.status column will not be updated
↳during
               the node deployment, node discovery and power operations. The
↳default
               is to update.

skiptables:  Comma separated list of tables to be skipped by dumpxCATdb

skipvalidate: If set to 1, then getcredentials and getpostscrips calls
↳will not
               be logged in syslog.

-----
DHCP ATTRIBUTES
-----
dhcpiinterfaces: The network interfaces DHCP should listen on. If it is the
↳same for all
               nodes, use a comma-separated list of the NICs. To specify
↳different NICs
               for different nodes, use the format: "xcatmn|eth1,eth2;
↳service|bond0",
               where xcatmn is the name of the management node, DHCP should
↳listen on
               the eth1 and eth2 interfaces. All the nodes in group
↳'service' should
               listen on the 'bond0' interface.

               To disable the genesis kernel from being sent to specific
↳interfaces, a
               ':noboot' option can be appended to the interface name. For
↳example,
               if the management node has two interfaces, eth1 and eth2,
↳disable
               genesis from being sent to eth1 using: "eth1:noboot,eth2".

```

(continues on next page)

(continued from previous page)

dhcsetup: If set to 'n', it will skip the dhcp setup process in the nodeset.  
cmd.

dhcplease: The lease time for the dhcp client. The default value is 43200.

disjointdhcps: If set to '1', the .leases file on a service node only  
contains  
the nodes it manages. And when 'sharedtftp' is disabled,  
nodeset handles  
boot loader configuration on a service node only for the  
nodes it manages.  
The default value is '0'. It means include all the nodes in  
the subnet.

pruneservices: Whether to enable service pruning when noderm is run (i.e.  
removing DHCP entries when noderm is executed)

managedaddressmode: The mode of networking configuration during node  
provision.  
If set to 'static', the network configuration will be  
configured  
in static mode based on the node and network definition  
on MN.  
If set to 'dhcp', the network will be configured with  
dhcp protocol.  
The default is 'dhcp'.

#### DNS ATTRIBUTES

dnshandler: Name of plugin that handles DNS setup for makedns.

domain: The DNS domain name used for the cluster.

forwarders: The DNS servers at your site that can provide names outside of  
the cluster.  
The makedns command will configure the DNS on the management  
node to forward  
requests it does not know to these servers. Note that the DNS  
servers on the  
service nodes will ignore this value and always be configured to  
forward  
to the management node.

dnsforwardmode: (first or only or no). This is to set forward value in named.  
conf options section.  
"first": causes DNS requests to be forwarded before an attempt  
is made to resolve them via the root name servers.  
"only": all requests are forwarded and none sent to the root  
name servers.  
"no": no request will be forwarded. This is the default value if  
not specified.

(continues on next page)

(continued from previous page)

emptyzonesenable: (yes **or** no). This is to set empty-zones-enable value in `named.conf` options section.

master: The hostname of the xCAT management node, as known by the nodes.

nameservers: A comma delimited list of DNS servers that **each** node in the `cluster` should **use**. This value will end up in the nameserver settings of the `/etc/resolv.conf` on **each** node. It is common (but **not** required) **to set** this attribute value to the IP addr of the xCAT management node, **if** you have set up the DNS on the management node by running `makedns`. In a hierarchical cluster, you can also set this attribute to "`<xcatmaster>`" to mean the DNS server **for each** node should be **the** node that is managing it (either its service node **or** the **management** node).

externaldns: To specify that external dns is used. If externaldns is set to **any** value **then**, `makedns` command will **not** start the **local** nameserver on **xCAT MN**. Default is to start the **local** nameserver.

dnsupdaters: The value are `' '` separated string which will be added to the `zone config` section. This is an interface **for** user to add configuration **entries** to the zone sections in `named.conf`.

dnsinterfaces: The network interfaces DNS should **listen** on. If it is the **same** **for** all nodes, **use** a simple comma-separated list of NICs. To specify **different** NICs **for** different nodes, **use the format**: "`xcatmn|eth1,eth2;service|bond0`", where `xcatmn` is the name of the management node, **and** DNS **should listen** on the `eth1` **and** `eth2` interfaces. All the nodes in group `'service'` should **listen** on the `'bond0'` interface.

NOTE: If using this attribute to block certain interfaces, **make sure** the IP maps to your hostname of xCAT MN is **not** blocked since **xCAT needs** to **use this** IP to communicate with the **local** NDS server on MN.

(continues on next page)

(continued from previous page)

```

namedincludes: A comma-separated list of file paths that will be included in
                named.conf, using named 'include' statement. This can allow
↳users
                to include local configuration options that will not be
↳overwritten
                by 'makedns -n'

-----
HARDWARE CONTROL ATTRIBUTES
-----

blademaxp: The maximum number of concurrent processes for blade hardware
↳control.

ea_primary_hmc: The hostname of the HMC that the Integrated Switch Network
                Management Event Analysis should send hardware serviceable
                events to for processing and potentially sending to IBM.

ea_backup_hmc: The hostname of the HMC that the Integrated Switch Network
                Management Event Analysis should send hardware serviceable
                events to if the primary HMC is down.

enableASMI: (yes/1 or no/0). If yes, ASMI method will be used after fsp-api.
↳If no,
                when fsp-api is used, ASMI method will not be used. Default is
↳no.

fsptimeout: The timeout, in milliseconds, to use when communicating with
↳FSPs.

hwctrldispatch: Whether or not to send hw control operations to the service
                node of the target nodes. Default is 'y'.(At present, this
↳attribute
                is only used for IBM Flex System)

ipmidispatch: Whether or not to send ipmi hw control operations to the
↳service
                node of the target compute nodes. Default is 'y'.

ipmimaxp: The max # of processes for ipmi hw ctrl. The default is 64.
↳Currently,
                this is only used for HP hw control.

ipmiretries: The # of retries to use when communicating with BMCs. Default
↳is 3.

ipmisdrccache: If set to 'no', then the xCAT IPMI support will not cache
↳locally
                the target node's SDR cache to improve performance.

ipmitimeout: The timeout to use when communicating with BMCs. Default is 2.
                This attribute is currently not used.

```

(continues on next page)

(continued from previous page)

maxssh: The max # of SSH connections at any one time to the hw ctrl point.  
↳for PPC  
This parameter doesn't take effect on the rpower command.  
It takes effects on other PPC hardware control command  
getmacs/rnetboot/rbootseq and so on. Default is 8.

syspowerinterval: For SystemP CECs, this is the number of seconds the rpower.  
↳command  
will wait between performing the action for each CEC. For  
↳SystemX  
IPMI servers, this is the number of seconds the rpower.  
↳command will  
wait between powering on <syspowermaxnodes> nodes at a  
↳time. This  
value is used to control the power on speed in large.  
↳clusters.  
Default is 0.

syspowermaxnodes: The number of servers to power on at one time before.  
↳waiting  
'syspowerinterval' seconds to continue on to the next set.  
↳of  
nodes. If the noderange given to rpower includes nodes.  
↳served  
by different service nodes, it will try to spread each set.  
↳of  
nodes across the service nodes evenly. Currently only used.  
↳for  
IPMI servers and must be set if 'syspowerinterval' is set.

powerinterval: The number of seconds the rpower command to LPARs will wait.  
↳between  
performing the action for each LPAR. LPARs of different HCPs  
(HMCs or FSPs) are done in parallel. This is used to limit the  
cluster boot up speed in large clusters. Default is 0. This  
↳is  
currently only used for system p hardware.

ppcmaxp: The max # of processes for PPC hw ctrl. If there are more than.  
↳ppcmaxp  
hcps, this parameter will take effect. It will control the max.  
↳number of  
processes for PPC hardware control commands. Default is 64.

ppcretry: The max # of PPC hw connection attempts to HMC before failing.  
It only takes effect on the hardware control commands through HMC.  
Default is 3.

ppctimeout: The timeout, in milliseconds, to use when communicating with PPC.  
↳hw  
through HMC. It only takes effect on the hardware control.  
↳commands

(continues on next page)



(continued from previous page)

```

        through HMC. Default is 0.

snmpc: The snmp community string that xcat should use when communicating
↳with the
        switches.

-----
INSTALL/DEPLOYMENT ATTRIBUTES
-----

cleanupxcatpost: (yes/1 or no/0). Set to 'yes' or '1' to clean up the /
↳xcatpost
                directory on the stateless and statelite nodes after the
                postscripts are run. Default is no.

cleanupdiskfullxcatpost: (yes/1 or no/0). Set to 'yes' or '1' to clean up
↳the /xcatpost
                directory on the diskfull nodes after the
                postscripts are run with no errors. Default is no.

db2installloc: The location which the service nodes should mount for
↳hostname is
                the db2 code to install. Format is hostname:/path. If
↳mntdb2.
                omitted, it defaults to the management node. Default is /

defserialflow: The default serial flow - currently only used by the mknb
↳command.

defserialport: The default serial port - currently only used by mknb.

defserialspeed: The default serial speed - currently only used by mknb.

disablenodesetwarning: Allow the legacy xCAT non-osimage style nodeset to
↳execute.

genmacprefix: When generating mac addresses automatically, use this
↳manufacturing
                prefix (e.g. 00:11:aa)

genpasswords: Automatically generate random passwords for BMCs when
↳configuring
                them.

installdir: The local directory name used to hold the node deployment
↳packages(obsoleted).

installloc: The location from which the service nodes should mount the
                deployment packages in the format hostname:/path. If hostname is
                omitted, it defaults to the management node. The path must
                match the path in the installdir attribute.

iscsidir: The path to put the iscsi disks in on the mgmt node.

```

(continues on next page)

(continued from previous page)

`mnroutenames`: The name of the routes to be setup on the management node.  
It is a comma separated list of route names that are `defined`  
→in the routes table.

`runbootscripts`: If set to `'yes'` the scripts listed in the `postbootscripts` attribute in the `osimage` and `postscripts` tables will be run  
→during each reboot of stateful (diskful) nodes. This attribute has  
→no effect on stateless and statelite nodes. Run the following command after you change the value of this attribute:  
`'updatenode <nodes> -P setuppostbootscripts'`

`precreatemybootscripts`: (yes/1 or no/0). Default is `no`. If yes, it will instruct xCAT at nodeset and updatenode time to query the db  
→once for all of the nodes passed into the cmd and create the mybootscript  
→file for each node, and put them in a directory of `tftpdn`(such as: /  
→tftpdn) If no, it will not generate the mybootscript file in the `tftpdn`.

`secureroot`: If set to 1, xCAT will use secure mode to transfer root password  
→hash during the installation. Default is 0.

`setinstallnic`: Set the network configuration for installnic to be static.

`sharedtftp`: Set to 0 or no, xCAT should not assume the directory in `tftpdn` is mounted on all on Service Nodes. Default is 1/yes. If value is set to a hostname, the directory in `tftpdn` will be mounted from that hostname on the SN

`sharedinstall`: Indicates if a shared file system will be used for  
→installation resources. Possible values are: `'no'`, `'sns'`, or `'all'`. `'no'` means a shared file system is not being used. `'sns'` means a shared filesystem is being used across all service nodes. `'all'` means that the management as well as the service nodes are all using a common shared filesystem. The default is `'no'`.

`xcatconfdir`: Where xCAT config data is (default `/etc/xcat`).

`xcatdebugmode`: the xCAT debug level. xCAT provides a batch of techniques to help user debug problems while using xCAT, especially on  
→OS provision, such as collecting logs of the whole installation process and  
→accessing the installing system via ssh, etc. These techniques will be  
→enabled

(continues on next page)

(continued from previous page)

according to different xCAT debug levels specified by  
 ↪ 'xcatdebugmode',  
 currently supported values:  
     '0': disable debug mode  
     '1': enable basic debug mode  
     '2': enable expert debug mode  
 For the details on 'basic debug mode' and 'expert debug mode',  
 refer to xCAT documentation.

#### ----- REMOTESHELL ATTRIBUTES -----

nodesyncfiledir: The directory on the node, where xdcp will rsync the files  
 SNsyncfiledir: The directory on the Service Node, where xdcp will rsync the ↪  
 ↪ files  
                   from the MN that will eventually be rsync'd to the compute ↪  
 ↪ nodes.  
 sshbetweennodes: Comma separated list of groups of compute nodes to enable ↪  
 ↪ passwordless  
                   root ssh to the nodes during install or running 'xdsh -K'. ↪  
 ↪ The default  
                   is ALLGROUPS. Set to NOGROUPS to disable.  
                   Service Nodes are not affected by this attribute as they ↪  
 ↪ are always  
                   configured with passwordless root access.  
                   If using the zone table, this attribute is not used.

#### ----- SERVICES ATTRIBUTES -----

consoleondemand: When set to 'yes', conserver connects and creates the ↪  
 ↪ console  
                   output only when the user opens the console. Default is 'no  
 ↪ ' on  
                   Linux, 'yes' on AIX.  
 httpport: The port number that the booting/installing nodes should contact ↪  
 ↪ the  
                   http server on the MN/SN on. It is your responsibility to ↪  
 ↪ configure  
                   the http server to listen on that port - xCAT will not do that.  
 nmapoptions: Additional options for the nmap command. nmap is used in pping,  
                   nodestat, xdsh -v and updatenode commands. Sometimes additional  
                   performance tuning may be needed for nmap due to network traffic.  
                   For example, if the network response time is too slow, nmap may ↪  
 ↪ not  
                   give stable output. You can increase the timeout value by ↪  
 ↪ specifying

(continues on next page)

(continued from previous page)

```

--min-rtt-timeout 1s'. xCAT will append the options defined
here to
the nmap command.

ntpservers: A comma delimited list of NTP servers for the service node and
the compute node to sync with. The keyword <xcatmaster> means
that
the node's NTP server is the node that is managing it
(either its service node or the management node).

extntpservers: A comma delimited list of external NTP servers for the xCAT
management node to sync with. If it is empty, the NTP server
will use the management node's own hardware clock to calculate
the system date and time

svloglocal: If set to 1, syslog on the service node will not get forwarded
to the
management node.

timezone: (e.g. America/New_York)

tftpdirdir: The tftp directory path. Default is /tftpboot

tftpflags: The flags that used to start tftpd. Default is '-v -l -s /tftpboot
-m /etc/tftpmapfile4xcat.conf' if tftpflags is not set

useNmapfromMN: When set to yes, nodestat command should obtain the node
status
using nmap (if available) from the management node instead of
the
service node. This will improve the performance in a flat
network.

vsftpd: Default is 'n'. If set to 'y', xcatd on the management node will
automatically
start vsftpd. (vsftpd must be installed by the admin). This setting
does not
apply to service nodes. For service nodes, set servicenode.
ftpserver=1.

FQDNfirst: Fully Qualified Domain Name first. If set to 1/yes/enable, the /
etc/hosts
entries generated by 'makehosts' will put the FQDN before the
PQDN(Partially
Qualified Domain Name). Otherwise, the original behavior will be
performed.

hierarchicalattrs: A comma delimited list of table attributes(e.g.
postscripts, postbootscripts)
that will be included hierarchically. Attribute values
for all the node's groups
will be applied to the node in the groups' order except

```

(continues on next page)

(continued from previous page)

```

↳the repeat one.
    By default, comma is used to combine the values. But some
↳columns use different
    delimiter, to specify delimiter for those columns as
↳format of 'column:delimiter'.

dbtracelevel: The trace level for the database access log. To activate this
↳setting, please.
    restart xcatd or send HUP signal to the 'xcatd: DB Access'
↳process, Like: .
    pkill -f -HUP 'xcatd: DB Access'
    Current support values:
    0: disable the trace log for db
    1: trace the calls of database subroutines
    2: Besides the log from level 1, trace the event to build the
↳cache for the table
    3: Besides the log from level 2, trace the event with cache hit
    4: Besides the log from level 3, trace the SQL statement for
↳the db access
    With this configuration, xcat will send the log to syslog very
↳frequently, some of the
    log may be lost if imjournal is enabled by rsyslog.
    Please see https://github.com/xcat2/xcat-core/issues/3910 for
↳the detail.

-----
VIRTUALIZATION ATTRIBUTES
-----

usexhrm: Have xCAT execute the xHRM script when booting up KVM guests to
↳configure
    the virtual network bridge.

vcenterautojoin: When set to no, the VMWare plugin will not attempt to auto
↳remove
    and add hypervisors while trying to perform operations. If
↳users
    or tasks outside of xCAT perform the joining this assures
↳xCAT
    will not interfere.

vmwarereconfigonpower: When set to no, the VMWare plugin will make no effort
↳to
    push vm.cpus/vm.memory updates from xCAT to VMWare.

persistkvmguests: Keep the kvm definition on the kvm hypervisor when you
↳power off
    the kvm guest node. This is useful for you to manually
↳change the
    kvm xml definition file in virsh for debugging. Set
↳anything means
    enable.

```

(continues on next page)

(continued from previous page)

```

-----
XCAT DAEMON ATTRIBUTES
-----
tokenexpiredays: Number of days before REST API token will expire. The
↪default is 1.
                use 'never' if you want your token to never expire.
useflowcontrol: (yes/1 or no/0). If yes, the postscript processing on each
↪node
                contacts xcatd on the MN/SN using a lightweight UDP packet to
↪wait
                until xcatd is ready to handle the requests associated with
                postscripts. This prevents deploying nodes from flooding xcatd.
↪and
                locking out admin interactive use. This value works with the
                xcatmaxconnections and xcatmaxbatch attributes. Is not
↪supported on AIX.
                If the value is no, nodes sleep for a random time before
↪contacting
                xcatd, and retry. The default is no.
                See the following document for details:
                Hints_and_Tips_for_Large_Scale_Clusters

xcatmaxconnections: Number of concurrent xCAT protocol requests before
↪requests
                begin queueing. This applies to both client command
↪requests
                and node requests, e.g. to get postscripts. Default is
↪64.

xcatmaxbatchconnections: Number of concurrent xCAT connections allowed from
↪the nodes.
                Value must be less than xcatmaxconnections. Default is
↪50.

xcatdport: The port used by the xcatd daemon for client/server communication.

xcatiport: The port used by xcatd to receive install status updates from
↪nodes.

xcatlport: The port used by xcatd command log writer process to collect
↪command output.

xcatsslversion: This is the SSL_version option xcatd used and passed to
↪to empty.
                IO::Socket::SSL->start_SSL(). By default, this value is set
↪internally.
                In this case, xcatd will use SSLv23:!SSLv2:!SSLv3:!TLSv1
                For more detail, see https://metacpan.org/pod/IO::Socket::SSL

xcatsslciphers: The ssl cipher by xcatd. Default is 3DES.

```

value

The value of the attribute specified in the “key” column.

**comments**

Any user-written notes.

**disable**

Set to ‘yes’ or ‘1’ to comment out this row.

**SEE ALSO**

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

**statelite.5**

**NAME**

**statelite** - a table in the xCAT database.

**SYNOPSIS**

**statelite Attributes:** *node, image, statemnt, mntopts, comments, disable*

**DESCRIPTION**

The location on an NFS server where a nodes persistent files are stored. Any file marked persistent in the litefile table will be stored in the location specified in this table for that node.

**statelite Attributes:**

**node**

The name of the node or group that will use this location.

**image**

Reserved for future development, not used.

**statemnt**

The persistent read/write area where a node’s persistent files will be written to, e.g: 10.0.0.1/state/. The node name will be automatically added to the pathname, so 10.0.0.1:/state, will become 10.0.0.1:/state/<nodename>.

**mntopts**

A comma-separated list of options to use when mounting the persistent directory. (Ex. ‘soft’) The default is to do a ‘hard’ mount.

**comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## storage.5

### NAME

**storage** - a table in the xCAT database.

### SYNOPSIS

**storage Attributes:** *node, osvolume, size, state, storagepool, hypervisor, fcprange, volumetag, type, controller, comments, disable*

### DESCRIPTION

#### storage Attributes:

##### node

The node name

##### osvolume

Specification of what storage to place the node OS image onto. Examples include:

```
localdisk (Install to first non-FC attached disk)
usbdisk (Install to first USB mass storage device seen)
wwn=0x50000393c813840c (Install to storage device with given WWN)
```

##### size

Size of the volume. Examples include: 10G, 1024M.

##### state

State of the volume. The valid values are: free, used, and allocated

##### storagepool

Name of storage pool where the volume is assigned.

##### hypervisor

Name of the hypervisor where the volume is configured.

##### fcprange

A range of acceptable fibre channels that the volume can use. Examples include: 3B00-3C00;4B00-4C00.

##### volumetag

A specific tag used to identify the volume in the autoyast or kickstart template.



### type

The plugin used to drive storage configuration (e.g. svc)

### controller

The management address to attach/detach new volumes. In the scenario involving multiple controllers, this data must be passed as argument rather than by table value

### comments

Any user-written notes.

### disable

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## switch.5

## NAME

**switch** - a table in the xCAT database.

## SYNOPSIS

**switch Attributes:** *node, switch, port, vlan, interface, comments, disable*

## DESCRIPTION

Contains what switch port numbers each node is connected to.

### switch Attributes:

#### node

The node name or group name.

#### switch

The switch hostname.

#### port

The port number in the switch that this node is connected to. On a simple 1U switch, an administrator can generally enter the number as printed next to the ports, and xCAT will understand switch representation differences. On stacked switches or switches with line cards, administrators should usually use the CLI representation (i.e. 2/0/1 or 5/8). One notable exception is stacked SMC 8848M switches, in which you must add 56 for the proceeding switch, then the port number. For example, port 3 on the second switch in an SMC8848M stack would be 59

#### vlan

The ID for the tagged vlan that is created on this port using `mkvlan` and `chvlan` commands.

**interface**

The interface name from the node perspective. For example, `eth0`. For the primary nic, it can be empty, the word “primary” or “primary:ethx” where ethx is the interface name.

**comments**

Any user-written notes.

**disable**

Set to ‘yes’ or ‘1’ to comment out this row.

**SEE ALSO**

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

**switches.5**

**NAME**

**switches** - a table in the xCAT database.

**SYNOPSIS**

**switches Attributes:** *switch, snmpversion, username, password, privacy, auth, linkports, sshusername, sshpassword, protocol, switchtype, comments, disable*

**DESCRIPTION**

Parameters to use when interrogating switches

**switches Attributes:**

**switch**

The hostname/address of the switch to which the settings apply

**snmpversion**

The version to use to communicate with switch. SNMPv1 is assumed by default.

**username**

The username to use for SNMPv3 communication, ignored for SNMPv1

**password**

The password string for SNMPv3 or community string for SNMPv1/SNMPv2. Falls back to `passwd` table, and site `snmpc` value if using SNMPv1/SNMPv2.

**privacy**

The privacy protocol to use for v3. xCAT will use authNoPriv if this is unspecified. DES is recommended to use if v3 enabled, as it is the most readily available.

**auth**

The authentication protocol to use for SNMPv3. SHA is assumed if v3 enabled and this is unspecified

**linkports**

The ports that connect to other switches. Currently, this column is only used by vlan configuration. The format is: “port\_number:switch,port\_number:switch...”. Refer to the switch table for details on how to specify the port numbers.

**sshusername**

The remote login user name. It can be for ssh or telnet. If it is for telnet, set protocol to “telnet”. If the sshusername is blank, the username, password and protocol will be retrieved from the passwd table with “switch” as the key.

**sshpasword**

The remote login password. It can be for ssh or telnet. If it is for telnet, set protocol to “telnet”. If the sshusername is blank, the username, password and protocol will be retrieved from the passwd table with “switch” as the key.

**protocol**

Protocol for running remote commands for the switch. The valid values are: ssh, telnet. ssh is the default. If the sshusername is blank, the username, password and protocol will be retrieved from the passwd table with “switch” as the key. The passwd.comments attribute is used for protocol.

**switchtype**

The type of switch. It is used to identify the file name that implements the functions for this switch. The valid values are: Mellanox, Cisco, BNT and Juniper.

**comments****disable****SEE ALSO**

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

**taskstate.5****NAME**

**taskstate** - a table in the xCAT database.

## SYNOPSIS

**taskstate Attributes:** *node, command, state, pid, reserve, disable*

## DESCRIPTION

The task state for the node.

### **taskstate Attributes:**

#### **node**

The node name.

#### **command**

Current command is running

#### **state**

The task state(callback, running) for the node.

#### **pid**

The process id of the request process.

#### **reserve**

used to lock the node

#### **disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

## token.5

## NAME

**token** - a table in the xCAT database.

## SYNOPSIS

**token Attributes:** *tokenid, username, expire, created, access, comments, disable*

## DESCRIPTION

The token of users for authentication.

### **token Attributes:**

#### **tokenid**

It is a UUID as an unified identify for the user.

#### **username**

The user name.

#### **expire**

The expire time for this token.

#### **created**

Creation time for this token.

#### **access**

Last access time for this token.

#### **comments**

Any user-provided notes.

#### **disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**models(1), chtab(8), tabdump(8), tabedit(8)**

### **virtsd.5**

## NAME

**virtsd** - a table in the xCAT database.

## SYNOPSIS

**virtsd Attributes:** *node, sdtype, stype, location, host, cluster, datacenter, comments, disable*

## **DESCRIPTION**

The parameters which used to create the Storage Domain

### **virtasd Attributes:**

#### **node**

The name of the storage domain

#### **sdtype**

The type of storage domain. Valid values: data, iso, export

#### **stype**

The type of storage. Valid values: nfs, fcp, iscsi, localfs

#### **location**

The path of the storage

#### **host**

For rhev, a hypervisor host needs to be specified to manage the storage domain as SPM (Storage Pool Manager). But the SPM role will be failed over to another host when this host down.

#### **cluster**

A cluster of hosts

#### **datacenter**

A collection for all host, vm that will shared the same storages, networks.

#### **comments**

#### **disable**

## **SEE ALSO**

**nodels(1), chtab(8), tabdump(8), tabedit(8)**

### **vm.5**

## **NAME**

**vm** - a table in the xCAT database.

## SYNOPSIS

**vm Attributes:** *node, mgr, host, migrationdest, storage, storagemodel, storagecache, storageformat, cfgstore, memory, cpus, nics, nicmodel, bootorder, clockoffset, virtflags, master, vncport, textconsole, powerstate, beacon, datacenter, cluster, guestostype, othersettings, physlots, vidmodel, vidproto, vidpassword, comments, disable*

## DESCRIPTION

Virtualization parameters

### vm Attributes:

#### node

The node or static group name

#### mgr

The function manager for the virtual machine

#### host

The system that currently hosts the VM

#### migrationdest

A noderange representing candidate destinations for migration (i.e. similar systems, same SAN, or other criteria that xCAT can use)

#### storage

A list of storage files or devices to be used. i.e. `dir:///cluster/vm/<nodename>` or `nfs://<server>/path/to/folder/`

#### storagemodel

Model of storage devices to provide to guest

#### storagecache

Select caching scheme to employ. E.g. KVM understands 'none', 'writethrough' and 'writeback'

#### storageformat

Select disk format to use by default (e.g. raw versus qcow2)

#### cfgstore

Optional location for persistent storage separate of emulated hard drives for virtualization solutions that require persistent store to place configuration data

#### memory

Megabytes of memory the VM currently should be set to.

#### cpus

Number of CPUs the node should see.

#### nics

Network configuration parameters. Of the general form [physnet:]interface,... Generally, interface describes the vlan entity (default for native, tagged for tagged, vl[number] for a specific vlan. physnet is a virtual switch name or port description that is used for some virtualization technologies to construct virtual switches. hypervisor.netmap can map names to hypervisor specific layouts, or the descriptions described there may be used directly here where possible.

**nicmodel**

Model of NICs that will be provided to VMs (i.e. e1000, rtl8139, virtio, etc)

**bootorder**

Boot sequence (i.e. net,hd)

**clockoffset**

Whether to have guest RTC synced to “localtime” or “utc” If not populated, xCAT will guess based on the nodetype.os contents.

**virtflags**

**General flags used by the virtualization method.**

**For example, in Xen it could, among other things, specify paravirtualized setup, or direct kernel boot. For a hypervisor/dom0 entry, it is the virtualization method (i.e. “xen”). For KVM, the following flag=value pairs are recognized:**

**imageformat=[raw|fullraw|qcow2]**

raw is a generic sparse file that allocates storage on demand fullraw is a generic, non-sparse file that preallocates all space qcow2 is a sparse, copy-on-write capable format implemented at the virtualization layer rather than the filesystem level

**clonemethod=[qemu-img|relink]**

qemu-img allows use of qcow2 to generate virtualization layer copy-on-write relink uses a generic filesystem facility to clone the files on your behalf, but requires filesystem support such as btrfs

**placement\_affinity=[migratable|user\_migratable|pinned]**

**master**

The name of a master image, if any, this virtual machine is linked to. This is generally set by clonevm and indicates the deletion of a master that would invalidate the storage of this virtual machine

**vncport**

Tracks the current VNC display port (currently not meant to be set

**textconsole**

Tracks the Psuedo-TTY that maps to the serial port or console of a VM

**powerstate**

This flag is used by xCAT to track the last known power state of the VM.

**beacon**

This flag is used by xCAT to track the state of the identify LED with respect to the VM.

**datacenter**

Optionally specify a datacenter for the VM to exist in (only applicable to VMWare)

**cluster**

Specify to the underlying virtualization infrastructure a cluster membership for the hypervisor.



**guestostype**

This allows administrator to specify an identifier for OS to pass through to virtualization stack. Normally this should be ignored as xCAT will translate from nodetype.os rather than requiring this field be used

**othersettings**

**This is a semicolon-delimited list of key-value pairs to be included in a vmx file of VMware or KVM. DO NOT use ‘chdef <node> -p|-m vmothersetting=...’ to add options to it or delete options from it because chdef uses commas, not semicolons, to separate items.**

**Hugepage on POWER systems:**

Specify the hugepage and/or bsr (Barrier Synchronization Register) values, e.g., ‘hugepage:1,bsr:2’.

**KVM CPU mode:**

Specify how the host CPUs are utilized, e.g., ‘cpumode:host-passthrough’, ‘cpumode:host-model’. With the passthrough mode, the performance of x86 VMs can be improved significantly.

**KVM CPU pinning:**

Specify which host CPUs are used, e.g., ‘vcpupin:’0-15,^8’, where ‘-’ denotes the range and ‘^’ denotes exclusion. This option allows a comma-delimited list.

**KVM memory binding:**

Specify which nodes that host memory are used, e.g., ‘membind:0’, where the memory in node0 of the hypervisor is used. /sys/devices/system/node has node0 and node8 on some POWER systems, node0 and node1 on some x86\_64 systems. This option allows a guest VM to access specific memory regions.

**PCI passthrough:**

PCI devices can be assigned to a virtual machine for exclusive usage, e.g., ‘dev-passthrough:pci\_0001\_01\_00\_0,pci\_0000\_03\_00\_0’. A PCI device can also be expressed as ‘devpassthrough:0001:01:00.1’. The devices are put in a comma-delimited list. The PCI device names can be obtained by running **virsh nodedev-list** on the host.

**VM machine type:**

Specify a machine type for VM creation on the host, e.g., ‘machine:pc’. Typical machine types are pc, q35, and pseries.

**physlots**

Specify the physical slots drc index that will assigned to the partition, the delimiter is ‘,’ and the drc index must started with ‘0x’. For more details, reference manpage for ‘lsvm’.

**vidmodel**

Model of video adapter to provide to guest. For example, qxl in KVM

**vidproto**

Request a specific protocol for remote video access be set up. For example, spice in KVM.

**vidpassword**

Password to use instead of temporary random tokens for VNC and SPICE access

**comments****disable**

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## vmmaster.5

## NAME

**vmmaster** - a table in the xCAT database.

## SYNOPSIS

**vmmaster Attributes:** *name, os, arch, profile, storage, storagemodel, nics, vintage, originator, virttype, specializeparameters, comments, disable*

## DESCRIPTION

Inventory of virtualization images for use with clonevm. Manual intervention in this table is not intended.

### vmmaster Attributes:

#### **name**

The name of a master

#### **os**

The value of nodetype.os at the time the master was captured

#### **arch**

The value of nodetype.arch at the time of capture

#### **profile**

The value of nodetype.profile at time of capture

#### **storage**

The storage location of bulk master information

#### **storagemodel**

The default storage style to use when modifying a vm cloned from this master

#### **nic**

The nic configuration and relationship to vlans/bonds/etc

#### **vintage**

When this image was created

#### **originator**

The user who created the image

#### **virttype**

The type of virtualization this image pertains to (e.g. vmware, kvm, etc)

#### **specializeparameters**

Implementation specific arguments, currently only “autoLogonCount=<number>” for ESXi clonevme

#### **comments**

#### **disable**

### **SEE ALSO**

**nodes(1), chtag(8), tabdump(8), tabedit(8)**

### **vpd.5**

### **NAME**

**vpd** - a table in the xCAT database.

### **SYNOPSIS**

**vpd Attributes:** *node, serial, mtm, side, asset, uuid, comments, disable*

### **DESCRIPTION**

The Machine type, Model, and Serial numbers of each node.

#### **vpd Attributes:**

##### **node**

The node name or group name.

##### **serial**

The serial number of the node.

##### **mtm**

The machine type and model number of the node. E.g. 7984-6BU

##### **side**

<BPA>-<port> or <FSP>-<port>. The side information for the BPA/FSP. The side attribute refers to which BPA/FSP, A or B, which is determined by the slot value returned from lsslp command. It also lists the physical port within each BPA/FSP which is determined by the IP address order from the lsslp response. This information is used internally when communicating with the BPAs/FSPs

##### **asset**

A field for administrators to use to correlate inventory numbers they may have to accommodate

##### **uuid**

The UUID applicable to the node

**comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

**SEE ALSO**

**models(1), chtab(8), tabdump(8), tabedit(8)**

**websrv.5**

**NAME**

**websrv** - a table in the xCAT database.

**SYNOPSIS**

**websrv Attributes:** *node, port, username, password, comments, disable*

**DESCRIPTION**

Web service parameters

**websrv Attributes:**

**node**

The web service hostname.

**port**

The port of the web service.

**username**

Userid to use to access the web service.

**password**

Password to use to access the web service.

**comments**

Any user-written notes.

**disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## winimage.5

### NAME

**winimage** - a table in the xCAT database.

### SYNOPSIS

**winimage Attributes:** *imagename, template, installto, partitionfile, winpepath, comments, disable*

### DESCRIPTION

Information about a Windows operating system image that can be used to deploy cluster nodes.

#### **winimage Attributes:**

##### **imagename**

The name of this xCAT OS image definition.

##### **template**

The fully qualified name of the template file that is used to create the windows unattend.xml file for diskful installation.

##### **installto**

The disk and partition that the Windows will be deployed to. The valid format is <disk>:<partition>. If not set, default value is 0:1 for bios boot mode(legacy) and 0:3 for uefi boot mode; If setting to 1, it means 1:1 for bios boot and 1:3 for uefi boot

##### **partitionfile**

The path of partition configuration file. Since the partition configuration for bios boot mode and uefi boot mode are different, this configuration file can include both configurations if you need to support both bios and uefi mode. Either way, you must specify the boot mode in the configuration. Example of partition configuration file: [BIOS]xxxxxxx[UEFI]yyyyyyy. To simplify the setting, you also can set installto in partitionfile with section like [INSTALLTO]0:1

##### **winpepath**

The path of winpe which will be used to boot this image. If the real path is /tftpboot/winboot/winpe1/, the value for winpepath should be set to winboot/winpe1

##### **comments**

Any user-written notes.

##### **disable**

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## xcatdb.5

### NAME

An overview of the xCAT database.

### DESCRIPTION

The xCAT database contains user settings for the cluster and information gathered from the cluster. It consists of a series of tables, which are described below. To get more information about a particular table, run `man` for that table name. The tables can be manipulated directly using the **tabedit** or **chtab** commands. They can be viewed using **nodels** or **tabdump**.

Alternatively, the xCAT database can be viewed and edited as logical objects, instead of flat tables. In this mode, xCAT takes care of which table each attribute should go in. To treat the database as logical object definitions, use the commands: **lsdef**, **mkdef**, **chdef**, **rmdef**. See Object Definitions below.

xCAT allows the use of different database applications, depending on the needs of your cluster. The default database is SQLite, which is a daemonless, zero-config database. But you could instead choose to use something like postgresql for greater scalability and remote access in the hierarchical/service node case. To use a different database or a different location, create the file `/etc/xcat/cfgloc`. See the appropriate xCAT documentation for the format of the file for the database you choose. The following example `/etc/xcat/cfgloc` file is for PostgreSQL:

```
Pg:dbname=xcat;host=<mgmtnode>|<pgadminuserid>|<pgadminpasswd>
```

where `mgmtnode` is the hostname of the management node adapter on the cluster side, and the `pgadminuserid` and `pgadminpasswd` are the database admin and password.

### GROUPS AND REGULAR EXPRESSIONS IN TABLES

The xCAT database has a number of tables, some with rows that are keyed by node name (such as `noderes` and `nodehm`) and others that are not keyed by node name (for example, the `policy` table). The tables that are keyed by node name have some extra features that enable a more template-based style to be used:

Any group name can be used in lieu of a node name in the node field, and that row will then provide “default” attribute values for any node in that group. A row with a specific node name can then override one or more attribute values for that specific node. For example, if the `nodehm` table contains:

```
#node,power,mgt,cons,termserver,termport,conserver,serialport,serialspeed,serialflow,
↪getmac,cmdmapping,comments,disable
"mygroup",,"ipmi",,,,,,"19200",,,,,
"node1",,,,,,,,"115200",,,,,
```

In the above example, the node group called `mygroup` sets `mgt=ipmi` and `serialspeed=19200`. Any nodes that are in this group will have those attribute values, unless overridden. For example, if `node2` is a member of `mygroup`, it will automatically inherit these attribute values (even though it is not explicitly listed in this table). In the case of `node1` above, it inherits `mgt=ipmi`, but overrides the `serialspeed` to be `115200`, instead of `19200`. A useful, typical way to use

this capability is to create a node group for your nodes and for all the attribute values that are the same for every node, set them at the group level. Then you only have to set attributes for each node that vary from node to node.

xCAT extends the group capability so that it can also be used for attribute values that vary from node to node in a very regular pattern. For example, if in the ipmi table you want the bmc attribute to be set to whatever the nodename is with “-bmc” appended to the end of it, then use this in the ipmi table:

```
#node,bmc,bmcport,taggedvlan,bmcid,username,password,comments,disable
"compute","/z/-bmc/",,,,,,,,,,
```

In this example, “compute” is a node group that contains all of the compute nodes. The 2nd attribute (bmc) is a regular expression that is similar to a substitution pattern. The 1st part “z” matches the end of the node name and substitutes “-bmc”, effectively appending it to the node name.

Another example is if node1 is to have IP address 10.0.0.1, node2 is to have IP address 10.0.0.2, etc., then this could be represented in the hosts table with the single row:

```
#node,ip,hostnames,otherinterfaces,comments,disable
"compute","|node(\d+)|10.0.0.($1+0)|",,,, ,
```

In this example, the regular expression in the ip attribute uses “|” to separate the 1st and 2nd part. This means that xCAT will allow arithmetic operations in the 2nd part. In the 1st part, “(d+)”, will match the number part of the node name and put that in a variable called \$1. The 2nd part is what value to give the ip attribute. In this case it will set it to the string “10.0.0.” and the number that is in \$1. (Zero is added to \$1 just to remove any leading zeroes.)

A more involved example is with the mp table. If your blades have node names node01, node02, etc., and your chassis node names are cmm01, cmm02, etc., then you might have an mp table like:

```
#node,mpa,id,nodetype,comments,disable
"blade","|\D+(\d+)|cmm(sprintf('%02d',($1-1)/14+1))|",",|\D+(\d+)|((($1-1)%14+1)|",,
```

Before you panic, let me explain each column:

#### blade

This is a group name. In this example, we are assuming that all of your blades belong to this group. Each time the xCAT software accesses the **mp** table to get the management module and slot number of a specific blade (e.g. **node20**), this row will match (because **node20** is in the **blade** group). Once this row is matched for **node20**, then the processing described in the following items will take place.

**|D+(d+)|cmm(sprintf('%02d',(\$1-1)/14+1))|**

This is a perl substitution pattern that will produce the value for the second column of the table (the management module hostname). The text **D+(d+)** between the 1st two vertical bars is a regular expression that matches the node name that was searched for in this table (in this example **node20**). The text that matches within the 1st set of parentheses is set to \$1. (If there was a 2nd set of parentheses, it would be set to \$2, and so on.) In our case, the **D+** matches the non-numeric part of the name (**node**) and the **d+** matches the numeric part (**20**). So \$1 is set to **20**. The text **cmm(sprintf('%02d',(\$1-1)/14+1))** between the 2nd and 3rd vertical bars produces the string that should be used as the value for the mpa attribute for node20. Since \$1 is set to 20, the expression **(\$1-1)/14+1** equals 19/14 + 1, which equals 2. (The division is integer division, so 19/14 equals 1. Fourteen is used as the divisor, because there are 14 blades in each chassis.) The value of 2 is then passed into sprintf() with a format string to add a leading zero, if necessary, to always make the number two digits. Lastly the string **cmm** is added to the beginning, making the resulting string **cmm02**, which will be used as the hostname of the management module.

**|D+(d+)|(((\$1-1)%14+1)|**

This item is similar to the one above. This substitution pattern will produce the value for the 3rd column (the chassis slot number for this blade). Because this row was the match for **node20**, the parentheses within

the 1st set of vertical bars will set \$1 to 20. Since % means modulo division, the expression **(\$1-1)%14+1** will evaluate to **6**.

See <http://www.perl.com/doc/manual/html/pod/perlre.html> for information on perl regular expressions.

## Easy Regular Expressions

As of xCAT 2.8.1, you can use a modified version of the regular expression support described in the previous section. You do not need to enter the node information (1st part of the expression), it will be derived from the input nodename. You only need to supply the 2nd part of the expression to determine the value to give the attribute. For examples, see

[http://xcat-docs.readthedocs.org/en/latest/guides/admin-guides/basic\\_concepts/xcat\\_db/regexp\\_db.html#easy-regular-expressions](http://xcat-docs.readthedocs.org/en/latest/guides/admin-guides/basic_concepts/xcat_db/regexp_db.html#easy-regular-expressions)

## Regular Expression Helper Functions

xCAT provides several functions that can simplify regular expressions.

### **a2idx**

ASCII Character to Index

### **a2zidx**

ASCII Character to 0-Index

### **dim2idx**

Dimensions to Index

### **skip**

Skip indices

### **ipadd**

Add to an IP address

[http://xcat-docs.readthedocs.org/en/latest/guides/admin-guides/basic\\_concepts/xcat\\_db/regexp\\_db.html#regular-expression-helper-functions](http://xcat-docs.readthedocs.org/en/latest/guides/admin-guides/basic_concepts/xcat_db/regexp_db.html#regular-expression-helper-functions)

## OBJECT DEFINITIONS

Because it can get confusing what attributes need to go in what tables, the xCAT database can also be viewed and edited as logical objects, instead of flat tables. Use **mkdef**, **chdef**, **lsdef**, and **rmdef** to create, change, list, and delete objects. When using these commands, the object attributes will be stored in the same tables, as if you edited the tables by hand. The only difference is that the object commands take care of knowing which tables all of the information should go in.



## xCAT Object Name Format:

**xCAT Object Name Format** is defined by the following regex:

```
^([A-Za-z-]+)([0-9]+)(([A-Za-z-]+[A-Za-z0-9-]*)*)
```

In plain English, an object name is in **xCAT Object Name Format** if starting from the beginning there are:

\*

one or more alpha characters of any case and any number of “-” in any combination

\*

followed by one or more numbers

\*

then optionally followed by one alpha character of any case or “-”

\*

followed by any combination of case mixed alphanumerics and “-”

## Object Types

To run man for any of the object definitions below, use section 7. For example: **man 7 node**

The object types are:

auditlog(7)|auditlog.7

boottarget(7)|boottarget.7

eventlog(7)|eventlog.7

firmware(7)|firmware.7

group(7)|group.7

kit(7)|kit.7

kitcomponent(7)|kitcomponent.7

kitrepo(7)|kitrepo.7

monitoring(7)|monitoring.7

network(7)|network.7

node(7)|node.7

notification(7)|notification.7

osdistro(7)|osdistro.7

osdistroudate(7)|osdistroudate.7

osimage(7)|osimage.7

pdu(7)|pdu.7

policy(7)|policy.7

rack(7)|rack.7

route(7)|route.7

site(7)|site.7  
taskstate(7)|taskstate.7  
zone(7)|zone.7  
zvmivp(7)|zvmivp.7

## TABLES

To manipulate the tables directly, use **nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**, **nodeadd(8)**, **nodech(1)**.

To run man for any of the table descriptions below, use section 5. For example: **man 5 nodehm**

The tables are:

auditlog(5)|auditlog.5

Audit Data log.

bootparams(5)|bootparams.5

Current boot settings to be sent to systems attempting network boot for deployment, stateless, or other reasons. Mostly automatically manipulated by xCAT.

boottarget(5)|boottarget.5

Specify non-standard initrd, kernel, and parameters that should be used for a given profile.

cfgmgt(5)|cfgmgt.5

Configuration management data for nodes used by non-xCAT osimage management services to install and configure software on a node.

chain(5)|chain.5

Controls what operations are done (and in what order) when a node is discovered and deployed.

deps(5)|deps.5

Describes dependencies some nodes have on others. This can be used, e.g., by `rpower -d` to power nodes on or off in the correct order.

discoverydata(5)|discoverydata.5

Discovery data which sent from genesis.

domain(5)|domain.5

Mapping of nodes to domain attributes

eventlog(5)|eventlog.5

Stores the events occurred.

firmware(5)|firmware.5

Maps node to firmware values to be used for setup at node discovery or later

hosts(5)|hosts.5

IP addresses and hostnames of nodes. This info is optional and is only used to populate `/etc/hosts` and DNS via `makehosts` and `makedns`. Using regular expressions in this table can be a quick way to populate `/etc/hosts`.

hwinv(5)|hwinv.5

The hardware inventory for the node.

hypervisor(5)|hypervisor.5

Hypervisor parameters

ipmi(5)|ipmi.5

Settings for nodes that are controlled by an on-board BMC via IPMI.

iscsi(5)|iscsi.5

Contains settings that control how to boot a node from an iSCSI target

kit(5)|kit.5

This table stores all kits added to the xCAT cluster.

kitcomponent(5)|kitcomponent.5

This table stores all kit components added to the xCAT cluster.

kitrepo(5)|kitrepo.5

This table stores all kits added to the xCAT cluster.

kvm\_masterdata(5)|kvm\_masterdata.5

Persistent store for KVM plugin for masters

kvm\_nodedata(5)|kvm\_nodedata.5

Persistent store for KVM plugin, not intended for manual modification.

linuximage(5)|linuximage.5

Information about a Linux operating system image that can be used to deploy cluster nodes.

litefile(5)|litefile.5

The litefile table specifies the directories and files on the statelite nodes that should be readwrite, persistent, or readonly overlay. All other files in the statelite nodes come from the readonly statelite image.

litetree(5)|litetree.5

Directory hierarchy to traverse to get the initial contents of node files. The files that are specified in the litefile table are searched for in the directories specified in this table.

mac(5)|mac.5

The MAC address of the node's install adapter. Normally this table is populated by getmacs or node discovery, but you can also add entries to it manually.

mic(5)|mic.5

The host, slot id and configuration of the mic (Many Integrated Core).

monitoring(5)|monitoring.5

Controls what external monitoring tools xCAT sets up and uses. Entries should be added and removed from this table using the provided xCAT commands monstart and monstop.

monsetting(5)|monsetting.5

Specifies the monitoring plug-in specific settings. These settings will be used by the monitoring plug-in to customize the behavior such as event filter, sample interval, responses etc. Entries should be added, removed or modified by chtag command. Entries can also be added or modified by the monstart command when a monitoring plug-in is brought up.

`mp(5)|mp.5`

Contains the hardware control info specific to blades. This table also refers to the mpa table, which contains info about each Management Module.

`mpa(5)|mpa.5`

Contains info about each Management Module and how to access it.

`networks(5)|networks.5`

Describes the networks in the cluster and info necessary to set up nodes on that network.

`nics(5)|nics.5`

Stores NIC details.

`nimimage(5)|nimimage.5`

All the info that specifies a particular AIX operating system image that can be used to deploy AIX nodes.

`nodegroup(5)|nodegroup.5`

Contains group definitions, whose membership is dynamic depending on characteristics of the node.

`nodehm(5)|nodehm.5`

Settings that control how each node's hardware is managed. Typically, an additional table that is specific to the hardware type of the node contains additional info. E.g. the ipmi, mp, and ppc tables.

`odelist(5)|odelist.5`

The list of all the nodes in the cluster, including each node's current status and what groups it is in.

`nodepos(5)|nodepos.5`

Contains info about the physical location of each node. Currently, this info is not used by xCAT, and therefore can be in whatever format you want. It will likely be used in xCAT in the future.

`nodes(5)|nodes.5`

Resources and settings to use when installing nodes.

`nodetype(5)|nodetype.5`

A few hardware and software characteristics of the nodes.

`notification(5)|notification.5`

Contains registrations to be notified when a table in the xCAT database changes. Users can add entries to have additional software notified of changes. Add and remove entries using the provided xCAT commands `regnotif` and `unregnotif`.

`openbmc(5)|openbmc.5`

Setting for nodes that are controlled by an on-board OpenBMC.

`osdistro(5)|osdistro.5`

Information about all the OS distros in the xCAT cluster

`osdistroudate(5)|osdistroudate.5`

Information about the OS distro updates in the xCAT cluster

`osimage(5)|osimage.5`

Basic information about an operating system image that can be used to deploy cluster nodes.

`passwd(5)|passwd.5`

Contains default userids and passwords for xCAT to access cluster components. In most cases, xCAT will also actually set the userid/password in the relevant component when it is being configured or installed. Userids/passwords for specific cluster components can be overridden in other tables, e.g. mpa, ipmi, ppchcp, etc.

pdu(5)|pdu.5

Parameters to use when interrogating pdus

pduoutlet(5)|pduoutlet.5

Contains list of outlet numbers on the pdu each node is connected to.

performance(5)|performance.5

Describes the system performance every interval unit of time.

policy(5)|policy.5

The policy table in the xCAT database controls who has authority to run specific xCAT operations. It is basically the Access Control List (ACL) for xCAT. It is sorted on the priority field before evaluating.

postscripts(5)|postscripts.5

The scripts that should be run on each node after installation or diskless boot.

ppc(5)|ppc.5

List of system p hardware: HMCs, IVMs, FSPs, BPCs, CECs, Frames.

ppcdirect(5)|ppcdirect.5

Info necessary to use FSPs/BPAs to control system p CECs/Frames.

ppchcp(5)|ppchcp.5

Info necessary to use HMCs and IVMs as hardware control points for LPARs.

prescripts(5)|prescripts.5

The scripts that will be run at the beginning and the end of the nodeset(Linux), nimnodeset(AIX) or mkdsklnode(AIX) command.

prodkey(5)|prodkey.5

Specify product keys for products that require them

rack(5)|rack.5

Rack information.

routes(5)|routes.5

Describes the additional routes needed to be setup in the os routing table. These routes usually are used to connect the management node to the compute node using the service node as gateway.

servicenode(5)|servicenode.5

List of all Service Nodes and services that will be set up on the Service Node.

site(5)|site.5

Global settings for the whole cluster. This table is different from the other tables in that each attribute is just named in the key column, rather than having a separate column for each attribute. The following is a list of attributes currently used by xCAT organized into categories.

statelite(5)|statelite.5

The location on an NFS server where a nodes persistent files are stored. Any file marked persistent in the litefile table will be stored in the location specified in this table for that node.

storage(5)|storage.5

switch(5)|switch.5

Contains what switch port numbers each node is connected to.

switches(5)|switches.5

Parameters to use when interrogating switches

taskstate(5)|taskstate.5

The task state for the node.

token(5)|token.5

The token of users for authentication.

virtsd(5)|virtsd.5

The parameters which used to create the Storage Domain

vm(5)|vm.5

Virtualization parameters

vmmaster(5)|vmmaster.5

Inventory of virtualization images for use with clonevm. Manual intervention in this table is not intended.

vpd(5)|vpd.5

The Machine type, Model, and Serial numbers of each node.

websrv(5)|websrv.5

Web service parameters

winimage(5)|winimage.5

Information about a Windows operating system image that can be used to deploy cluster nodes.

zone(5)|zone.5

Defines a cluster zone for nodes that share root ssh key access to each other.

zvm(5)|zvm.5

List of z/VM virtual servers.

zvmivp(5)|zvmivp.5

List of z/VM Installation Verification Procedures (IVPs) to be periodically run.

## SEE ALSO

**nodels(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**, **lsdef(1)**, **mkdef(1)**, **chdef(1)**, **rmdef(1)**

## xcatstanzafile.5

## NAME

**xcatstanzafile** - Format of a stanza file that can be used with xCAT data object definition commands.

## DESCRIPTION

A stanza file contains information that can be used to create xCAT data object definitions. A stanza file can be used as input to several xCAT commands. The stanza file contains one or more individual stanzas that provide information for individual object definitions as well as an optional default definition that applies to all subsequent object definitions of that type.

The following rules must be followed when creating a stanza file:

- \* An object stanza header consists of the object name followed by a colon, (“:”).
- \* Attribute lines must take the form of Attribute=Value.
- \* Attribute name might include the character dot (“.”), like passwd.HMC and nicips.eth0.
- \* Only one stanza can exist for each object name.
- \* All stanzas except for default stanzas must have a value set for “objtype”.
- \* Comments beginning with the “#” pound sign may be added to the file. A comment must be on a separate line.
- \* When parsing the file, tab characters and spaces are ignored.
- \* Each line of the file can have no more than one header or attribute definition.
- \* If the header name is “default-<object type>:” the attribute values in the stanza are considered default values for subsequent definitions in the file that are the same object type.
- \* Default stanzas can be specified multiple times and at any point in a stanza file. The values apply to all definitions following the default stanzas in a file. The default values are cumulative; a default attribute value will remain set until it is explicitly unset or changed.
- \* To turn off a default value, use another default stanza to set the attribute to have no value using a blank space.
- \* When a specific value for an attribute is provided in the stanza, it takes priority over any default value that had been set.

The format of a stanza file should look similar to the following.

```
default-<object type>:
    attr=val
    attr=val
    . . .

<object name>:
    objtype=<object type>
    attr=val
    attr=val
```

(continues on next page)

(continued from previous page)

```
. . .  
<object name>:  
  objtype=<object type>  
  attr=val  
  attr=val  
. . .
```

## EXAMPLES

1) Sample stanza file:

```
mysite:  
  objtype=site  
  rsh=/bin/rsh  
  rcp=/bin/rcp  
  installdir=/xcatinstall  
  domain=ppd.pok.ibm.com  
  
MSnet01:  
  objtype=network  
  gateway=1.2.3.4  
  netmask=255.255.255.0  
  nameserver=5.6.7.8  
  
default-node:  
  next_osimage=aix61  
  network=MSnet01  
  groups=all,compute  
  
node01:  
  objtype=node  
  MAC=A2E26002C003  
  xcatmaster=MS02.ppd.pok.com  
  nfsserver=IS227.ppd.pok.com  
  
node02:  
  objtype=node  
  MAC=A2E26002B004  
  xcatmaster=MS01.ppd.pok.com  
  nfsserver=IS127.ppd.pok.com  
  
grp01:  
  objtype=group  
  members=node1,node2,node3
```



## NOTES

This file is part of xCAT software product.

## SEE ALSO

mkdef(1)|mkdef.1, lsdef(1)|lsdef.1, rmdef(1)|rmdef.1, chdef(1)|chdef.1

## zone.5

## NAME

**zone** - a table in the xCAT database.

## SYNOPSIS

**zone Attributes:** *zonename, sshkeydir, sshbetweennodes, defaultzone, comments, disable*

## DESCRIPTION

Defines a cluster zone for nodes that share root ssh key access to each other.

### zone Attributes:

#### zonename

The name of the zone.

#### sshkeydir

Directory containing the shared root ssh RSA keys.

#### sshbetweennodes

Indicates whether passwordless ssh will be setup between the nodes of this zone. Values are yes/1 or no/0. Default is yes.

#### defaultzone

If nodes are not assigned to any other zone, they will default to this zone. If value is set to yes or 1.

#### comments

Any user-provided notes.

#### disable

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## zvm.5

## NAME

**zvm** - a table in the xCAT database.

## SYNOPSIS

**zvm Attributes:** *node, hcp, userid, nodetype, parent, comments, disable, discovered, status*

## DESCRIPTION

List of z/VM virtual servers.

### zvm Attributes:

#### **node**

The node name.

#### **hcp**

The hardware control point for this node.

#### **userid**

The z/VM userID of this node.

#### **nodetype**

The node type. Valid values: cec (Central Electronic Complex), lpar (logical partition), zvm (z/VM host operating system), and vm (virtual machine).

#### **parent**

The parent node. For LPAR, this specifies the CEC. For z/VM, this specifies the LPAR. For VM, this specifies the z/VM host operating system.

#### **comments**

Any user provided notes.

#### **disable**

Set to 'yes' or '1' to comment out this row.

#### **discovered**

Set to '1' to indicate this node was discovered.

#### **status**

The processing status. Key value pairs (key=value) indicating status of the node. Multiple pairs are separated by semi-colons. Keys include: CLONING, CLONE\_ONLY.

## SEE ALSO

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

## zvmivp.5

## NAME

**zvmivp** - a table in the xCAT database.

## SYNOPSIS

**zvmivp Attributes:** *id, ip, schedule, last\_run, type\_of\_run, access\_user, orch\_parms, prep\_parms, main\_ivp\_parms, comments, disable*

## DESCRIPTION

List of z/VM Installation Verification Procedures (IVPs) to be periodically run.

### zvmivp Attributes:

#### **id**

Unique identifier associated with the IVP run, e.g. 1.

#### **ip**

IP address of the target system, either the IP of the OpenStack compute node or the xCAT management node.

#### **schedule**

The hours (0-24) that the IVP should be run. Multiple hours are separated by a blank.

#### **last\_run**

The last time the IVP was run specified as a set of 3 blank delimited words: year, Julian date, and hour (in 24 hour format).

#### **type\_of\_run**

The type of run requested, 'fullivp' or 'basicivp'.

#### **access\_user**

User on the OpenStack node that is used to: push the IVP preparation script to the OpenStack system, drive the preparation script to validate the OpenStack configuration files, and return the created driver script to the xCAT MN system for the next part of the IVP. This user should be able to access the OpenStack configuration files that are scanned by the IVP.

#### **orch\_parms**

Parameters to pass to the IVP orchestrator script, `verifynode`.

#### **prep\_parms**

Parameters to pass to the phase 1 IVP preparation script.

### **main\_ivp\_parms**

Parameters to pass to the main IVP script.

### **comments**

Any user provided notes or description of the run.

### **disable**

Set to 'yes' or '1' to disable this IVP run.

## **SEE ALSO**

**models(1)**, **chtab(8)**, **tabdump(8)**, **tabedit(8)**

### **man7**

### **auditlog.7**

## **NAME**

**auditlog** - a logical object definition in the xCAT database.

## **SYNOPSIS**

**auditlog Attributes:** *args, audittime, clientname, clienttype, command, comments, disable, noderange, recid, status, userid*

## **DESCRIPTION**

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### **auditlog Attributes:**

**args** (auditlog.args)

The command argument list.

**audittime** (auditlog.audittime)

The timestamp for the audit entry.

**clientname** (auditlog.clientname)

The client machine, where the command originated.

**clienttype** (auditlog.clienttype)

Type of command: cli, java, webui, other.

**command** (auditlog.command)

Command executed. See `auditskipcmds` site table attribute to control which commands get logged.

**comments** (auditlog.comments)

Any user-provided notes.

**disable** (auditlog.disable)

Do not use. `tabprune` will not work if set to yes or 1

**noderange** (auditlog.noderange)

The noderange on which the command was run.

**recid** (auditlog.recid)

The record id.

**status** (auditlog.status)

Allowed or Denied.

**userid** (auditlog.userid)

The user running the command.

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## boottarget.7

## NAME

**boottarget** - a logical object definition in the xCAT database.

## SYNOPSIS

**boottarget Attributes:** *bprofile, comments, initrd, kcmdline, kernel*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### boottarget Attributes:

#### **bprofile** (boottarget.bprofile)

All nodes with a nodetype.profile value equal to this value and nodetype.os set to “boottarget”, will use the associated kernel, initrd, and kcmdline.

#### **comments** (boottarget.comments)

Any user-written notes.

#### **initrd** (boottarget.initrd)

The initial ramdisk image that network boot actions should use (could be a DOS floppy or hard drive image if using memdisk as kernel)

#### **kcmdline** (boottarget.kcmdline)

Arguments to be passed to the kernel

#### **kernel** (boottarget.kernel)

The kernel that network boot actions should currently acquire and use. Note this could be a chained boot loader such as memdisk or a non-linux boot loader

### SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

### eventlog.7

### NAME

**eventlog** - a logical object definition in the xCAT database.

### SYNOPSIS

**eventlog Attributes:** *application, comments, component, disable, eventtime, eventtype, id, message, monitor, monnode, node, rawdata, recid, severity*

### DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

## eventlog Attributes:

**application** (eventlog.application)

The application that reports the event.

**comments** (eventlog.comments)

Any user-provided notes.

**component** (eventlog.component)

The component where the event occurred.

**disable** (eventlog.disable)

Do not use. tabprune will not work if set to yes or 1

**eventtime** (eventlog.eventtime)

The timestamp for the event.

**eventtype** (eventlog.eventtype)

The type of the event.

**id** (eventlog.id)

The location or the resource name where the event occurred.

**message** (eventlog.message)

The full description of the event.

**monitor** (eventlog.monitor)

The name of the monitor that monitors this event.

**monnode** (eventlog.monnode)

The node that monitors this event.

**node** (eventlog.node)

The node where the event occurred.

**rawdata** (eventlog.rawdata)

The data that associated with the event.

**recid** (eventlog.recid)

The record id.

**severity** (eventlog.severity)

The severity of the event. Valid values are: informational, warning, critical.

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## firmware.7

### NAME

**firmware** - a logical object definition in the xCAT database.

### SYNOPSIS

**firmware Attributes:** *cfgfile*, *comments*, *disable*

### DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

#### firmware Attributes:

**cfgfile** (firmware.cfgfile)

The file to use.

**comments** (firmware.comments)

Any user-written notes.

**disable** (firmware.disable)

Set to 'yes' or '1' to comment out this row.

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## group.7

### NAME

**group** - a logical object definition in the xCAT database.



## SYNOPSIS

**group Attributes:** *addkcmdline, arch, authdomain, authkey, authtype, bmc, bmcpassword, bmcport, bmcusername, bmcvlan, cfmgr, cfmgtroles, cfgserver, chain, chassis, cmdmapping, community, cons, conserver, consoleenabled, consoleondemand, consport, cpucount, cputype, currchain, currstate, dhcpinterfaces, disksize, displayname, dockercpus, dockerflag, dockerhost, dockermemory, dockernics, domainadminpassword, domainadminuser, domain-type, getmac, groupname, grouptype, hcp, height, hostcluster, hostinterface, hostmanager, hostnames, hosttype, hwtype, id, initrd, installnic, interface, ip, iscsipassword, iscsiserver, iscsitarget, iscsiuserid, kcmdline, kernel, linkports, mac, membergroups, members, memory, mgt, micbridge, michost, micid, miconboot, micpowermgmt, micvlog, migrationdest, monserver, mpa, mtm, nameservers, netboot, nfsdir, nfsserver, nicaliases, niccustomscrips, nicdevices, nicextraparams, nichostnameprefixes, nichostnamesuffixes, nicips, nicnetworks, nicsadapter, nictypes, nimserver, node-type, ondiscover, os, osvolumes, otherinterfaces, ou, outlet, parent, passwd.HMC, passwd.admin, passwd.celogin, passwd.general, passwd.hscroot, password, pdu, pdutype, postbootscrips, postscripts, power, pprofile, prescripts-begin, prescripts-end, primarynic, privkey, privtype, productkey, profile, protocol, provmethod, rack, room, route-names, seclevel, serial, serialflow, serialport, serialspeed, servicenode, setupconserver, setupdhcp, setupftp, setupip-forward, setupldap, setupnameserver, setupnfs, setupnim, setupntp, setupproxymdhcp, setupftp, sfp, side, slot, slotid, slots, snmpauth, snmppassword, snmpprivacy, snmpuser, snmpusername, snmpversion, storagcontroller, storagetype, supernode, supportedarchs, supportproxymdhcp, switch, switchinterface, switchport, switchtype, switchvlan, syslog, termport, termserver, tfpdir, tfpsrv, unit, urlpath, usercomment, userid, username, vmbeacon, vmbootorder, vm-cfgstore, vmcluster, vmcpus, vmhost, vmmanager, vmmaster, vmmemory, vmnicnicmodel, vmnics, vmothersetting, vm-physlots, vmstorage, vmstoragecache, vmstorageformat, vmstoragemodel, vmtextconsole, vmvirtflags, vmvncport, web-port, wherevals, xcatmaster*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### group Attributes:

#### **addkcmdline** (bootparams.addkcmdline)

User specified kernel options for os provision process (no prefix) or the provisioned os (with prefix "R:"). Multiple options should be delimited with spaces(" ") and surrounded with quotes. To have the same option used for os provision process and for provisioned os, specify that option with and without the prefix: `addkcmdline="R::display=3 display=3"`

#### **arch** (nodetype.arch)

The hardware architecture of this node. Valid values: x86\_64, ppc64, x86, ia64.

#### **authdomain** (domain.authdomain)

If a node should participate in an AD domain or Kerberos realm distinct from domain indicated in site, this field can be used to specify that

#### **authkey** (pdu.authkey)

The authentication passphrase for SNMPv3

#### **authtype** (pdu.authtype)

The authentication protocol(MD5|SHA) to use for SNMPv3.

#### **bmc** (ipmi.bmc, openbmc.bmc)

The hostname of the BMC adapter.

or

The hostname of the BMC adapter.

**bmcpassword** (ipmi.password, openbmc.password)

The BMC password. If not specified, the key=ipmi row in the passwd table is used as the default.

or

The BMC password. If not specified, the key=openbmc row in the passwd table is used as the default.

**bmcport** (ipmi.bmcport)

In systems with selectable shared/dedicated ethernet ports, this parameter can be used to specify the preferred port. 0 means use the shared port, 1 means dedicated, blank is to not assign.

The following special cases exist for IBM System x servers:

For x3755 M3 systems, 0 means use the dedicated port, 1 means shared, blank is to not assign.

For certain systems which have a mezzanine or ML2 adapter, there is a second value to include:

For x3750 M4 (Model 8722):

0 2 1st 1Gbps interface for LOM

0 0 1st 10Gbps interface for LOM

0 3 2nd 1Gbps interface for LOM

0 1 2nd 10Gbps interface for LOM

For x3750 M4 (Model 8752), x3850/3950 X6, dx360 M4, x3550 M4, and x3650 M4:

0 Shared (1st onboard interface)

1 Dedicated

2 0 First interface on ML2 or mezzanine adapter

2 1 Second interface on ML2 or mezzanine adapter

2 2 Third interface on ML2 or mezzanine adapter

2 3 Fourth interface on ML2 or mezzanine adapter

**bmcusername** (ipmi.username, openbmc.username)

The BMC userid. If not specified, the key=ipmi row in the passwd table is used as the default.

or

The BMC userid. If not specified, the key=openbmc row in the passwd table is used as the default.

**bmcvlan** (ipmi.taggedvlan, openbmc.taggedvlan)

bmcsetup script will configure the network interface of the BMC to be tagged to the VLAN specified.

or

bmcsetup script will configure the network interface of the BMC to be tagged to the VLAN specified.

**cfgmgr** (cfgmgt.cfgmgr)

The name of the configuration manager service. Currently 'chef' and 'puppet' are supported services.

**cfgmgtroles** (cfgmgt.roles)

The roles associated with this node as recognized by the cfgmgr for the software that is to be installed and configured. These role names map to chef recipes or puppet manifest classes that should be used for this node. For example, chef OpenStack cookbooks have roles such as mysql-master,keystone, glance, nova-controller, nova-conductor, cinder-all.

**cfgserver** (cfgmgt.cfgserver)

The xCAT node name of the chef server or puppet master

**chain** (chain.chain)

A comma-delimited chain of actions to be performed automatically when this node is discovered for the first time. (xCAT and the DHCP server do not recognize the MAC address of the node when xCAT initializes the discovery process.) The last step in this process is to run the operations listed in the chain attribute, one by one. Valid values: boot, runcmd=<cmd>, runimage=<URL>, shell, standby. For example, to have the genesis kernel pause to the shell, use chain=shell.

**chassis** (nodepos.chassis)

The BladeCenter chassis the blade is in.

**cmdmapping** (nodehm.cmdmapping)

The fully qualified name of the file that stores the mapping between PCM hardware management commands and xCAT/third-party hardware management commands for a particular type of hardware device. Only used by PCM.

**community** (pdu.community)

The community string to use for SNMPv1/v2

**cons** (nodehm.cons)

The console method. If nodehm.serialport is set, this will default to the nodehm.mgt setting, otherwise it defaults to unused. Valid values: cyclades, mrv, or the values valid for the mgt attribute.

**conserver** (nodehm.conserver)

The hostname of the machine where the conserver daemon is running. If not set, the default is the xCAT management node.

**consoleenabled** (nodehm.consoleenabled)

A flag field to indicate whether the node is registered in the console server. If '1', console is enabled, if not set, console is not enabled.

**consoleondemand** (nodehm.consoleondemand)

This overrides the value from `site.consoleondemand`. Set to 'yes', 'no', '1' (equivalent to 'yes'), or '0' (equivalent to 'no'). If not set, the default is the value from `site.consoleondemand`.

**consport** (`openbmc.consport`)

The console port for OpenBMC.

**cpucount** (`hwinv.cpucount`)

The number of cpus for the node.

**cputype** (`hwinv.cputype`)

The cpu model name for the node.

**currchain** (`chain.currchain`)

The chain steps still left to do for this node. This attribute will be automatically adjusted by xCAT while xCAT-genesis is running on the node (either during node discovery or a special operation like firmware update). During node discovery, this attribute is initialized from the chain attribute and updated as the chain steps are executed.

**currstate** (`chain.currstate`)

The current or next chain step to be executed on this node by xCAT-genesis. Set by xCAT during node discovery or as a result of `nodeset`.

**dhcpinterfaces** (`servicenode.dhcpinterfaces`)

The network interfaces DHCP server should listen on for the target node. This attribute can be used for management node and service nodes. If defined, it will override the values defined in `site.dhcpinterfaces`. This is a comma separated list of device names. `!remote!` indicates a non-local network for relay DHCP. For example: `!remote!,eth0,eth1`

**disksize** (`hwinv.disksize`)

The size of the disks for the node in GB.

**displayname** (`mpa.displayname`)

Alternative name for BladeCenter chassis. Only used by PCM.

**dockercpus** (`vm.cpus`)

Number of CPUs the node should see.

**dockerflag** (`vm.othersettings`)

**This is a semicolon-delimited list of key-value pairs to be included in a vmx file of VMware or KVM. DO NOT use 'chdef <node> -p|-m vmothersetting=...' to add options to it or delete options from it because chdef uses commas, not semicolons, to separate items.**

**Hugepage on POWER systems:**

Specify the hugepage and/or bsr (Barrier Synchronization Register) values, e.g., `'hugepage:1,bsr:2'`.

**KVM CPU mode:**

Specify how the host CPUs are utilized, e.g., `'cpumode:host-passthrough'`, `'cpumode:host-model'`. With the passthrough mode, the performance of x86 VMs can be improved significantly.

**KVM CPU pinning:**

Specify which host CPUs are used, e.g., `'vcupin:'0-15,^8'`, where '-' denotes the range and '^' denotes exclusion. This option allows a comma-delimited list.

#### KVM memory binding:

Specify which nodes that host memory are used, e.g., 'membind:0', where the memory in node0 of the hypervisor is used. /sys/devices/system/node has node0 and node8 on some POWER systems, node0 and node1 on some x86\_64 systems. This option allows a guest VM to access specific memory regions.

#### PCI passthrough:

PCI devices can be assigned to a virtual machine for exclusive usage, e.g., 'dev-passthrough:pci\_0001\_01\_00\_0,pci\_0000\_03\_00\_0'. A PCI device can also be expressed as 'devpassthrough:0001:01:00.1'. The devices are put in a comma-delimited list. The PCI device names can be obtained by running **virsh nodedev-list** on the host.

#### VM machine type:

Specify a machine type for VM creation on the host, e.g., 'machine:pc'. Typical machine types are pc, q35, and pseries.

#### **dockerhost** (vm.host)

The system that currently hosts the VM

#### **dockermemory** (vm.memory)

Megabytes of memory the VM currently should be set to.

#### **dockernics** (vm.nics)

Network configuration parameters. Of the general form [physnet:]interface,.. Generally, interface describes the vlan entity (default for native, tagged for tagged, vl[number] for a specific vlan. physnet is a virtual switch name or port description that is used for some virtualization technologies to construct virtual switches. hypervisor.netmap can map names to hypervisor specific layouts, or the descriptions described there may be used directly here where possible.

#### **domainadminpassword** (domain.adminpassword)

Allow a node specific indication of Administrative user password for the domain. Most will want to ignore this in favor of passwd table.

#### **domainadminuser** (domain.adminuser)

Allow a node specific indication of Administrative user. Most will want to just use passwd table to indicate this once rather than by node.

#### **domaintype** (domain.type)

Type, if any, of authentication domain to manipulate. The only recognized value at the moment is activedirectory.

#### **getmac** (nodehm.getmac)

The method to use to get MAC address of the node with the getmac command. If not set, the mgt attribute will be used. Valid values: same as values for mgmt attribute.

#### **groupname** (nodegroup.groupname)

Name of the group.

#### **grouptype** (nodegroup.grouptype)

Static or Dynamic. A static group is defined to contain a specific set of cluster nodes. A dynamic node group is one that has its members determined by specifying a selection criteria for node attributes.

#### **hcp** (ppc.hcp, zvm.hcp)

The hardware control point for this node (HMC, IVM, Frame or CEC). Do not need to set for BPAs and FSPs.

or

The hardware control point for this node.

**height** (nodepos.height)

The server height in U(s).

**hostcluster** (hypervisor.cluster)

Specify to the underlying virtualization infrastructure a cluster membership for the hypervisor.

**hostinterface** (hypervisor.interface)

The definition of interfaces for the hypervisor. The format is [network-name:interfacename:bootprotocol:IP:netmask:gateway] that split with | for each interface

**hostmanager** (hypervisor.mgr)

The virtualization specific manager of this hypervisor when applicable

**hostnames** (hosts.hostnames)

Hostname aliases added to /etc/hosts for this node. Comma or blank separated list.

**hosttype** (hypervisor.type)

The plugin associated with hypervisor specific commands such as revacuate

**hwtype** (ppc.nodetype, zvm.nodetype, mp.nodetype, mic.nodetype)

The hardware type of the node. Only can be one of fsp, bpa, cec, frame, ivm, hmc and lpar

or

The node type. Valid values: cec (Central Electronic Complex), lpar (logical partition), zvm (z/VM host operating system), and vm (virtual machine).

or

The hardware type for mp node. Valid values: mm,cmm, blade.

or

The hardware type of the mic node. Generally, it is mic.

**id** (ppc.id, mp.id)

For LPARs: the LPAR numeric id; for CECs: the cage number; for Frames: the frame number.

or

The slot number of this blade in the BladeCenter chassis.

**initrd** (bootparams.initrd)

The initial ramdisk image that network boot actions should use (could be a DOS floppy or hard drive image if using memdisk as kernel)

**installnic** (noderes.installnic)

The network adapter on the node that will be used for OS deployment, the installnic can be set to the network adapter name or the mac address or the keyword "mac" which means that the network interface specified by the mac address in the mac table will be used. If not set, primarynic will be used. If primarynic is not set too, the keyword "mac" will be used as default.

**interface** (mac.interface)

The adapter interface name that will be used to install and manage the node. E.g. eth0 (for linux) or en0 (for AIX).)

**ip** (hosts.ip)

The IP address of the node. This is only used in makehosts. The rest of xCAT uses system name resolution to resolve node names to IP addresses.

**iscsipassword** (iscsi.passwd)

The password for the iscsi server containing the boot device for this node.

**iscsiserver** (iscsi.server)

The server containing the iscsi boot device for this node.

**iscsitarget** (iscsi.target)

The iscsi disk used for the boot device for this node. Filled in by xCAT.

**iscsiuserid** (iscsi.userid)

The userid of the iscsi server containing the boot device for this node.

**kcmdline** (bootparams.kcmdline)

(Deprecated, use addkcmdline instead) Arguments to be passed to the kernel.

**kernel** (bootparams.kernel)

The kernel that network boot actions should currently acquire and use. Note this could be a chained boot loader such as memdisk or a non-linux boot loader

**linkports** (switches.linkports)

The ports that connect to other switches. Currently, this column is only used by vlan configuration. The format is: "port\_number:switch,port\_number:switch...". Refer to the switch table for details on how to specify the port numbers.

**mac** (mac.mac)

The mac address or addresses for which xCAT will manage static bindings for this node. This may be simply a mac address, which would be bound to the node name (such as "01:02:03:04:05:0E"). This may also be a "!" delimited string of "mac address!hostname" format (such as "01:02:03:04:05:0E!node5|01:02:03:04:05:0F!node6-eth1"). If there are multiple nics connected to Management Network(usually for bond), in order to make sure the OS deployment finished successfully, the macs of those nics must be able to resolve to same IP address. First, users have to create alias of the node for each mac in the Management Network through either: 1. adding the alias into /etc/hosts for the node directly or: 2. setting the alias to the "hostnames" attribute and then run "makehost" against the node. Then, configure the "mac" attribute of the node like "mac1!node|mac2!node-alias". For the first mac address (mac1 in the example) set in "mac" attribute, do not need to set a "node name" string for it since the nodename of the node will be used for it by default.

**membergroups** (nodegroup.membergroups)

This attribute stores a comma-separated list of nodegroups that this nodegroup refers to. This attribute is only used by PCM.

**members** (nodegroup.members)

The value of the attribute is not used, but the attribute is necessary as a place holder for the object def commands. (The membership for static groups is stored in the nodelist table.)

**memory** (hwinv.memory)

The size of the memory for the node in MB.

**mgt** (nodehm.mgt)

The method to use to do general hardware management of the node. This attribute is used as the default if power or getmac is not set. Valid values: openbmc, ipmi, blade, hmc, ivm, fsp, bpa, kvm, esx, rhevm. See the power attribute for more details.

**micbridge** (mic.bridge)

The virtual bridge on the host node which the mic connected to.

**michost** (mic.host)

The host node which the mic card installed on.

**micid** (mic.id)

The device id of the mic node.

**miconboot** (mic.onboot)

Set mic to autoboot when mpss start. Valid values: yes|no. Default is yes.

**micpowermgt** (mic.powermgt)

Set the Power Management for mic node. This attribute is used to set the power management state that mic may get into when it is idle. Four states can be set: cpufreq, corec6, pc3 and pc6. The valid value for powermgt attribute should be [cpufreq=<on|off>]![corec6=<on|off>]![pc3=<on|off>]![pc6=<on|off>]. e.g. cpufreq=on!corec6=off!pc3=on!pc6=off. Refer to the doc of mic to get more information for power management.

**micvlog** (mic.vlog)

Set the Verbose Log to console. Valid values: yes|no. Default is no.

**migrationdest** (vm.migrationdest)

A nodestring representing candidate destinations for migration (i.e. similar systems, same SAN, or other criteria that xCAT can use

**monserver** (noderes.monserver)

The monitoring aggregation point for this node. The format is “x,y” where x is the ip address as known by the management node and y is the ip address as known by the node.

**mpa** (mp.mpa)

The management module used to control this blade.

**mtm** (vpd.mtm)

The machine type and model number of the node. E.g. 7984-6BU

**nameservers** (noderes.nameservers)

An optional node/group specific override for name server list. Most people want to stick to site or network defined nameserver configuration.

**netboot** (noderes.netboot)

The type of network booting to use for this node. Valid values:

Arch	OS	valid netboot options
x86, x86_64	ALL	pxe, xnba
ppc64	<=rhel6, <=sles11.3	yaboot
ppc64	>=rhels7, >=sles11.4	grub2, grub2-http, grub2-
↪ tftp		

(continues on next page)



(continued from previous page)

ppc64le NonVirtualize ALL	petitboot
ppc64le PowerKVM Guest ALL	grub2,grub2-http,grub2-
→tftp	

**nfsdir** (noderes.nfsdir)

The path that should be mounted from the NFS server.

**nfsserver** (noderes.nfsserver)

The NFS or HTTP server for this node (as known by this node).

**nicaliases** (nics.nicaliases)

**Comma-separated list of hostname aliases for each NIC.**

**Format:** eth0!<alias list>,eth1!<alias1 list>|<alias2 list>

For multiple aliases per nic use a space-separated list.

For example: eth0!moe larry curly,eth1!tom|jerry

**niccustomscripts** (nics.niccustomscripts)

Comma-separated list of custom scripts per NIC. <nic1>!<script1>,<nic2>!<script2>, e.g. eth0!configeth eth0, ib0!configib ib0. The xCAT object definition commands support to use niccustomscripts.<nicname> as the sub attribute .

**nicdevices** (nics.nicdevices)

Comma-separated list of NIC device per NIC, multiple ethernet devices can be bonded as bond device, these ethernet devices are separated by | . <nic1>!<dev1>|<dev3>,<nic2>!<dev2>, e.g. bond0!eth0|eth2,br0!bond0. The xCAT object definition commands support to use nicdevices.<nicname> as the sub attributes.

**nicextraparams** (nics.nicextraparams)

**Comma-separated list of extra parameters that will be used for each NIC configuration.**

**If only one ip address is associated with each NIC:**

<nic1>!<param1=value1 param2=value2>,<nic2>!<param3=value3>, for example, eth0!MTU=1500,ib0!MTU=65520 CONNECTED\_MODE=yes.

**If multiple ip addresses are associated with each NIC:**

<nic1>!<param1=value1 param2=value2>|<param3=value3>,<nic2>!<param4=value4 param5=value5>|<param6=value6>, for example, eth0!MTU=1500|MTU=1460,ib0!MTU=65520 CONNECTED\_MODE=yes.

**The semicolon separator is needed if there are multiple values for extra parameters:**

bond0!BONDING\_OPTS=lacp\_rate=1;miimon=100;mode=802.3ad

The xCAT object definition commands support to use nicextraparams.<nicname> as the sub attributes.

**nichostnameprefixes** (nics.nichostnameprefixes)

**Comma-separated list of hostname prefixes per NIC.**

**If only one ip address is associated with each NIC:**

<nic1>!<ext1>,<nic2>!<ext2>,..., for example, eth0!eth0-,ib0!ib-

**If multiple ip addresses are associated with each NIC:**

<nic1>!<ext1>|<ext2>,<nic2>!<ext1>|<ext2>,..., for example, eth0!eth0-|eth0-ipv6i-,ib0!ib-|ib-ipv6-.

The xCAT object definition commands support to use `nichostnameprefixes.<nicname>` as the sub attributes. Note: According to DNS rules a hostname must be a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9) and minus sign (-). When you are specifying “`nichostnameprefixes`” or “`nicaliases`” make sure the resulting hostnames will conform to this naming convention

**nichostnamesuffixes** (`nics.nichostnamesuffixes`)

**Comma-separated list of hostname suffixes per NIC.**

**If only one ip address is associated with each NIC:**

`<nic1>|<ext1>,<nic2>|<ext2>,...`, for example, `eth0!-eth0,ib0!-ib0`

**If multiple ip addresses are associated with each NIC:**

`<nic1>|<ext1>|<ext2>,<nic2>|<ext1>|<ext2>,...`, for example, `eth0!-eth0|eth0-ipv6,ib0!-ib0|ib0-ipv6`.

The xCAT object definition commands support to use `nichostnamesuffixes.<nicname>` as the sub attributes.

Note: According to DNS rules a hostname must be a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9) and minus sign (-). When you are specifying “`nichostnamesuffixes`” or “`nicaliases`” make sure the resulting hostnames will conform to this naming convention

**nicips** (`nics.nicips`)

**Comma-separated list of IP addresses per NIC.**

**To specify one ip address per NIC:**

`<nic1>|<ip1>,<nic2>|<ip2>,...`, for example, `eth0!10.0.0.100,ib0!11.0.0.100`

**To specify multiple ip addresses per NIC:**

`<nic1>|<ip1>|<ip2>,<nic2>|<ip1>|<ip2>,...`, for example, `eth0!10.0.0.100|fd55::214:5eff:fe15:849b,ib0!11.0.0.100|2001:db8::1`

The xCAT object definition commands support to use `nicips.<nicname>` as the sub attributes.

Note: The primary IP address must also be stored in the `hosts.ip` attribute. The `nichostnamesuffixes` should specify one hostname suffix for each ip address.

**nicnetworks** (`nics.nicnetworks`)

**Comma-separated list of networks connected to each NIC.**

**If only one ip address is associated with each NIC:**

`<nic1>|<network1>,<nic2>|<network2>,...`, for example, `eth0!10_0_0_0-255_255_0_0,ib0!11_0_0_0-255_255_0_0`

**If multiple ip addresses are associated with each NIC:**

`<nic1>|<network1>|<network2>,<nic2>|<network1>|<network2>,...`, for example, `eth0!10_0_0_0-255_255_0_0|fd55:faaf:e1ab:336::/64,ib0!11_0_0_0-255_255_0_0|2001:db8:1:0::/64`. The xCAT object definition commands support to use `nicnetworks.<nicname>` as the sub attributes.

**nicadapter** (`nics.nicsadapter`)

Comma-separated list of NIC information collected by `getadapter`. `<nic1>|<param1=value1 param2=value2>,<nic2>|<param4=value4 param5=value5>,...`, for example, `enP3p3s0f1!mac=98:be:94:59:fa:cd linkstate=DOWN,enP3p3s0f2!mac=98:be:94:59:fa:ce candidate-name=enP3p3s0f2/enx98be9459face`

**nictypes** (`nics.nictypes`)

Comma-separated list of NIC types per NIC. <nic1>!<type1>,<nic2>!<type2>, e.g. eth0!Ethernet,ib0!Infiniband. The xCAT object definition commands support to use nictypes.<nicname> as the sub attributes.

#### **nimserver** (noderes.nimserver)

Not used for now. The NIM server for this node (as known by this node).

#### **nodetype** (nodetype.nodetype, pdu.nodetype)

A comma-delimited list of characteristics of this node. Valid values: ppc, blade, vm (virtual machine), osi (OS image), mm, mn, rsa, switch.

or

The node type should be pdu

#### **ondiscover** (chain.ondiscover)

This attribute is currently not used by xCAT. The “nodediscover” operation is always done during node discovery.

#### **os** (nodetype.os)

The operating system deployed on this node. Valid values: AIX, rhels\*, rhelc\*, rhas\*, centos\*, alma\*, rocky\*, SL\*, fedora\*, sles\* (where \* is the version #). As a special case, if this is set to “boottarget”, then it will use the initrd/kernel/parameters specified in the row in the boottarget table in which boottarget.bprofile equals nodetype.profile.

#### **osvolume** (storage.osvolume)

Specification of what storage to place the node OS image onto. Examples include:

```
localdisk (Install to first non-FC attached disk)
usbdisk (Install to first USB mass storage device seen)
wwn=0x50000393c813840c (Install to storage device with given WWN)
```

#### **otherinterfaces** (hosts.otherinterfaces)

Other IP addresses to add for this node. Format: -<ext>:<ip>,<intfhostname>:<ip>,...

#### **ou** (domain.ou)

For an LDAP described machine account (i.e. Active Directory), the organizational unit to place the system. If not set, defaults to cn=Computers,dc=your,dc=domain

#### **outlet** (pdu.outlet)

The pdu outlet count

#### **parent** (ppc.parent)

For LPARs: the CEC; for FSPs: the CEC; for CEC: the frame (if one exists); for BPA: the frame; for frame: the building block number (which consists 1 or more service nodes and compute/storage nodes that are serviced by them - optional).

#### **passwd.HMC** (ppcdirect.password)

Password of the FSP/BPA(for ASMI) and CEC/Frame(for DFM). If not filled in, xCAT will look in the passwd table for key=fsp. If not in the passwd table, the default used is admin.

#### **passwd.admin** (ppcdirect.password)

Password of the FSP/BPA(for ASMI) and CEC/Frame(for DFM). If not filled in, xCAT will look in the passwd table for key=fsp. If not in the passwd table, the default used is admin.

**passwd.celogin** (ppcdirect.password)

Password of the FSP/BPA(for ASMI) and CEC/Frame(for DFM). If not filled in, xCAT will look in the passwd table for key=fsp. If not in the passwd table, the default used is admin.

**passwd.general** (ppcdirect.password)

Password of the FSP/BPA(for ASMI) and CEC/Frame(for DFM). If not filled in, xCAT will look in the passwd table for key=fsp. If not in the passwd table, the default used is admin.

**passwd.hscroot** (ppcdirect.password)

Password of the FSP/BPA(for ASMI) and CEC/Frame(for DFM). If not filled in, xCAT will look in the passwd table for key=fsp. If not in the passwd table, the default used is admin.

**password** (ppchcp.password, mpa.password, websrv.password, pdu.password, switches.sshpassword)

Password of the HMC or IVM. If not filled in, xCAT will look in the passwd table for key=hmc or key=ivm. If not in the passwd table, the default used is abc123 for HMCs and padmin for IVMs.

or

Password to use to access the management module. If not specified, the key=blade row in the passwd table is used as the default.

or

Password to use to access the web service.

or

The remote login password

or

The remote login password. It can be for ssh or telnet. If it is for telnet, set protocol to “telnet”. If the sshusername is blank, the username, password and protocol will be retrieved from the passwd table with “switch” as the key.

**pdu** (pduoutlet.pdu)

a comma-separated list of outlet number for each PDU, ex: pdu1:outlet1,pdu2:outlet1

**pdu type** (pdu.pdu type)

The type of pdu

**postbootscripts** (postscripts.postbootscripts)

Comma separated list of scripts that should be run on this node after diskful installation or diskless boot. Each script can take zero or more parameters. For example: “script1 p1 p2,script2,...”. On AIX these scripts are run during the processing of /etc/inittab. On Linux they are run at the init.d time. xCAT automatically adds the scripts in the xcatdefaults.postbootscripts attribute to run first in the list. Please note that the postbootscripts specified for “xcatdefaults” will be assigned to node automatically, they can not be removed from “postbootscripts” attribute of a node with “chdef -m” command

**postscripts** (postscripts.postscripts)

Comma separated list of scripts that should be run on this node after diskful installation or diskless boot. Each script can take zero or more parameters. For example: “script1 p1 p2,script2,...”. xCAT automatically adds the postscripts from the xcatdefaults.postscripts attribute of the table to run first on the nodes after install or diskless boot. For installation of RedHat, CentOS, Fedora, the scripts will be run before the reboot. For installation of SLES, the scripts will be run after the reboot but before the init.d process. For diskless deployment, the scripts will be run at the init.d time, and xCAT will automatically add the list of scripts from the postbootscripts attribute to run after postscripts list. For installation of AIX, the

scripts will run after the reboot and acts the same as the postbootscripts attribute. For AIX, use the postbootscripts attribute. Please note that the postscripts specified for “xcatdefaults” will be assigned to node automatically, they can not be removed from “postscripts” attribute of a node with “chdef -m” command

#### **power** (nodehm.power)

The method to use to control the power of the node. If not set, the mgt attribute will be used. Valid values: ipmi, blade, hmc, ivm, fsp, kvm, esx, rhevm. If “ipmi”, xCAT will search for this node in the ipmi table for more info. If “blade”, xCAT will search for this node in the mp table. If “hmc”, “ivm”, or “fsp”, xCAT will search for this node in the ppc table.

#### **pprofile** (ppc.pprofile)

The LPAR profile that will be used the next time the LPAR is powered on with rpower. For DFM, the pprofile attribute should be set to blank

#### **prescripts-begin** (prescripts.begin)

**The scripts to be run at the beginning of the nodeset(Linux), nimnodeset(AIX) or mkdsklsnode(AIX) command.**

**The format is:**

```
[action1:]s1,s2...[| action2:s3,s4,s5...]
```

**where:**

- action1 and action2 for Linux are the nodeset actions specified in the command. For AIX, action1 and action1 can be ‘diskless’ for mkdsklsnode command’ and ‘standalone’ for nimnodeset command.
- s1 and s2 are the scripts to run for action1 in order.
- s3, s4, and s5 are the scripts to run for actions2.

If actions are omitted, the scripts apply to all actions. Examples:

```
myscript1,myscript2 (all actions) diskless:myscript1,myscript2 (AIX) in-
stall:myscript1,myscript2|netboot:myscript3 (Linux)
```

All the scripts should be copied to /install/prescripts directory. The following two environment variables will be passed to each script:

**NODES** a coma separated list of node names that need to run the script for **ACTION** current nodeset action.

If ‘#xCAT setting:MAX\_INSTANCE=number’ is specified in the script, the script will get invoked for each node in parallel, but no more than number of instances will be invoked at a time. If it is not specified, the script will be invoked once for all the nodes.

#### **prescripts-end** (prescripts.end)

The scripts to be run at the end of the nodeset(Linux), nimnodeset(AIX),or mkdsklsnode(AIX) command. The format is the same as the ‘begin’ column.

#### **primarynic** (noderes.primarynic)

This attribute will be deprecated. All the used network interface will be determined by installnic. The network adapter on the node that will be used for xCAT management, the primarynic can be set to the network adapter name or the mac address or the keyword “mac” which means that the network interface specified by the mac address in the mac table will be used. Default is eth0.

#### **privkey** (pdu.privkey)

The privacy passphrase to use for SNMPv3.

**privtype** (pdu.privtype)

The privacy protocol(AES|DES) to use for SNMPv3.

**productkey** (prodkey.key)

The product key relevant to the aforementioned node/group and product combination

**profile** (nodetype.profile)

The string to use to locate a kickstart or autoyast template to use for OS deployment of this node. If the provmethod attribute is set to an osimage name, that takes precedence, and profile need not be defined. Otherwise, the os, profile, and arch are used to search for the files in /install/custom first, and then in /opt/xcat/share/xcat.

**protocol** (switches.protocol)

Protocol for running remote commands for the switch. The valid values are: ssh, telnet. ssh is the default. If the sshusername is blank, the username, password and protocol will be retrieved from the passwd table with “switch” as the key. The passwd.comments attribute is used for protocol.

**provmethod** (nodetype.provmethod)

The provisioning method for node deployment. The valid values are install, netboot, statelite or an os image name from the osimage table. If an image name is specified, the osimage definition stored in the osimage table and the linuximage table (for Linux) or nimimage table (for AIX) are used to locate the files for templates, pkglists, syncfiles, etc. On Linux, if install, netboot or statelite is specified, the os, profile, and arch are used to search for the files in /install/custom first, and then in /opt/xcat/share/xcat.

**rack** (nodepos.rack)

The frame the node is in.

**room** (nodepos.room)

The room where the node is located.

**routenames** (noderes.routenames)

A comma separated list of route names that refer to rows in the routes table. These are the routes that should be defined on this node when it is deployed.

**seclevel** (pdu.seclevel)

The Security Level(noAuthNoPriv|authNoPriv|authPriv) to use for SNMPv3.

**serial** (vpd.serial)

The serial number of the node.

**serialflow** (nodehm.serialflow)

The flow control value of the serial port for this node. For SOL this is typically ‘hard’.

**serialport** (nodehm.serialport)

The serial port for this node, in the linux numbering style (0=COM1/ttyS0, 1=COM2/ttyS1). For SOL on IBM blades, this is typically 1. For rackmount IBM servers, this is typically 0.

**serialspeed** (nodehm.serialspeed)

The speed of the serial port for this node. For SOL this is typically 19200.

**servicenode** (noderes.servicenode)

A comma separated list of node names (as known by the management node) that provides most services for this node. The first service node on the list that is accessible will be used. The 2nd node on the list is generally considered to be the backup service node for this node when running commands like `snmove`.

**setupconserver** (servicenode.conserver)

Do we set up console service on this service node? Valid values: 0, 1, or 2. If 0, it does not change the current state of the service. If 1, configures and starts `conserver` daemon. If 2, configures and starts `goconserver` daemon.

**setupdhcp** (servicenode.dhcpserver)

Do we set up DHCP on this service node? Not supported on AIX. Valid values: 1 or 0. If 1, runs `makedhcp -n`. If 0, it does not change the current state of the service.

**setupftp** (servicenode.ftpsrv)

Do we set up a ftp server on this service node? Not supported on AIX Valid values: 1 or 0. If 1, configure and start `vsftpd`. (You must manually install `vsftpd` on the service nodes before this.) If 0, it does not change the current state of the service. xCAT is not using ftp for compute nodes provisioning or any other xCAT features, so this attribute can be set to 0 if the ftp service will not be used for other purposes

**setupipforward** (servicenode.ipforward)

Do we set up ip forwarding on this service node? Valid values: 1 or 0. If 0, it does not change the current state of the service.

**setupldap** (servicenode.ldapsrv)

Do we set up ldap caching proxy on this service node? Not supported on AIX. Valid values: 1 or 0. If 0, it does not change the current state of the service.

**setupnameserver** (servicenode.namesrv)

Do we set up DNS on this service node? Valid values: 2, 1, or 0. If 2, creates `named.conf` as dns slave, using the management node as dns master, and starts `named`. If 1, creates `named.conf` file with forwarding to the management node and starts `named`. If 0, it does not change the current state of the service.

**setupnfs** (servicenode.nfssrv)

Do we set up file services (HTTP,FTP,or NFS) on this service node? For AIX will only setup NFS, not HTTP or FTP. Valid values: 1 or 0. If 0, it does not change the current state of the service.

**setupnim** (servicenode.nimsrv)

Not used. Do we set up a NIM server on this service node? Valid values: 1 or 0. If 0, it does not change the current state of the service.

**setupntp** (servicenode.ntpsrv)

Not used. Use `setupntp postscript` to setup a ntp server on this service node? Valid values: 1 or 0. If 0, it does not change the current state of the service.

**setupproxydhcp** (servicenode.proxydhcp)

Do we set up proxydhcp service on this node? valid values: 1 or 0. If 1, the proxydhcp daemon will be enabled on this node.

**setuptftp** (servicenode.tftpsrv)

Do we set up TFTP on this service node? Not supported on AIX. Valid values: 1 or 0. If 1, configures and starts `atftp`. If 0, it does not change the current state of the service.

**sfp** (ppc.sfp)

The Service Focal Point of this Frame. This is the name of the HMC that is responsible for collecting hardware service events for this frame and all of the CECs within this frame.

**side** (vpd.side)

<BPA>-<port> or <FSP>-<port>. The side information for the BPA/FSP. The side attribute refers to which BPA/FSP, A or B, which is determined by the slot value returned from lsslp command. It also lists the physical port within each BPA/FSP which is determined by the IP address order from the lsslp response. This information is used internally when communicating with the BPAs/FSPs

**slot** (nodepos.slot)

The slot number of the blade in the chassis. For PCM, a comma-separated list of slot numbers is stored

**slotid** (mp.id)

The slot number of this blade in the BladeCenter chassis.

**slots** (mpa.slots)

The number of available slots in the chassis. For PCM, this attribute is used to store the number of slots in the following format: <slot rows>,<slot columns>,<slot orientation> Where:

```
<slot rows> = number of rows of slots in chassis
<slot columns> = number of columns of slots in chassis
<slot orientation> = set to 0 if slots are vertical, and set to 1 if slots of
↳horizontal
```

**snmpauth** (switches.auth)

The authentication protocol to use for SNMPv3. SHA is assumed if v3 enabled and this is unspecified

**snmppassword** (switches.password)

The password string for SNMPv3 or community string for SNMPv1/SNMPv2. Falls back to passwd table, and site snmpc value if using SNMPv1/SNMPv2.

**snmpprivacy** (switches.privacy)

The privacy protocol to use for v3. xCAT will use authNoPriv if this is unspecified. DES is recommended to use if v3 enabled, as it is the most readily available.

**snmpuser** (pdu.snmpuser)

The username to use for SNMPv3 communication, ignored for SNMPv1

**snmpusername** (switches.username)

The username to use for SNMPv3 communication, ignored for SNMPv1

**snmpversion** (pdu.snmpversion, switches.snmpversion)

The version to use to communicate with switch. SNMPv1 is assumed by default.

or

The version to use to communicate with switch. SNMPv1 is assumed by default.

**storagcontroller** (storage.controller)

The management address to attach/detach new volumes. In the scenario involving multiple controllers, this data must be passed as argument rather than by table value

**storagetype** (storage.type)

The plugin used to drive storage configuration (e.g. svc)



**supernode** (ppc.supernode)

Indicates the connectivity of this CEC in the HFI network. A comma separated list of 2 ids. The first one is the supernode number the CEC is part of. The second one is the logical location number (0-3) of this CEC within the supernode.

**supportedarchs** (nodetype.supportedarchs)

Comma delimited list of architectures this node can execute.

**supportproxydhcp** (noderes.proxydhcp)

To specify whether the node supports proxydhcp protocol. Valid values: yes or 1, no or 0. Default value is yes.

**switch** (switch.switch)

The switch hostname.

**switchinterface** (switch.interface)

The interface name from the node perspective. For example, eth0. For the primary nic, it can be empty, the word “primary” or “primary:ethx” where ethx is the interface name.

**switchport** (switch.port)

The port number in the switch that this node is connected to. On a simple 1U switch, an administrator can generally enter the number as printed next to the ports, and xCAT will understand switch representation differences. On stacked switches or switches with line cards, administrators should usually use the CLI representation (i.e. 2/0/1 or 5/8). One notable exception is stacked SMC 8848M switches, in which you must add 56 for the proceeding switch, then the port number. For example, port 3 on the second switch in an SMC8848M stack would be 59

**switchtype** (switches.switchtype)

The type of switch. It is used to identify the file name that implements the functions for this switch. The valid values are: Mellanox, Cisco, BNT and Juniper.

**switchvlan** (switch.vlan)

The ID for the tagged vlan that is created on this port using mkvlan and chvlan commands.

**syslog** (noderes.syslog)

To configure how to configure syslog for compute node. Valid values:blank(not set), ignore. blank - run postscript syslog; ignore - do NOT run postscript syslog

**termport** (nodehm.termport)

The port number on the terminal server that this node is connected to.

**termserver** (nodehm.termserver)

The hostname of the terminal server.

**tftpdn** (noderes.tftpdn)

The directory that roots this nodes contents from a tftp and related perspective. Used for NAS offload by using different mountpoints.

**tftpserver** (noderes.tftpserver)

The TFTP server for this node (as known by this node). If not set, it defaults to networks.tftpserver.

**unit** (nodepos.u)

The vertical position of the node in the frame

**urlpath** (mpa.urlpath)

URL path for the Chassis web interface. The full URL is built as follows: <hostname>/<urlpath>

**usercomment** (nodegroup.comments)

Any user-written notes.

**userid** (zvm.userid)

The z/VM userID of this node.

**username** (ppchcp.username, mpa.username, websrv.username, pdu.username, switches.sshusername)

Userid of the HMC or IVM. If not filled in, xCAT will look in the passwd table for key=hmc or key=ivm. If not in the passwd table, the default used is hscroot for HMCs and padmin for IVMs.

or

Userid to use to access the management module.

or

Userid to use to access the web service.

or

The remote login user name

or

The remote login user name. It can be for ssh or telnet. If it is for telnet, set protocol to “telnet”. If the sshusername is blank, the username, password and protocol will be retrieved from the passwd table with “switch” as the key.

**vmbeacon** (vm.beacon)

This flag is used by xCAT to track the state of the identify LED with respect to the VM.

**vmbootorder** (vm.bootorder)

Boot sequence (i.e. net,hd)

**vmcfgstore** (vm.cfgstore)

Optional location for persistent storage separate of emulated hard drives for virtualization solutions that require persistent store to place configuration data

**vmcluster** (vm.cluster)

Specify to the underlying virtualization infrastructure a cluster membership for the hypervisor.

**vmcpus** (vm.cpus)

Number of CPUs the node should see.

**vmhost** (vm.host)

The system that currently hosts the VM

**vmmanager** (vm.mgr)

The function manager for the virtual machine

**vmmaster** (vm.master)

The name of a master image, if any, this virtual machine is linked to. This is generally set by clonevm and indicates the deletion of a master that would invalidate the storage of this virtual machine

**vmmemory** (vm.memory)

Megabytes of memory the VM currently should be set to.

#### **vmnicnicmodel** (vm.nicmodel)

Model of NICs that will be provided to VMs (i.e. e1000, rtl8139, virtio, etc)

#### **vmnics** (vm.nics)

Network configuration parameters. Of the general form [physnet:]interface,... Generally, interface describes the vlan entity (default for native, tagged for tagged, vl[number] for a specific vlan. physnet is a virtual switch name or port description that is used for some virtualization technologies to construct virtual switches. hypervisor.netmap can map names to hypervisor specific layouts, or the descriptions described there may be used directly here where possible.

#### **vmothersetting** (vm.othersettings)

**This is a semicolon-delimited list of key-value pairs to be included in a vmx file of VMware or KVM. DO NOT use 'chdef <node> -p|-m vmothersetting=...' to add options to it or delete options from it because chdef uses commas, not semicolons, to separate items.**

##### **Hugepage on POWER systems:**

Specify the hugepage and/or bsr (Barrier Synchronization Register) values, e.g., 'hugepage:1,bsr:2'.

##### **KVM CPU mode:**

Specify how the host CPUs are utilized, e.g., 'cpumode:host-passthrough', 'cpumode:host-model'. With the passthrough mode, the performance of x86 VMs can be improved significantly.

##### **KVM CPU pinning:**

Specify which host CPUs are used, e.g., 'vcpupin:'0-15,^8'', where '-' denotes the range and '^' denotes exclusion. This option allows a comma-delimited list.

##### **KVM memory binding:**

Specify which nodes that host memory are used, e.g., 'membind:0', where the memory in node0 of the hypervisor is used. /sys/devices/system/node has node0 and node8 on some POWER systems, node0 and node1 on some x86\_64 systems. This option allows a guest VM to access specific memory regions.

##### **PCI passthrough:**

PCI devices can be assigned to a virtual machine for exclusive usage, e.g., 'dev-passthrough:pci\_0001\_01\_00\_0,pci\_0000\_03\_00\_0'. A PCI device can also be expressed as 'devpassthrough:0001:01:00.1'. The devices are put in a comma-delimited list. The PCI device names can be obtained by running **virsh nodedev-list** on the host.

##### **VM machine type:**

Specify a machine type for VM creation on the host, e.g., 'machine:pc'. Typical machine types are pc, q35, and pseries.

#### **vmphyslots** (vm.physlots)

Specify the physical slots drc index that will assigned to the partition, the delimiter is ',', and the drc index must started with '0x'. For more details, reference manpage for 'lsvm'.

#### **vmstorage** (vm.storage)

A list of storage files or devices to be used. i.e. dir:///cluster/vm/<nodename> or nfs://<server>/path/to/folder/

#### **vmstoragecache** (vm.storagecache)

Select caching scheme to employ. E.g. KVM understands 'none', 'writethrough' and 'writeback'

#### **vmstorageformat** (vm.storageformat)

Select disk format to use by default (e.g. raw versus qcow2)

**vmstoragemodel** (vm.storagemodel)

Model of storage devices to provide to guest

**vmtextconsole** (vm.textconsole)

Tracks the Psuedo-TTY that maps to the serial port or console of a VM

**vmvirtflags** (vm.virtflags)

**General flags used by the virtualization method.**

**For example, in Xen it could, among other things, specify paravirtualized setup, or direct kernel boot. For a hypervisor/dom0 entry, it is the virtualization method (i.e. “xen”). For KVM, the following flag=value pairs are recognized:**

**imageformat=[raw|fullraw|qcow2]**

raw is a generic sparse file that allocates storage on demand fullraw is a generic, non-sparse file that preallocates all space qcow2 is a sparse, copy-on-write capable format implemented at the virtualization layer rather than the filesystem level

**clonemethod=[qemu-img|relink]**

qemu-img allows use of qcow2 to generate virtualization layer copy-on-write relink uses a generic filesystem facility to clone the files on your behalf, but requires filesystem support such as btrfs

**placement\_affinity=[migratable|user\_migratable|pinned]**

**vmvncport** (vm.vncport)

Tracks the current VNC display port (currently not meant to be set)

**webport** (websrv.port)

The port of the web service.

**wherevals** (nodegroup.wherevals)

A list of “attr\*val” pairs that can be used to determine the members of a dynamic group, the delimiter is “::” and the operator \* can be ==, =~, != or !~.

**xcatmaster** (noderes.xcatmaster)

The hostname of the xCAT service node (as known by this node). This acts as the default value for nfsserver and tftpserver, if they are not set. If xcatmaster is not set, the node will use whoever responds to its boot request as its master. For the directed bootp case for POWER, it will use the management node if xcatmaster is not set.

## SEE ALSO

**mkdef(1), chdef(1), lsdef(1), rmdef(1)**

## kit.7

### NAME

**kit** - a logical object definition in the xCAT database.

### SYNOPSIS

**kit Attributes:** *basename, description, isinternal, kitdeployparams, kitdir, kitname, ostype, release, version*

### DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

#### **kit Attributes:**

**basename** (kit.basename)

The kit base name

**description** (kit.description)

The Kit description.

**isinternal** (kit.isinternal)

A flag to indicated if the Kit is internally used. When set to 1, the Kit is internal. If 0 or undefined, the kit is not internal.

**kitdeployparams** (kit.kitdeployparams)

The file containing the default deployment parameters for this Kit. These parameters are added to the OS Image definition.s list of deployment parameters when one or more Kit Components from this Kit are added to the OS Image.

**kitdir** (kit.kitdir)

The path to Kit Installation directory on the Mgt Node.

**kitname** (kit.kitname)

The unique generated kit name, when kit is added to the cluster.

**ostype** (kit.ostype)

The kit OS type. Linux or AIX.

**release** (kit.release)

The kit release

**version** (kit.version)

The kit version

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## kitcomponent.7

### NAME

**kitcomponent** - a logical object definition in the xCAT database.

### SYNOPSIS

**kitcomponent Attributes:** *basename, description, driverpacks, exlist, genimage\_postinstall, kitcompdeps, kitcompname, kitname, kitpkgdeps, kitreponame, postbootscripts, prerequisite, release, serverroles, version*

### DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

#### **kitcomponent Attributes:**

**basename** (kitcomponent.basename)

Kit Component basename.

**description** (kitcomponent.description)

The Kit component description.

**driverpacks** (kitcomponent.driverpacks)

Comma-separated List of driver package names. These must be full names like: pkg1-1.0-1.x86\_64.rpm.

**exlist** (kitcomponent.exlist)

Exclude list file containing the files/directories to exclude when building a diskless image.

**genimage\_postinstall** (kitcomponent.genimage\_postinstall)

Comma-separated list of postinstall scripts that will run during the genimage.

**kitcompdeps** (kitcomponent.kitcompdeps)

Comma-separated list of kit components that this kit component depends on.

**kitcompname** (kitcomponent.kitcompname)

The unique Kit Component name. It is auto-generated when the parent Kit is added to the cluster.

**kitname** (kitcomponent.kitname)

The Kit name which this Kit Component belongs to.

**kitpkgdeps** (kitcomponent.kitpkgdeps)

Comma-separated list of packages that this kit component depends on.

**kitreponame** (kitcomponent.kitreponame)

The Kit Package Repository name which this Kit Component belongs to.

**postbootscripts** (kitcomponent.postbootscripts)

Comma-separated list of postbootscripts that will run during the node boot.

**prerequisite** (kitcomponent.prerequisite)

Prerequisite for this kit component, the prerequisite includes ospkgdeps,preinstall,preupgrade,preuninstall scripts

**release** (kitcomponent.release)

Kit Component release.

**serverroles** (kitcomponent.serverroles)

The types of servers that this Kit Component can install on. Valid types are: mgtnode, servicenode, compute

**version** (kitcomponent.version)

Kit Component version.

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

**kitrepo.7**

## NAME

**kitrepo** - a logical object definition in the xCAT database.

## SYNOPSIS

**kitrepo Attributes:** *compat\_osbasenames, kitname, kitrepor, kitreponame, osarch, osbasename, osmajorversion, osminorversion*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### **kitrepo Attributes:**

**compat\_osbasenames** (kitrepo.compat\_osbasenames)

List of compatible OS base names.

**kitname** (kitrepo.kitname)

The Kit name which this Kit Package Repository belongs to.

**kitrepodir** (kitrepo.kitrepodir)

The path to Kit Repository directory on the Mgt Node.

**kitreponame** (kitrepo.kitreponame)

The unique generated kit repo package name, when kit is added to the cluster.

**osarch** (kitrepo.osarch)

The OS distro arch which this repository is based on.

**osbasename** (kitrepo.osbasename)

The OS distro name which this repository is based on.

**osmajorversion** (kitrepo.osmajorversion)

The OS distro major version which this repository is based on.

**osminorversion** (kitrepo.osminorversion)

The OS distro minor version which this repository is based on. If this attribute is not set, it means that this repo applies to all minor versions.

### **SEE ALSO**

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

### **monitoring.7**

#### **NAME**

**monitoring** - a logical object definition in the xCAT database.

#### **SYNOPSIS**

**monitoring Attributes:** *comments, disable, name, nodestatmon*



## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### monitoring Attributes:

**comments** (monitoring.comments)

Any user-written notes.

**disable** (monitoring.disable)

Set to 'yes' or '1' to comment out this row.

**name** (monitoring.name)

The name of the monitoring plug-in module. The plug-in must be put in /lib/perl/xCAT\_monitoring/. See the man page for monstart for details.

**nodestatmon** (monitoring.nodestatmon)

Specifies if the monitoring plug-in is used to feed the node status to the xCAT cluster. Any one of the following values indicates "yes": y, Y, yes, Yes, YES, 1. Any other value or blank (default), indicates "no".

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## network.7

## NAME

**network** - a logical object definition in the xCAT database.

## SYNOPSIS

**network Attributes:** *ddnsdomain, dhcpserver, domain, dynamicrange, gateway, logservers, mask, mgtifname, mtu, nameservers, net, netname, nodehostname, ntpservers, staticrange, staticrangeincrement, tftpserver, usercomment, vlanid*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### network Attributes:

#### **ddnsdomain** (networks.ddnsdomain)

A domain to be combined with nodename to construct FQDN for DDNS updates induced by DHCP. This is not passed down to the client as “domain”

#### **dhcpserver** (networks.dhcpserver)

The DHCP server that is servicing this network. Required to be explicitly set for pooled service node operation.

#### **domain** (networks.domain)

The DNS domain name (ex. cluster.com).

#### **dynamicrange** (networks.dynamicrange)

The IP address range used by DHCP to assign dynamic IP addresses for requests on this network. This should not overlap with entities expected to be configured with static host declarations, i.e. anything ever expected to be a node with an address registered in the mac table.

#### **gateway** (networks.gateway)

The network gateway. It can be set to an ip address or the keyword <xcatmaster>, the keyword <xcatmaster> indicates the cluster-facing ip address configured on this management node or service node. Leaving this field blank means that there is no gateway for this network.

#### **logservers** (networks.logservers)

The log servers for this network. Used in creating the DHCP network definition. Assumed to be the DHCP server if not set.

#### **mask** (networks.mask)

The network mask.

#### **mgtifname** (networks.mgtifname)

The interface name of the management/service node facing this network. !remote!<nicname> indicates a non-local network on a specific nic for relay DHCP.

#### **mtu** (networks.mtu)

The default MTU for the network, If multiple networks are applied to the same nic on the SN and/or CN, the MTU shall be the same for those networks.

#### **nameservers** (networks.nameservers)

A comma delimited list of DNS servers that each node in this network should use. This value will end up in the nameserver settings of the /etc/resolv.conf on each node in this network. If this attribute value is set to the IP address of an xCAT node, make sure DNS is running on it. In a hierarchical cluster, you can also set this attribute to “<xcatmaster>” to mean the DNS server for each node in this network should be the node that is managing it (either its service node or the management node). Used in creating the DHCP network definition, and DNS configuration.

**net** (networks.net)

The network address.

**netname** (networks.netname)

Name used to identify this network definition.

**nodehostname** (networks.nodehostname)

A regular expression used to specify node name to network-specific hostname. i.e. “/z/-secondary/” would mean that the hostname of “n1” would be n1-secondary on this network. By default, the nodename is assumed to equal the hostname, followed by nodename-interfacesname.

**ntpservers** (networks.ntpservers)

The ntp servers for this network. Used in creating the DHCP network definition. Assumed to be the DHCP server if not set.

**staticrange** (networks.staticrange)

The IP address range used to dynamically assign static IPs to newly discovered nodes. This should not overlap with the dynamicrange nor overlap with entities that were manually assigned static IPs. The format for the attribute value is: <startip>-<endip>.

**staticrangeincrement** (networks.staticrangeincrement)

**tftpserver** (networks.tftpserver)

The TFTP server that is servicing this network. If not set, the DHCP server is assumed.

**usercomment** (networks.comments)

Any user-written notes.

**vlanid** (networks.vlanid)

The vlan ID if this network is within a vlan.

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## node.7

## NAME

**node** - a logical object definition in the xCAT database.

## SYNOPSIS

**node Attributes:** *addkcmdline, appstatus, appstatustime, arch, authdomain, authkey, authtype, bmc, bmcpasssword, bmcport, bmcusername, bmcvlanid, cfgmgr, cfgmgmtroles, cfgserver, chain, chassis, cmdmapping, community, cons, conserver, consoleenabled, consoleondemand, consport, cpucount, cputype, currchain, currstate, dhcpinterfaces, disk-size, displayname, dockercpus, dockerflag, dockerhost, dockermemory, dockernics, domainadminpassword, domainadminuser, domaintype, getmac, groups, hcp, height, hidden, hostcluster, hostinterface, hostmanager, hostnames, host-type, hwtype, id, initrd, installnic, interface, ip, iscsiipassword, iscsiserver, iscsitarget, iscsiuserid, kcmdline, kernel, linkports, mac, memory, mgt, micbridge, michost, micid, miconboot, micpowermgt, micvlog, migrationdest, monserver,*

*mpa, mtm, nameservers, netboot, nfsdir, nfsserver, nicaliases, niccustomscripts, nicdevices, nicextraparams, nichostnameprefixes, nichostnamesuffixes, nicips, nicnetworks, nicsadapter, nictypes, nimserver, node, nodetype, ondiscover, os, osvolumes, otherinterfaces, ou, outlet, parent, passwd.HMC, passwd.admin, passwd.cellogin, passwd.general, passwd.hscroot, password, pdu, pdutype, postbootscripts, postscripts, power, pprofile, prescripts-begin, prescripts-end, primarynic, primarysn, privkey, privtype, productkey, profile, protocol, provmethod, rack, room, routenames, seclevel, serial, serialflow, serialport, serialspeed, servicenode, setupconserver, setupdhcp, setupftp, setupipforward, setupldap, setupnameserver, setupnfs, setupnim, setupntp, setupproxypdhc, setupftp, sfp, side, slot, slotid, slots, snmpauth, snmppassword, snmpprivacy, snmpuser, snmpusername, snmpversion, status, statustime, storagcontroller, storage type, supernode, supportedarchs, supportproxypdhc, switch, switchinterface, switchport, switchtype, switchvlan, syslog, termport, termserver, tftpdir, tftpserver, unit, updatestatus, updatestatustime, urlpath, usercomment, userid, username, vmbeacon, vmbootorder, vmcfsstore, vmcluster, vmcpus, vmhost, vmmanager, vmmaster, vmmemory, vmnicnicmodel, vmnics, vmothersetting, vmphyslots, vmstorage, vmstoragecache, vmstorageformat, vmstoragemodel, vmtextconsole, vmvirtflags, vmvncport, webport, xcatmaster, zonename*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### node Attributes:

#### **addkcmdline** (bootparams.addkcmdline)

User specified kernel options for os provision process (no prefix) or the provisioned os (with prefix “R::”). Multiple options should be delimited with spaces(” “) and surrounded with quotes. To have the same option used for os provision process and for provisioned os, specify that option with and without the prefix: `addkcmdline=”R::display=3 display=3”`

#### **appstatus** (nodelist.appstatus)

A comma-delimited list of application status. For example: `‘sshd=up,ftp=down,ll=down’`

#### **appstatustime** (nodelist.appstatustime)

The date and time when appstatus was updated.

#### **arch** (nodetype.arch)

The hardware architecture of this node. Valid values: `x86_64`, `ppc64`, `x86`, `ia64`.

#### **authdomain** (domain.authdomain)

If a node should participate in an AD domain or Kerberos realm distinct from domain indicated in site, this field can be used to specify that

#### **authkey** (pdu.authkey)

The authentication passphrase for SNMPv3

#### **authtype** (pdu.authtype)

The authentication protocol(MD5|SHA) to use for SNMPv3.

#### **bmc** (ipmi.bmc, openbmc.bmc)

The hostname of the BMC adapter.

or

The hostname of the BMC adapter.

**bmcpassword** (ipmi.password, openbmc.password)

The BMC password. If not specified, the key=ipmi row in the passwd table is used as the default.

or

The BMC password. If not specified, the key=openbmc row in the passwd table is used as the default.

**bmcport** (ipmi.bmcport)

In systems with selectable shared/dedicated ethernet ports, this parameter can be used to specify the preferred port. 0 means use the shared port, 1 means dedicated, blank is to not assign.

The following special cases exist for IBM System x servers:

For x3755 M3 systems, 0 means use the dedicated port, 1 means shared, blank is to not assign.

For certain systems which have a mezzanine or ML2 adapter, there is a second value to include:

For x3750 M4 (Model 8722):

0 2 1st 1Gbps interface for LOM

0 0 1st 10Gbps interface for LOM

0 3 2nd 1Gbps interface for LOM

0 1 2nd 10Gbps interface for LOM

For x3750 M4 (Model 8752), x3850/3950 X6, dx360 M4, x3550 M4, and x3650 M4:

0 Shared (1st onboard interface)

1 Dedicated

2 0 First interface on ML2 or mezzanine adapter

2 1 Second interface on ML2 or mezzanine adapter

2 2 Third interface on ML2 or mezzanine adapter

2 3 Fourth interface on ML2 or mezzanine adapter

**bmcusername** (ipmi.username, openbmc.username)

The BMC userid. If not specified, the key=ipmi row in the passwd table is used as the default.

or

The BMC userid. If not specified, the key=openbmc row in the passwd table is used as the default.

**bmcvlantag** (ipmi.taggedvlan, openbmc.taggedvlan)

bmcsetup script will configure the network interface of the BMC to be tagged to the VLAN specified.

or

bmcsetup script will configure the network interface of the BMC to be tagged to the VLAN specified.

**cfgmgr** (cfgmgt.cfgmgr)

The name of the configuration manager service. Currently 'chef' and 'puppet' are supported services.

**cfgmgtroles** (cfgmgt.roles)

The roles associated with this node as recognized by the cfgmgr for the software that is to be installed and configured. These role names map to chef recipes or puppet manifest classes that should be used for this node. For example, chef OpenStack cookbooks have roles such as mysql-master,keystone, glance, nova-controller, nova-conductor, cinder-all.

**cfgserver** (cfgmgt.cfgserver)

The xCAT node name of the chef server or puppet master

**chain** (chain.chain)

A comma-delimited chain of actions to be performed automatically when this node is discovered for the first time. (xCAT and the DHCP server do not recognize the MAC address of the node when xCAT initializes the discovery process.) The last step in this process is to run the operations listed in the chain attribute, one by one. Valid values: boot, runcmd=<cmd>, runimage=<URL>, shell, standby. For example, to have the genesis kernel pause to the shell, use chain=shell.

**chassis** (nodepos.chassis)

The BladeCenter chassis the blade is in.

**cmdmapping** (nodehm.cmdmapping)

The fully qualified name of the file that stores the mapping between PCM hardware management commands and xCAT/third-party hardware management commands for a particular type of hardware device. Only used by PCM.

**community** (pdu.community)

The community string to use for SNMPv1/v2

**cons** (nodehm.cons)

The console method. If nodehm.serialport is set, this will default to the nodehm.mgt setting, otherwise it defaults to unused. Valid values: cyclades, mrv, or the values valid for the mgt attribute.

**conserver** (nodehm.conserver)

The hostname of the machine where the conserver daemon is running. If not set, the default is the xCAT management node.

**consoleenabled** (nodehm.consoleenabled)

A flag field to indicate whether the node is registered in the console server. If '1', console is enabled, if not set, console is not enabled.

**consoleondemand** (nodehm.consoleondemand)

This overrides the value from site.consoleondemand. Set to 'yes', 'no', '1' (equivalent to 'yes'), or '0' (equivalent to 'no'). If not set, the default is the value from site.consoleondemand.

**consport** (openbmc.consport)

The console port for OpenBMC.

**cpucount** (hwinv.cpucount)

The number of cpus for the node.

**cputype** (hwinv.cputype)

The cpu model name for the node.

**currchain** (chain.currchain)

The chain steps still left to do for this node. This attribute will be automatically adjusted by xCAT while xCAT-genesis is running on the node (either during node discovery or a special operation like firmware update). During node discovery, this attribute is initialized from the chain attribute and updated as the chain steps are executed.

**currstate** (chain.currstate)

The current or next chain step to be executed on this node by xCAT-genesis. Set by xCAT during node discovery or as a result of nodeset.

**dhcpinterfaces** (servicenode.dhcpinterfaces)

The network interfaces DHCP server should listen on for the target node. This attribute can be used for management node and service nodes. If defined, it will override the values defined in site.dhcpinterfaces. This is a comma separated list of device names. !remote! indicates a non-local network for relay DHCP. For example: !remote!,eth0,eth1

**disksize** (hwinv.disksize)

The size of the disks for the node in GB.

**displayname** (mpa.displayname)

Alternative name for BladeCenter chassis. Only used by PCM.

**dockercpus** (vm.cpus)

Number of CPUs the node should see.

**dockerflag** (vm.othersettings)

**This is a semicolon-delimited list of key-value pairs to be included in a vmx file of VMware or KVM. DO NOT use 'chdef <node> -p-l-m vmothersetting=...' to add options to it or delete options from it because chdef uses commas, not semicolons, to separate items.**

#### **Hugepage on POWER systems:**

Specify the hugepage and/or bsr (Barrier Synchronization Register) values, e.g., 'hugepage:1,bsr:2'.

#### **KVM CPU mode:**

Specify how the host CPUs are utilized, e.g., 'cpumode:host-passthrough', 'cpumode:host-model'. With the passthrough mode, the performance of x86 VMs can be improved significantly.

#### **KVM CPU pinning:**

Specify which host CPUs are used, e.g., 'vcupin:'0-15,^8'', where '-' denotes the range and '^' denotes exclusion. This option allows a comma-delimited list.

#### **KVM memory binding:**

Specify which nodes that host memory are used, e.g., 'membind:0', where the memory in node0 of the hypervisor is used. /sys/devices/system/node has node0 and node8 on some POWER systems, node0 and node1 on some x86\_64 systems. This option allows a guest VM to access specific memory regions.

**PCI passthrough:**

PCI devices can be assigned to a virtual machine for exclusive usage, e.g., ‘dev-passthrough:pci\_0001\_01\_00\_0,pci\_0000\_03\_00\_0’. A PCI device can also be expressed as ‘devpassthrough:0001:01:00.1’. The devices are put in a comma-delimited list. The PCI device names can be obtained by running **virsh nodedev-list** on the host.

**VM machine type:**

Specify a machine type for VM creation on the host, e.g., ‘machine:pc’. Typical machine types are pc, q35, and pseries.

**dockerhost** (vm.host)

The system that currently hosts the VM

**dockermemory** (vm.memory)

Megabytes of memory the VM currently should be set to.

**dockernics** (vm.nics)

Network configuration parameters. Of the general form [physnet:]interface,.. Generally, interface describes the vlan entity (default for native, tagged for tagged, vl[number] for a specific vlan. physnet is a virtual switch name or port description that is used for some virtualization technologies to construct virtual switches. hypervisor.netmap can map names to hypervisor specific layouts, or the descriptions described there may be used directly here where possible.

**domainadminpassword** (domain.adminpassword)

Allow a node specific indication of Administrative user password for the domain. Most will want to ignore this in favor of passwd table.

**domainadminuser** (domain.adminuser)

Allow a node specific indication of Administrative user. Most will want to just use passwd table to indicate this once rather than by node.

**domaintype** (domain.type)

Type, if any, of authentication domain to manipulate. The only recognized value at the moment is activedirectory.

**getmac** (nodehm.getmac)

The method to use to get MAC address of the node with the getmac command. If not set, the mgt attribute will be used. Valid values: same as values for mgmt attribute.

**groups** (nodelist.groups)

A comma-delimited list of groups this node is a member of. Group names are arbitrary, except all nodes should be part of the ‘all’ group. Internal group names are designated by using \_\_<groupname>. For example, \_\_Unmanaged, could be the internal name for a group of nodes that is not managed by xCAT. Admins should avoid using the \_\_ characters when defining their groups.

**hcp** (ppc.hcp, zvm.hcp)

The hardware control point for this node (HMC, IVM, Frame or CEC). Do not need to set for BPAs and FSPs.

or

The hardware control point for this node.

**height** (nodepos.height)

The server height in U(s).



**hidden** (nodelist.hidden)

Used to hide fsp and bpa definitions, 1 means not show them when running lsdef and nodels

**hostcluster** (hypervisor.cluster)

Specify to the underlying virtualization infrastructure a cluster membership for the hypervisor.

**hostinterface** (hypervisor.interface)

The definition of interfaces for the hypervisor. The format is [network-name:interfacename:bootprotocol:IP:netmask:gateway] that split with | for each interface

**hostmanager** (hypervisor.mgr)

The virtualization specific manager of this hypervisor when applicable

**hostnames** (hosts.hostnames)

Hostname aliases added to /etc/hosts for this node. Comma or blank separated list.

**hosttype** (hypervisor.type)

The plugin associated with hypervisor specific commands such as revacuate

**hwtype** (ppc.nodetype, zvm.nodetype, mp.nodetype, mic.nodetype)

The hardware type of the node. Only can be one of fsp, bpa, cec, frame, ivm, hmc and lpar

or

The node type. Valid values: cec (Central Electronic Complex), lpar (logical partition), zvm (z/VM host operating system), and vm (virtual machine).

or

The hardware type for mp node. Valid values: mm,cmm, blade.

or

The hardware type of the mic node. Generally, it is mic.

**id** (ppc.id, mp.id)

For LPARs: the LPAR numeric id; for CECs: the cage number; for Frames: the frame number.

or

The slot number of this blade in the BladeCenter chassis.

**initrd** (bootparams.initrd)

The initial ramdisk image that network boot actions should use (could be a DOS floppy or hard drive image if using memdisk as kernel)

**installnic** (noderes.installnic)

The network adapter on the node that will be used for OS deployment, the installnic can be set to the network adapter name or the mac address or the keyword "mac" which means that the network interface specified by the mac address in the mac table will be used. If not set, primarynic will be used. If primarynic is not set too, the keyword "mac" will be used as default.

**interface** (mac.interface)

The adapter interface name that will be used to install and manage the node. E.g. eth0 (for linux) or en0 (for AIX).)

**ip** (hosts.ip)

The IP address of the node. This is only used in makehosts. The rest of xCAT uses system name resolution to resolve node names to IP addresses.

**iscsipassword** (iscsi.passwd)

The password for the iscsi server containing the boot device for this node.

**iscsiserver** (iscsi.server)

The server containing the iscsi boot device for this node.

**iscsitarget** (iscsi.target)

The iscsi disk used for the boot device for this node. Filled in by xCAT.

**iscsiuserid** (iscsi.userid)

The userid of the iscsi server containing the boot device for this node.

**kcmdline** (bootparams.kcmdline)

(Deprecated, use addkcmdline instead) Arguments to be passed to the kernel.

**kernel** (bootparams.kernel)

The kernel that network boot actions should currently acquire and use. Note this could be a chained boot loader such as memdisk or a non-linux boot loader

**linkports** (switches.linkports)

The ports that connect to other switches. Currently, this column is only used by vlan configuration. The format is: “port\_number:switch,port\_number:switch...”. Refer to the switch table for details on how to specify the port numbers.

**mac** (mac.mac)

The mac address or addresses for which xCAT will manage static bindings for this node. This may be simply a mac address, which would be bound to the node name (such as “01:02:03:04:05:0E”). This may also be a “|” delimited string of “mac address!hostname” format (such as “01:02:03:04:05:0E!node5|01:02:03:04:05:0F!node6-eth1”). If there are multiple nics connected to Management Network(usually for bond), in order to make sure the OS deployment finished successfully, the macs of those nics must be able to resolve to same IP address. First, users have to create alias of the node for each mac in the Management Network through either: 1. adding the alias into /etc/hosts for the node directly or: 2. setting the alias to the “hostnames” attribute and then run “makehost” against the node. Then, configure the “mac” attribute of the node like “mac1!node|mac2!node-alias”. For the first mac address (mac1 in the example) set in “mac” attribute, do not need to set a “node name” string for it since the nodename of the node will be used for it by default.

**memory** (hwinv.memory)

The size of the memory for the node in MB.

**mgt** (nodehm.mgt)

The method to use to do general hardware management of the node. This attribute is used as the default if power or getmac is not set. Valid values: openbmc, ipmi, blade, hmc, ivm, fsp, bpa, kvm, esx, rhevm. See the power attribute for more details.

**micbridge** (mic.bridge)

The virtual bridge on the host node which the mic connected to.

**michost** (mic.host)

The host node which the mic card installed on.

**micid** (mic.id)

The device id of the mic node.

**miconboot** (mic.onboot)

Set mic to autoboot when mpss start. Valid values: yes|no. Default is yes.

**micpowermgt** (mic.powermgt)

Set the Power Management for mic node. This attribute is used to set the power management state that mic may get into when it is idle. Four states can be set: cpufreq, corec6, pc3 and pc6. The valid value for powermgt attribute should be [cpufreq=<on|off>]![corec6=<on|off>]![pc3=<on|off>]![pc6=<on|off>]. e.g. cpufreq=on!corec6=off!pc3=on!pc6=off. Refer to the doc of mic to get more information for power management.

**micvlog** (mic.vlog)

Set the Verbose Log to console. Valid values: yes|no. Default is no.

**migrationdest** (vm.migrationdest)

A nodestring representing candidate destinations for migration (i.e. similar systems, same SAN, or other criteria that xCAT can use)

**monserver** (noderes.monserver)

The monitoring aggregation point for this node. The format is “x,y” where x is the ip address as known by the management node and y is the ip address as known by the node.

**mpa** (mp.mpa)

The management module used to control this blade.

**mtm** (vpd.mtm)

The machine type and model number of the node. E.g. 7984-6BU

**nameservers** (noderes.nameservers)

An optional node/group specific override for name server list. Most people want to stick to site or network defined nameserver configuration.

**netboot** (noderes.netboot)

The type of network booting to use for this node. Valid values:

Arch	OS	valid netboot options
x86, x86_64	ALL	pxe, xnba
ppc64	<=rhel6, <=sles11.3	yaboot
ppc64	>=rhels7, >=sles11.4	grub2,grub2-http,grub2-
↪tftp		
ppc64le NonVirtualize	ALL	petitboot
ppc64le PowerKVM Guest	ALL	grub2,grub2-http,grub2-
↪tftp		

**nfsdir** (noderes.nfsdir)

The path that should be mounted from the NFS server.

**nfsserver** (noderes.nfsserver)

The NFS or HTTP server for this node (as known by this node).

**nicaliases** (nics.nicaliases)

**Comma-separated list of hostname aliases for each NIC.**

**Format:** `eth0!<alias list>,eth1!<alias1 list>|<alias2 list>`

For multiple aliases per nic use a space-separated list.

For example: `eth0!moe larry curly,eth1!tomljerry`

**niccustomscripts** (nics.niccustomscripts)

Comma-separated list of custom scripts per NIC. `<nic1>!<script1>,<nic2>!<script2>`, e.g. `eth0!configeth eth0, ib0!configib ib0`. The xCAT object definition commands support to use `niccustomscripts.<nicname>` as the sub attribute .

**nicdevices** (nics.nicdevices)

Comma-separated list of NIC device per NIC, multiple ethernet devices can be bonded as bond device, these ethernet devices are separated by `|` . `<nic1>!<dev1>|<dev3>,<nic2>!<dev2>`, e.g. `bond0!eth0|eth2,br0!bond0`. The xCAT object definition commands support to use `nicdevices.<nicname>` as the sub attributes.

**nicextraparams** (nics.nicextraparams)**Comma-separated list of extra parameters that will be used for each NIC configuration.****If only one ip address is associated with each NIC:**

`<nic1>!<param1=value1 param2=value2>,<nic2>!<param3=value3>`, for example, `eth0!MTU=1500,ib0!MTU=65520 CONNECTED_MODE=yes`.

**If multiple ip addresses are associated with each NIC:**

`<nic1>!<param1=value1 param2=value2>|<param3=value3>,<nic2>!<param4=value4 param5=value5>|<param6=value6>`, for example, `eth0!MTU=1500|MTU=1460,ib0!MTU=65520 CONNECTED_MODE=yes`.

**The semicolon separator is needed if there are multiple values for extra parameters:**

`bond0!BONDING_OPTS=lacp_rate=1;miimon=100;mode=802.3ad`

The xCAT object definition commands support to use `nicextraparams.<nicname>` as the sub attributes.

**nichostnameprefixes** (nics.nichostnameprefixes)**Comma-separated list of hostname prefixes per NIC.****If only one ip address is associated with each NIC:**

`<nic1>!<ext1>,<nic2>!<ext2>,...`, for example, `eth0!eth0-,ib0!ib-`

**If multiple ip addresses are associated with each NIC:**

`<nic1>!<ext1>|<ext2>,<nic2>!<ext1>|<ext2>,...`, for example, `eth0!eth0-|eth0-ipv6i-,ib0!ib-|ib-ipv6-`.

The xCAT object definition commands support to use `nichostnameprefixes.<nicname>` as the sub attributes. Note: According to DNS rules a hostname must be a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9) and minus sign (-). When you are specifying “nichostnameprefixes” or “nicaliasess” make sure the resulting hostnames will conform to this naming convention

**nichostnamesuffixes** (nics.nichostnamesuffixes)**Comma-separated list of hostname suffixes per NIC.****If only one ip address is associated with each NIC:**

`<nic1>!<ext1>,<nic2>!<ext2>,...`, for example, `eth0!-eth0,ib0!-ib0`

### If multiple ip addresses are associated with each NIC:

<nic1>|<ext1>|<ext2>,<nic2>|<ext1>|<ext2>,..., for example, eth0!-eth0|-eth0-ipv6,ib0!-ib0|-ib0-ipv6.

The xCAT object definition commands support to use nichostnamesuffixes.<nicname> as the sub attributes.

Note: According to DNS rules a hostname must be a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9) and minus sign (-). When you are specifying "nichostnamesuffixes" or "nicaliases" make sure the resulting hostnames will conform to this naming convention

**nicips** (nics.nicips)

### Comma-separated list of IP addresses per NIC.

#### To specify one ip address per NIC:

<nic1>|<ip1>,<nic2>|<ip2>,..., for example, eth0!10.0.0.100,ib0!11.0.0.100

#### To specify multiple ip addresses per NIC:

<nic1>|<ip1>|<ip2>,<nic2>|<ip1>|<ip2>,..., for example, eth0!10.0.0.100|fd55::214:5eff:fe15:849b,ib0!11.0.0.100|2001::1

The xCAT object definition commands support to use nicips.<nicname> as the sub attributes.

Note: The primary IP address must also be stored in the hosts.ip attribute. The nichostnamesuffixes should specify one hostname suffix for each ip address.

**nicnetworks** (nics.nicnetworks)

### Comma-separated list of networks connected to each NIC.

#### If only one ip address is associated with each NIC:

<nic1>|<network1>,<nic2>|<network2>, for example, eth0!10\_0\_0\_0-255\_255\_0\_0,ib0!11\_0\_0\_0-255\_255\_0\_0

#### If multiple ip addresses are associated with each NIC:

<nic1>|<network1>|<network2>,<nic2>|<network1>|<network2>, for example, eth0!10\_0\_0\_0-255\_255\_0\_0|fd55:faaf:e1ab:336::/64,ib0!11\_0\_0\_0-255\_255\_0\_0|2001:db8:1:0::/64. The xCAT object definition commands support to use nicnetworks.<nicname> as the sub attributes.

**nicadapter** (nics.nicsadapter)

Comma-separated list of NIC information collected by getadapter. <nic1>|<param1=value1 param2=value2>,<nic2>|<param4=value4 param5=value5>, for example, enP3p3s0f1!mac=98:be:94:59:fa:cd linkstate=DOWN,enP3p3s0f2!mac=98:be:94:59:fa:ce candidate-name=enP3p3s0f2/enx98be9459face

**nictypes** (nics.nictypes)

Comma-separated list of NIC types per NIC. <nic1>|<type1>,<nic2>|<type2>, e.g. eth0!Ethernet,ib0!Infiniband. The xCAT object definition commands support to use nictypes.<nicname> as the sub attributes.

**nimserver** (noderes.nimserver)

Not used for now. The NIM server for this node (as known by this node).

**node** (nodelist.node)

The hostname of a node in the cluster.

**nodetype** (nodetype.nodetype, pdu.nodetype)

A comma-delimited list of characteristics of this node. Valid values: ppc, blade, vm (virtual machine), osi (OS image), mm, mn, rsa, switch.

or

The node type should be pdu

**ondiscover** (chain.ondiscover)

This attribute is currently not used by xCAT. The “nodediscover” operation is always done during node discovery.

**os** (nodetype.os)

The operating system deployed on this node. Valid values: AIX, rhels\*, rhelc\*, rhas\*, centos\*, alma\*, rocky\*, SL\*, fedora\*, sles\* (where \* is the version #). As a special case, if this is set to “boottarget”, then it will use the initrd/kernel/parameters specified in the row in the boottarget table in which boottarget.bprofile equals nodetype.profile.

**osvolume** (storage.osvolume)

Specification of what storage to place the node OS image onto. Examples include:

localdisk (Install to first non-FC attached disk)  
usbdisk (Install to first USB mass storage device seen)  
wwn=0x50000393c813840c (Install to storage device with given WWN)

**otherinterfaces** (hosts.otherinterfaces)

Other IP addresses to add for this node. Format: -<ext>:<ip>,<intfhostname>:<ip>,...

**ou** (domain.ou)

For an LDAP described machine account (i.e. Active Directory), the organizational unit to place the system. If not set, defaults to cn=Computers,dc=your,dc=domain

**outlet** (pdu.outlet)

The pdu outlet count

**parent** (ppc.parent)

For LPARs: the CEC; for FSPs: the CEC; for CEC: the frame (if one exists); for BPA: the frame; for frame: the building block number (which consists 1 or more service nodes and compute/storage nodes that are serviced by them - optional).

**passwd.HMC** (ppcdirect.password)

Password of the FSP/BPA(for ASMI) and CEC/Frame(for DFM). If not filled in, xCAT will look in the passwd table for key=fsp. If not in the passwd table, the default used is admin.

**passwd.admin** (ppcdirect.password)

Password of the FSP/BPA(for ASMI) and CEC/Frame(for DFM). If not filled in, xCAT will look in the passwd table for key=fsp. If not in the passwd table, the default used is admin.

**passwd.cellogin** (ppcdirect.password)

Password of the FSP/BPA(for ASMI) and CEC/Frame(for DFM). If not filled in, xCAT will look in the passwd table for key=fsp. If not in the passwd table, the default used is admin.

**passwd.general** (ppcdirect.password)

Password of the FSP/BPA(for ASMI) and CEC/Frame(for DFM). If not filled in, xCAT will look in the passwd table for key=fsp. If not in the passwd table, the default used is admin.

**passwd.hscroot** (ppcdirect.password)

Password of the FSP/BPA(for ASMI) and CEC/Frame(for DFM). If not filled in, xCAT will look in the passwd table for key=fsp. If not in the passwd table, the default used is admin.

**password** (ppchcp.password, mpa.password, websrv.password, pdu.password, switches.sshpassword)

Password of the HMC or IVM. If not filled in, xCAT will look in the passwd table for key=hmc or key=ivm. If not in the passwd table, the default used is abc123 for HMCs and padmin for IVMs.

or

Password to use to access the management module. If not specified, the key=blade row in the passwd table is used as the default.

or

Password to use to access the web service.

or

The remote login password

or

The remote login password. It can be for ssh or telnet. If it is for telnet, set protocol to “telnet”. If the sshusername is blank, the username, password and protocol will be retrieved from the passwd table with “switch” as the key.

**pdu** (pduoutlet.pdu)

a comma-separated list of outlet number for each PDU, ex: pdu1:outlet1,pdu2:outlet1

**pdu type** (pdu.pdu type)

The type of pdu

**postbootscripts** (postscripts.postbootscripts)

Comma separated list of scripts that should be run on this node after diskful installation or diskless boot. Each script can take zero or more parameters. For example: “script1 p1 p2,script2,...”. On AIX these scripts are run during the processing of /etc/inittab. On Linux they are run at the init.d time. xCAT automatically adds the scripts in the xcatdefaults.postbootscripts attribute to run first in the list. Please note that the postbootscripts specified for “xcatdefaults” will be assigned to node automatically, they can not be removed from “postbootscripts” attribute of a node with “chdef -m” command

**postscripts** (postscripts.postscripts)

Comma separated list of scripts that should be run on this node after diskful installation or diskless boot. Each script can take zero or more parameters. For example: “script1 p1 p2,script2,...”. xCAT automatically adds the postscripts from the xcatdefaults.postscripts attribute of the table to run first on the nodes after install or diskless boot. For installation of RedHat, CentOS, Fedora, the scripts will be run before the reboot. For installation of SLES, the scripts will be run after the reboot but before the init.d process. For diskless deployment, the scripts will be run at the init.d time, and xCAT will automatically add the list of scripts from the postbootscripts attribute to run after postscripts list. For installation of AIX, the scripts will run after the reboot and acts the same as the postbootscripts attribute. For AIX, use the postbootscripts attribute. Please note that the postscripts specified for “xcatdefaults” will be assigned to node automatically, they can not be removed from “postscripts” attribute of a node with “chdef -m” command

**power** (nodehm.power)

The method to use to control the power of the node. If not set, the mgt attribute will be used. Valid values: ipmi, blade, hmc, ivm, fsp, kvm, esx, rhevm. If “ipmi”, xCAT will search for this node in the ipmi table

for more info. If “blade”, xCAT will search for this node in the mp table. If “hmc”, “ivm”, or “fsp”, xCAT will search for this node in the ppc table.

**pprofile** (ppc.pprofile)

The LPAR profile that will be used the next time the LPAR is powered on with rpower. For DFM, the pprofile attribute should be set to blank

**prescripts-begin** (prescripts.begin)

**The scripts to be run at the beginning of the nodeset(Linux), nimnodeset(AIX) or mkdsklsnode(AIX) command.**

**The format is:**

[action1:]s1,s2...[| action2:s3,s4,s5...]

**where:**

- action1 and action2 for Linux are the nodeset actions specified in the command. For AIX, action1 and action1 can be ‘diskless’ for mkdsklsnode command’ and ‘standalone’ for nimnodeset command.
- s1 and s2 are the scripts to run for action1 in order.
- s3, s4, and s5 are the scripts to run for actions2.

If actions are omitted, the scripts apply to all actions. Examples:

myscript1,myscript2 (all actions) diskless:myscript1,myscript2 (AIX) in-  
stall:myscript1,myscript2|netboot:myscript3 (Linux)

All the scripts should be copied to /install/prescripts directory. The following two environment variables will be passed to each script:

NODES a coma separated list of node names that need to run the script for ACTION current nodeset action.

If ‘#xCAT setting:MAX\_INSTANCE=number’ is specified in the script, the script will get invoked for each node in parallel, but no more than number of instances will be invoked at a time. If it is not specified, the script will be invoked once for all the nodes.

**prescripts-end** (prescripts.end)

The scripts to be run at the end of the nodeset(Linux), nimnodeset(AIX),or mkdsklsnode(AIX) command. The format is the same as the ‘begin’ column.

**primarynic** (noderes.primarynic)

This attribute will be deprecated. All the used network interface will be determined by installnic. The network adapter on the node that will be used for xCAT management, the primarynic can be set to the network adapter name or the mac address or the keyword “mac” which means that the network interface specified by the mac address in the mac table will be used. Default is eth0.

**primarysn** (nodelist.primarysn)

Not used currently. The primary servicenode, used by this node.

**privkey** (pdu.privkey)

The privacy passphrase to use for SNMPv3.

**privtype** (pdu.privtype)

The privacy protocol(AES|DES) to use for SNMPv3.

**productkey** (prodkey.key)



The product key relevant to the aforementioned node/group and product combination

**profile** (nodetype.profile)

The string to use to locate a kickstart or autoyast template to use for OS deployment of this node. If the provmethod attribute is set to an osimage name, that takes precedence, and profile need not be defined. Otherwise, the os, profile, and arch are used to search for the files in /install/custom first, and then in /opt/xcat/share/xcat.

**protocol** (switches.protocol)

Protocol for running remote commands for the switch. The valid values are: ssh, telnet. ssh is the default. If the sshusername is blank, the username, password and protocol will be retrieved from the passwd table with “switch” as the key. The passwd.comments attribute is used for protocol.

**provmethod** (nodetype.provmethod)

The provisioning method for node deployment. The valid values are install, netboot, statelite or an os image name from the osimage table. If an image name is specified, the osimage definition stored in the osimage table and the linuximage table (for Linux) or nimimage table (for AIX) are used to locate the files for templates, pkglists, syncfiles, etc. On Linux, if install, netboot or statelite is specified, the os, profile, and arch are used to search for the files in /install/custom first, and then in /opt/xcat/share/xcat.

**rack** (nodepos.rack)

The frame the node is in.

**room** (nodepos.room)

The room where the node is located.

**routenames** (noderes.routenames)

A comma separated list of route names that refer to rows in the routes table. These are the routes that should be defined on this node when it is deployed.

**seclevel** (pdu.seclevel)

The Security Level(noAuthNoPriv|authNoPriv|authPriv) to use for SNMPv3.

**serial** (vpd.serial)

The serial number of the node.

**serialflow** (nodehm.serialflow)

The flow control value of the serial port for this node. For SOL this is typically ‘hard’.

**serialport** (nodehm.serialport)

The serial port for this node, in the linux numbering style (0=COM1/ttyS0, 1=COM2/ttyS1). For SOL on IBM blades, this is typically 1. For rackmount IBM servers, this is typically 0.

**serialspeed** (nodehm.serialspeed)

The speed of the serial port for this node. For SOL this is typically 19200.

**servicenode** (noderes.servicenode)

A comma separated list of node names (as known by the management node) that provides most services for this node. The first service node on the list that is accessible will be used. The 2nd node on the list is generally considered to be the backup service node for this node when running commands like snmove.

**setupconserver** (servicenode.conserver)

Do we set up console service on this service node? Valid values: 0, 1, or 2. If 0, it does not change the current state of the service. If 1, configures and starts conserver daemon. If 2, configures and starts goconserver daemon.

**setupdhcp** (servicenode.dhcpserver)

Do we set up DHCP on this service node? Not supported on AIX. Valid values:1 or 0. If 1, runs makedhcp -n. If 0, it does not change the current state of the service.

**setupftp** (servicenode.ftpserver)

Do we set up a ftp server on this service node? Not supported on AIX Valid values:1 or 0. If 1, configure and start vsftpd. (You must manually install vsftpd on the service nodes before this.) If 0, it does not change the current state of the service. xCAT is not using ftp for compute nodes provisioning or any other xCAT features, so this attribute can be set to 0 if the ftp service will not be used for other purposes

**setupipforward** (servicenode.ipforward)

Do we set up ip forwarding on this service node? Valid values:1 or 0. If 0, it does not change the current state of the service.

**setupldap** (servicenode.ldapservice)

Do we set up ldap caching proxy on this service node? Not supported on AIX. Valid values:1 or 0. If 0, it does not change the current state of the service.

**setupnameserver** (servicenode.nameserver)

Do we set up DNS on this service node? Valid values: 2, 1, or 0. If 2, creates named.conf as dns slave, using the management node as dns master, and starts named. If 1, creates named.conf file with forwarding to the management node and starts named. If 0, it does not change the current state of the service.

**setupnfs** (servicenode.nfsservice)

Do we set up file services (HTTP,FTP,or NFS) on this service node? For AIX will only setup NFS, not HTTP or FTP. Valid values:1 or 0.If 0, it does not change the current state of the service.

**setupnim** (servicenode.nimserver)

Not used. Do we set up a NIM server on this service node? Valid values:1 or 0. If 0, it does not change the current state of the service.

**setupntp** (servicenode.ntpservice)

Not used. Use setupntp postscript to setup a ntp server on this service node? Valid values:1 or 0. If 0, it does not change the current state of the service.

**setupproxydhcp** (servicenode.proxydhcp)

Do we set up proxydhcp service on this node? valid values: 1 or 0. If 1, the proxydhcp daemon will be enabled on this node.

**setuptftp** (servicenode.tftpservice)

Do we set up TFTP on this service node? Not supported on AIX. Valid values:1 or 0. If 1, configures and starts atftp. If 0, it does not change the current state of the service.

**sfp** (ppc.sfp)

The Service Focal Point of this Frame. This is the name of the HMC that is responsible for collecting hardware service events for this frame and all of the CECs within this frame.

**side** (vpd.side)

<BPA>-<port> or <FSP>-<port>. The side information for the BPA/FSP. The side attribute refers to which BPA/FSP, A or B, which is determined by the slot value returned from lsslp command. It also lists the physical port within each BPA/FSP which is determined by the IP address order from the lsslp response. This information is used internally when communicating with the BPAs/FSPs

#### **slot** (nodepos.slot)

The slot number of the blade in the chassis. For PCM, a comma-separated list of slot numbers is stored

#### **slotid** (mp.id)

The slot number of this blade in the BladeCenter chassis.

#### **slots** (mpa.slots)

The number of available slots in the chassis. For PCM, this attribute is used to store the number of slots in the following format: <slot rows>,<slot columns>,<slot orientation> Where:

```
<slot rows> = number of rows of slots in chassis
<slot columns> = number of columns of slots in chassis
<slot orientation> = set to 0 if slots are vertical, and set to 1 if slots of
↪ horizontal
```

#### **snmpauth** (switches.auth)

The authentication protocol to use for SNMPv3. SHA is assumed if v3 enabled and this is unspecified

#### **snmppassword** (switches.password)

The password string for SNMPv3 or community string for SNMPv1/SNMPv2. Falls back to passwd table, and site snmpc value if using SNMPv1/SNMPv2.

#### **snmpprivacy** (switches.privacy)

The privacy protocol to use for v3. xCAT will use authNoPriv if this is unspecified. DES is recommended to use if v3 enabled, as it is the most readily available.

#### **snmpuser** (pdu.snmpuser)

The username to use for SNMPv3 communication, ignored for SNMPv1

#### **snmpusername** (switches.username)

The username to use for SNMPv3 communication, ignored for SNMPv1

#### **snmpversion** (pdu.snmpversion, switches.snmpversion)

The version to use to communicate with switch. SNMPv1 is assumed by default.

or

The version to use to communicate with switch. SNMPv1 is assumed by default.

#### **status** (nodelist.status)

The current status of this node. This attribute will be set by xCAT software. Valid values: defined, booting, netbooting, booted, discovering, configuring, installing, alive, standingby, powering-off, unreachable. If blank, defined is assumed. The possible status change sequences are: For installation: defined->[discovering]->[configuring]->[standingby]->installing->booting->[postbooting]->booted->[alive], For diskless deployment: defined->[discovering]->[configuring]->[standingby]->netbooting->[postbooting]->booted->[alive], For booting: [alive/unreachable]->booting->[postbooting]->booted->[alive], For powering off: [alive]->p powering-off->[unreachable], For monitoring: alive->unreachable. Discovering and configuring are for x Series discovery process. Alive and unreachable are set only when there is a monitoring plug-in start monitor the node status for xCAT. Note that the status values will not reflect the real

node status if you change the state of the node from outside of xCAT (i.e. power off the node using HMC GUI).

**statustime** (nodelist.statustime)

The data and time when the status was updated.

**storagecontroller** (storage.controller)

The management address to attach/detach new volumes. In the scenario involving multiple controllers, this data must be passed as argument rather than by table value

**storagetype** (storage.type)

The plugin used to drive storage configuration (e.g. svc)

**supernode** (ppc.supernode)

Indicates the connectivity of this CEC in the HFI network. A comma separated list of 2 ids. The first one is the supernode number the CEC is part of. The second one is the logical location number (0-3) of this CEC within the supernode.

**supportedarchs** (nodetype.supportedarchs)

Comma delimited list of architectures this node can execute.

**supportproxydhcp** (noderes.proxydhcp)

To specify whether the node supports proxydhcp protocol. Valid values: yes or 1, no or 0. Default value is yes.

**switch** (switch.switch)

The switch hostname.

**switchinterface** (switch.interface)

The interface name from the node perspective. For example, eth0. For the primary nic, it can be empty, the word “primary” or “primary:ethx” where ethx is the interface name.

**switchport** (switch.port)

The port number in the switch that this node is connected to. On a simple 1U switch, an administrator can generally enter the number as printed next to the ports, and xCAT will understand switch representation differences. On stacked switches or switches with line cards, administrators should usually use the CLI representation (i.e. 2/0/1 or 5/8). One notable exception is stacked SMC 8848M switches, in which you must add 56 for the proceeding switch, then the port number. For example, port 3 on the second switch in an SMC8848M stack would be 59

**switchtype** (switches.switchtype)

The type of switch. It is used to identify the file name that implements the functions for this switch. The valid values are: Mellanox, Cisco, BNT and Juniper.

**switchvlan** (switch.vlan)

The ID for the tagged vlan that is created on this port using mkvlan and chvlan commands.

**syslog** (noderes.syslog)

To configure how to configure syslog for compute node. Valid values:blank(not set), ignore. blank - run postscript syslog; ignore - do NOT run postscript syslog

**termport** (nodehm.termport)

The port number on the terminal server that this node is connected to.

**termserver** (nodehm.termserver)

The hostname of the terminal server.

**tftpdirdir** (nodes.tftpdirdir)

The directory that roots this nodes contents from a tftp and related perspective. Used for NAS offload by using different mountpoints.

**tftpserver** (nodes.tftpserver)

The TFTP server for this node (as known by this node). If not set, it defaults to networks.tftpserver.

**unit** (nodepos.u)

The vertical position of the node in the frame

**updatestatus** (nodelist.updatestatus)

The current node update status. Valid states are synced, out-of-sync, syncing, failed.

**updatestatustime** (nodelist.updatestatustime)

The date and time when the updatestatus was updated.

**urlpath** (mpa.urlpath)

URL path for the Chassis web interface. The full URL is built as follows: <hostname>/<urlpath>

**usercomment** (nodelist.comments)

Any user-written notes.

**userid** (zvm.userid)

The z/VM userID of this node.

**username** (ppchcp.username, mpa.username, websrv.username, pdu.username, switches.sshusername)

Userid of the HMC or IVM. If not filled in, xCAT will look in the passwd table for key=hmc or key=ivm. If not in the passwd table, the default used is hscroot for HMCs and padmin for IVMs.

or

Userid to use to access the management module.

or

Userid to use to access the web service.

or

The remote login user name

or

The remote login user name. It can be for ssh or telnet. If it is for telnet, set protocol to “telnet”. If the sshusername is blank, the username, password and protocol will be retrieved from the passwd table with “switch” as the key.

**vmbeacon** (vm.beacon)

This flag is used by xCAT to track the state of the identify LED with respect to the VM.

**vmbootorder** (vm.bootorder)

Boot sequence (i.e. net,hd)

**vmcfgstore** (vm.cfgstore)

Optional location for persistent storage separate of emulated hard drives for virtualization solutions that require persistent store to place configuration data

**vmcluster** (vm.cluster)

Specify to the underlying virtualization infrastructure a cluster membership for the hypervisor.

**vmcpus** (vm.cpus)

Number of CPUs the node should see.

**vmhost** (vm.host)

The system that currently hosts the VM

**vmmanager** (vm.mgr)

The function manager for the virtual machine

**vmmaster** (vm.master)

The name of a master image, if any, this virtual machine is linked to. This is generally set by clonevm and indicates the deletion of a master that would invalidate the storage of this virtual machine

**vmmemory** (vm.memory)

Megabytes of memory the VM currently should be set to.

**vmnicnicmodel** (vm.nicmodel)

Model of NICs that will be provided to VMs (i.e. e1000, rtl8139, virtio, etc)

**vmnics** (vm.nics)

Network configuration parameters. Of the general form [physnet:]interface,.. Generally, interface describes the vlan entity (default for native, tagged for tagged, vl[number] for a specific vlan. physnet is a virtual switch name or port description that is used for some virtualization technologies to construct virtual switches. hypervisor.netmap can map names to hypervisor specific layouts, or the descriptions described there may be used directly here where possible.

**vmothersetting** (vm.othersettings)

**This is a semicolon-delimited list of key-value pairs to be included in a vmx file of VMware or KVM. DO NOT use ‘chdef <node> -p|-m vmothersetting=...’ to add options to it or delete options from it because chdef uses commas, not semicolons, to separate items.**

**Hugepage on POWER systems:**

Specify the hugepage and/or bsr (Barrier Synchronization Register) values, e.g., ‘hugepage:1,bsr:2’.

**KVM CPU mode:**

Specify how the host CPUs are utilized, e.g., ‘cpumode:host-passthrough’, ‘cpumode:host-model’. With the passthrough mode, the performance of x86 VMs can be improved significantly.

**KVM CPU pinning:**

Specify which host CPUs are used, e.g., ‘vcpupin:’0-15,^8’, where ‘-’ denotes the range and ‘^’ denotes exclusion. This option allows a comma-delimited list.

**KVM memory binding:**

Specify which nodes that host memory are used, e.g., ‘membind:0’, where the memory in node0 of the hypervisor is used. /sys/devices/system/node has node0 and node8 on some POWER systems, node0 and node1 on some x86\_64 systems. This option allows a guest VM to access specific memory regions.

### PCI passthrough:

PCI devices can be assigned to a virtual machine for exclusive usage, e.g., ‘dev-passthrough:pci\_0001\_01\_00\_0,pci\_0000\_03\_00\_0’. A PCI device can also be expressed as ‘devpassthrough:0001:01:00.1’. The devices are put in a comma-delimited list. The PCI device names can be obtained by running **virsh nodedev-list** on the host.

### VM machine type:

Specify a machine type for VM creation on the host, e.g., ‘machine:pc’. Typical machine types are pc, q35, and pseries.

### vmphyslots (vm.physlots)

Specify the physical slots drc index that will assigned to the partition, the delimiter is ‘,’ and the drc index must started with ‘0x’. For more details, reference manpage for ‘lsvm’.

### vmstorage (vm.storage)

A list of storage files or devices to be used. i.e. dir:///cluster/vm/<nodename> or nfs://<server>/path/to/folder/

### vmstoragecache (vm.storagecache)

Select caching scheme to employ. E.g. KVM understands ‘none’, ‘writethrough’ and ‘writeback’

### vmstorageformat (vm.storageformat)

Select disk format to use by default (e.g. raw versus qcow2)

### vmstoragemodel (vm.storagemodel)

Model of storage devices to provide to guest

### vmtextconsole (vm.textconsole)

Tracks the Psuedo-TTY that maps to the serial port or console of a VM

### vmvirtflags (vm.virtflags)

#### General flags used by the virtualization method.

**For example, in Xen it could, among other things, specify paravirtualized setup, or direct kernel boot. For a hypervisor/dom0 entry, it is the virtualization method (i.e. “xen”). For KVM, the following flag=value pairs are recognized:**

#### imageformat=[raw|fullraw|qcow2]

raw is a generic sparse file that allocates storage on demand fullraw is a generic, non-sparse file that preallocates all space qcow2 is a sparse, copy-on-write capable format implemented at the virtualization layer rather than the filesystem level

#### clonemethod=[qemu-img|relink]

qemu-img allows use of qcow2 to generate virtualization layer copy-on-write relink uses a generic filesystem facility to clone the files on your behalf, but requires filesystem support such as btrfs

#### placement\_affinity=[migratable|user\_migratable|pinned]

### vmvncport (vm.vncport)

Tracks the current VNC display port (currently not meant to be set

### webport (websrv.port)

The port of the web service.

### xcatmaster (noderes.xcatmaster)

The hostname of the xCAT service node (as known by this node). This acts as the default value for `nfsserver` and `tftpserver`, if they are not set. If `xcatmaster` is not set, the node will use whoever responds to its boot request as its master. For the directed bootp case for POWER, it will use the management node if `xcatmaster` is not set.

**zonename** (nodelist.zonename)

The name of the zone to which the node is currently assigned. If undefined, then it is not assigned to any zone.

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## notification.7

### NAME

**notification** - a logical object definition in the xCAT database.

### SYNOPSIS

**notification Attributes:** *comments, filename, tableops, tables*

### DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

#### **notification Attributes:**

**comments** (notification.comments)

Any user-written notes.

**filename** (notification.filename)

The path name of a file that implements the callback routine when the monitored table changes. Can be a perl module or a command. See the `regnotif` man page for details.

**tableops** (notification.tableops)

Specifies the table operation to monitor for. Valid values: “d” (rows deleted), “a” (rows added), “u” (rows updated).

**tables** (notification.tables)

Comma-separated list of xCAT database tables to monitor.



## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## osdistro.7

## NAME

**osdistro** - a logical object definition in the xCAT database.

## SYNOPSIS

**osdistro Attributes:** *arch, basename, dirpaths, majorversion, minorversion, osdistroname, type*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### osdistro Attributes:

**arch** (osdistro.arch)

The OS distro arch (e.g. x86\_64)

**basename** (osdistro.basename)

The OS base name (e.g. rhels)

**dirpaths** (osdistro.dirpaths)

Directory paths where OS distro is store. There could be multiple paths if OS distro has more than one ISO image. (e.g. /install/rhels6.2/x86\_64,...)

**majorversion** (osdistro.majorversion)

The OS distro major version.(e.g. 6)

**minorversion** (osdistro.minorversion)

The OS distro minor version. (e.g. 2)

**osdistroname** (osdistro.osdistroname)

Unique name (e.g. rhels6.2-x86\_64)

**type** (osdistro.type)

Linux or AIX

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## osdistroudate.7

## NAME

**osdistroudate** - a logical object definition in the xCAT database.

## SYNOPSIS

**osdistroudate Attributes:** *dirpath*, *downloadtime*, *osdistroname*, *osupdatename*, *usercomment*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### osdistroudate Attributes:

**dirpath** (osdistroudate.dirpath)

Path to where OS distro update is stored. (e.g. /install/osdistroudates/rhels6.2-x86\_64-20120716-update)

**downloadtime** (osdistroudate.downloadtime)

The timestamp when OS distro update was downloaded..

**osdistroname** (osdistroudate.osdistroname)

The OS distro name to update. (e.g. rhels)

**osupdatename** (osdistroudate.osupdatename)

Name of OS update. (e.g. rhn-update1)

**usercomment** (osdistroudate.comments)

Any user-written notes.

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## osimage.7

### NAME

**osimage** - a logical object definition in the xCAT database.

### SYNOPSIS

**osimage Attributes:** *addkcmdline, boottarget, bosinst\_data, cfmdir, configdump, crashkernelsize, description, driverupdatesrc, dump, environvar, exlist, fb\_script, groups, home, image\_data, imagename, imagetype, installp\_bundle, installto, isdeletable, kerneldir, kernelver, kitcomponents, krpmver, lpp\_source, mkysyb, netdrivers, nimmethod, nimtype, nodebootif, osarch, osdistrname, osname, osupdatenname, osvers, otherifce, otherpkgdir, otherpkglist, otherpkgs, paging, partitionfile, permission, pkgdir, pkglist, postbootscripts, postinstall, postscripts, profile, provmethod, resolv\_conf, root, rootfstype, rootimgdir, script, serverrole, shared\_home, shared\_root, spot, synclists, template, tmp, usercomment, winpepath*

### DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

#### osimage Attributes:

**addkcmdline** (linuximage.addkcmdline)

User specified kernel options for os provision process(no prefix) or the provisioned os(with prefix "R::"). The options should be delimited with spaces(" "). This attribute is ignored if linuximage.boottarget is set.

**boottarget** (linuximage.boottarget)

The name of the boottarget definition. When this attribute is set, xCAT will use the kernel, initrd and kernel params defined in the boottarget definition instead of the default.

**bosinst\_data** (nimimage.bosinst\_data)

The name of a NIM bosinst\_data resource.

**cfmdir** (osimage.cfmdir)

CFM directory name for PCM. Set to /install/osimages/<osimage name>/cfmdir by PCM.

**configdump** (nimimage.configdump)

Specifies the type of system dump to be collected. The values are selective, full, and none. The default is selective.

**crashkernelsize** (linuximage.crashkernelsize)

the size that assigned to the kdump kernel. If the kernel size is not set, 256M will be the default value.

**description** (osimage.description)

OS Image Description

**driverupdatesrc** (linuximage.driverupdatesrc)

The source of the drivers which need to be loaded during the boot. Two types of driver update source are supported: Driver update disk and Driver rpm package. The value for this attribute should be comma separated sources. Each source should be the format `tab:full_path_of_source_file`. The tab keyword can be: `dud` (for Driver update disk) and `rpm` (for driver rpm). If missing the tab, the rpm format is the default. e.g. `dud:/install/dud/dd.img,rpm:/install/rpm/d.rpm`

**dump** (`linuximage.dump`, `nimimage.dump`)

The NFS directory to hold the Linux kernel dump file (`vmcore`) when the node with this image crashes, its format is "`nfs://<nfs_server_ip>/<kdump_path>`". If you want to use the node's "xcatmaster" (its SN or MN), `<nfs_server_ip>` can be left blank. For example, "`nfs:///<kdump_path>`" means the NFS directory to hold the kernel dump file is on the node's SN, or MN if there's no SN.

or

The name of the NIM dump resource.

**environvar** (`osimage.environvar`)

Comma delimited environment variables for the `osimage`

**exlist** (`linuximage.exlist`)

The fully qualified name of the file that stores the file names and directory names that will be excluded from the image during `packimage` command. It is used for diskless image only.

**fb\_script** (`nimimage.fb_script`)

The name of a NIM `fb_script` resource.

**groups** (`osimage.groups`)

A comma-delimited list of image groups of which this image is a member. Image groups can be used in the `litfile` and `litetree` table instead of a single image name. Group names are arbitrary.

**home** (`nimimage.home`)

The name of the NIM home resource.

**image\_data** (`nimimage.image_data`)

The name of a NIM `image_data` resource.

**imagename** (`osimage.imagename`)

The name of this xCAT OS image definition.

**imagetype** (`osimage.imagetype`)

The type of operating system image this definition represents (`linux`, `AIX`).

**installp\_bundle** (`nimimage.installp_bundle`)

One or more comma separated NIM `installp_bundle` resources.

**installto** (`winimage.installto`)

The disk and partition that the Windows will be deployed to. The valid format is `<disk>:<partition>`. If not set, default value is 0:1 for bios boot mode(legacy) and 0:3 for uefi boot mode; If setting to 1, it means 1:1 for bios boot and 1:3 for uefi boot

**isdeletable** (`osimage.isdeletable`)

A flag to indicate whether this image profile can be deleted. This attribute is only used by PCM.

**kerneldir** (`linuximage.kerneldir`)

The directory name where the 3rd-party kernel is stored. It is used for diskless image only.

**kernelver** (linuximage.kernelver)

The version of linux kernel used in the linux image. If the kernel version is not set, the default kernel in rootimgdir will be used

**kitcomponents** (osimage.kitcomponents)

List of Kit Component IDs assigned to this OS Image definition.

**krpmver** (linuximage.krpmver)

The rpm version of kernel packages (for SLES only). If it is not set, the default rpm version of kernel packages will be used.

**lpp\_source** (nimimage.lpp\_source)

The name of the NIM lpp\_source resource.

**mksysb** (nimimage.mksysb)

The name of a NIM mksysb resource.

**netdrivers** (linuximage.netdrivers)

The ethernet device drivers of the nodes which will use this linux image, at least the device driver for the nodes' installnic should be included

**nimmethod** (nimimage.nimmethod)

The NIM install method to use, (ex. rte, mksysb).

**nimtype** (nimimage.nimtype)

The NIM client type- standalone, diskless, or dataless.

**nodebootif** (linuximage.nodebootif)

The network interface the stateless/statelite node will boot over (e.g. eth0)

**osarch** (osimage.osarch)

The hardware architecture of this node. Valid values: x86\_64, ppc64, x86, ia64.

**osdistrname** (osimage.osdistrname)

The name of the OS distro definition. This attribute can be used to specify which OS distro to use, instead of using the osname,osvers,and osarch attributes. For \*kit commands, the attribute will be used to read the osdistro table for the osname, osvers, and osarch attributes. If defined, the osname, osvers, and osarch attributes defined in the osimage table will be ignored.

**osname** (osimage.osname)

Operating system name- AIX or Linux.

**osupdatename** (osimage.osupdatename)

A comma-separated list of OS distro updates to apply to this osimage.

**osvers** (osimage.osvers)

The Linux operating system deployed on this node. Valid values: rhels\*,rhelc\*, rhas\*,centos\*,alma\*, rocky\*,SL\*, fedora\*, sles\* (where \* is the version #).

**otherifce** (linuximage.otherifce)

Other network interfaces (e.g. eth1) in the image that should be configured via DHCP

**otherpkgdir** (linuximage.otherpkgdir)

The base directory and urls of internet repos from which the non-distro packages are retrived. Only 1 local directory is supported at present. The entries should be delimited with comma “,”. Currently, the internet repos are only supported on Ubuntu and Redhat.

**otherpkglist** (linuximage.otherpkglist)

The fully qualified name of the file that stores non-distro package lists that will be included in the image. It could be set to multiple paths. The multiple paths must be separated by “,”.

**otherpkgs** (nimimage.otherpkgs)

One or more comma separated installp or rpm packages. The rpm packages must have a prefix of ‘R:’, (ex. R:foo.rpm)

**paging** (nimimage.paging)

The name of the NIM paging resource.

**partitionfile** (linuximage.partitionfile, winimage.partitionfile)

**Only available for diskful osimages and statelite osimages(localdisk enabled). The full path of the partition file or the script to generate the partition file. The valid value includes:**

“<the absolute path of the parititon file>”: For diskful osimages, the partition file contains the partition definition that will be inserted directly into the template file for os installation. The syntax and format of the partition file should confirm to the corresponding OS installer of the Linux distributions(e.g. kickstart for RedHat, autoyast for SLES, pressed for Ubuntu). For statelite osimages, when the localdisk is enabled, the partition file with specific syntax and format includes the partition scheme of the local disk, please refer to the statelite documentation for details. “s:<the absolute path of the partition script>”: a shell script to generate the partition file “/tmp/partitionfile” inside the installer before the installation start. “d:<the absolute path of the disk name file>”: only available for ubuntu osimages, includes the name(s) of the disks to partition in traditional, non-devfs format(e.g. /dev/sdx, not e.g. /dev/discs/disc0/disc), and be delimited with space. All the disks involved in the partition file should be specified. “s:d:<the absolute path of the disk script>”: only available for ubuntu osimages, a script to generate the disk name file “/tmp/xcat.install\_disk” inside the debian installer. This script is run in the “pressed/early\_command” section. “c:<the absolute path of the additional pressed config file>”: only availbe for ubuntu osimages, contains the additional pressed entries in “d-i ...” form. This can be used to specify some additional preseed options to support RAID or LVM in Ubuntu. “s:c:<the absolute path of the additional pressed config script>”: only available for ubuntu osimages, runs in pressed/early\_command and set the preseed values with “debconf-set”. The multiple values should be delimited with comma “,”

or

The path of partition configuration file. Since the partition configuration for bios boot mode and uefi boot mode are different, this configuration file can include both configurations if you need to support both bios and uefi mode. Either way, you must specify the boot mode in the configuration. Example of partition configuration file: [BIOS]xxxxxxx[UEFI]yyyyyyy. To simplify the setting, you also can set installto in partitionfile with section like [INSTALLTO]0:1

**permission** (linuximage.permission)

The mount permission of /.statelite directory is used, its default value is 755

**pkgdir** (linuximage.pkgdir)

The name of the directory where the distro packages are stored. It could be set to multiple paths. The multiple paths must be separated by “,”. The first path in the value of osimage.pkgdir must be the OS base pkg dir path, such as pkgdir=/install/rhels6.2/x86\_64,/install/updates . In the os base pkg path, there are default repository data. And in the other pkg path(s), the users should make sure there are repository data. If not, use “createrepo” command to create them. For ubuntu, multiple mirrors can be specified in

the pkgdir attribute, the mirrors must be prefixed by the protocol(http/ssh) and delimited with “,” between each other.

#### **pkglist** (linuximage.pkglist)

The fully qualified name of the file that stores the distro packages list that will be included in the image. Make sure that if the pkgs in the pkglist have dependency pkgs, the dependency pkgs should be found in one of the pkgdir

#### **postbootscripts** (osimage.postbootscripts)

Comma separated list of scripts that should be run on this after diskful installation or diskless boot. On AIX these scripts are run during the processing of /etc/inittab. On Linux they are run at the init.d time. xCAT automatically adds the scripts in the xcatdefaults.postbootscripts attribute to run first in the list. See the site table runbootscripts attribute.

#### **postinstall** (linuximage.postinstall)

Supported in diskless image only. The fully qualified name of the scripts and the user-specified arguments running in non-chroot mode after the package installation but before initrd generation during genimage. If multiple scripts are specified, they should be separated with comma “,”. The arguments passed to each postinstall script include 4 implicit arguments(<rootimage path>,<os version>,<os arch>,<profile>) and the user-specified arguments. A set of osimage attributes are exported as the environment variables to be used in the postinstall scripts:

```
IMG_ARCH(The architecture of the osimage, such as "ppc64le","x86_64"),
IMG_NAME(The name of the osimage, such as "rhels7.3-ppc64le-netboot-compute"),
IMG_OSVER(The os release of the osimage, such as "rhels7.3","sles11.4"),
IMG_KERNELVERSION(the "kernelver" attribute of the osimage),
IMG_PROFILE(the profile of the osimage, such as "service","compute"),
IMG_PKGLIST(the "pkglist" attribute of the osimage),
IMG_PKGDIR(the "pkgdir" attribute of the osimage),
IMG_OTHERPKGLIST(the "otherpkglist" attribute of the osimage),
IMG_OTHERPKGDIR(the "otherpkgdir" attribute of the osimage),
IMG_ROOTIMGDIR(the "rootimgdir" attribute of the osimage)
```

#### **postscripts** (osimage.postscripts)

Comma separated list of scripts that should be run on this image after diskful installation or diskless boot. For installation of RedHat, CentOS, Fedora, the scripts will be run before the reboot. For installation of SLES, the scripts will be run after the reboot but before the init.d process. For diskless deployment, the scripts will be run at the init.d time, and xCAT will automatically add the list of scripts from the postbootscripts attribute to run after postscripts list. For installation of AIX, the scripts will run after the reboot and acts the same as the postbootscripts attribute. For AIX, use the postbootscripts attribute. See the site table runbootscripts attribute.

#### **profile** (osimage.profile)

The node usage category. For example compute, service.

#### **provmethod** (osimage.provmethod)

The provisioning method for node deployment. The valid values are install, netboot,statelite,boottarget,dualboot,sysclone. If boottarget is set, you must set linuximage.boottarget to the name of the boottarget definition. It is not used by AIX.

#### **resolv\_conf** (nimimage.resolv\_conf)

The name of the NIM resolv\_conf resource.

#### **root** (nimimage.root)

The name of the NIM root resource.

**rootfstype** (osimage.rootfstype)

The filesystem type for the rootfs is used when the provmethod is statelite. The valid values are nfs or ramdisk. The default value is nfs

**rootimgdir** (linuximage.rootimgdir)

The directory name where the image is stored. It is generally used for diskless image. it also can be used in sysclone environment to specify where the image captured from golden client is stored. in sysclone environment, rootimgdir is generally assigned to some default value by xcat, but you can specify your own store directory. just one thing need to be noticed, wherever you save the image, the name of last level directory must be the name of image. for example, if your image name is testimage and you want to save this image under home directoy, rootimgdir should be assigned to value /home/testimage/

**script** (nimimage.script)

The name of a NIM script resource.

**serverrole** (osimage.serverrole)

The role of the server created by this osimage. Default roles: mgtnode, servicenode, compute, login, storage, utility.

**shared\_home** (nimimage.shared\_home)

The name of the NIM shared\_home resource.

**shared\_root** (nimimage.shared\_root)

A shared\_root resource represents a directory that can be used as a / (root) directory by one or more diskless clients.

**spot** (nimimage.spot)

The name of the NIM SPOT resource.

**synclists** (osimage.synclists)

The fully qualified name of a file containing a list of files to synchronize on the nodes. Can be a comma separated list of multiple synclist files. The synclist generated by PCM named /install/osimages/<imagename>/synclist.cfm is reserved for use only by PCM and should not be edited by the admin.

**template** (linuximage.template, winimage.template)

The fully qualified name of the template file that will be used to create the OS installer configuration file for stateful installations (e.g. kickstart for RedHat, autoyast for SLES).

or

The fully qualified name of the template file that is used to create the windows unattend.xml file for diskful installation.

**tmp** (nimimage.tmp)

The name of the NIM tmp resource.

**usercomment** (linuximage.comments, nimimage.comments)

Any user-written notes.

or

Any user-provided notes.



**winpepath** (winimage.winpepath)

The path of winpe which will be used to boot this image. If the real path is /tftpboot/winboot/winpe1/, the value for winpepath should be set to winboot/winpe1

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## pdu.7

## NAME

**pdu** - a logical object definition in the xCAT database.

## SYNOPSIS

**pdu Attributes:** *node*, *nodetype*, *outlet*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### pdu Attributes:

**node** (pdu.node)

The hostname/address of the pdu to which the settings apply

**nodetype** (pdu.nodetype)

The node type should be pdu

**outlet** (pdu.outlet)

The pdu outlet count

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## policy.7

### NAME

**policy** - a logical object definition in the xCAT database.

### SYNOPSIS

**policy Attributes:** *commands, host, name, noderange, parameters, priority, rule, time, usercomment*

### DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

#### policy Attributes:

**commands** (policy.commands)

The list of commands that this rule applies to. Default is "\*" (all commands).

**host** (policy.host)

The host from which users may issue the commands specified by this rule. Default is "\*" (all hosts). Only all or one host is supported

**name** (policy.name)

The username that is allowed to perform the commands specified by this rule. Default is "\*" (all users).

**noderange** (policy.noderange)

The Noderange that this rule applies to. Default is "\*" (all nodes). Not supported with the \*def commands.

**parameters** (policy.parameters)

A regular expression that matches the command parameters (everything except the noderange) that this rule applies to. Default is "\*" (all parameters). Not supported with the \*def commands.

**priority** (policy.priority)

The priority value for this rule. This value is used to identify this policy data object (i.e. this rule) The table is sorted on this field with the lower the number the higher the priority. For example 1.0 is higher priority than 4.1 is higher than 4.9.

**rule** (policy.rule)

Specifies how this rule should be applied. Valid values are: allow, trusted. Allow will allow the user to run the commands. Any other value will deny the user access to the commands. Trusted means that once this client has been authenticated via the certificate, all other information that is sent (e.g. the username) is believed without question. This authorization should only be given to the xcatd on the management node at this time.

**time** (policy.time)

Time ranges that this command may be executed in. This is not supported.

**usercomment** (policy.comments)

Any user-written notes.

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

**rack.7**

## NAME

**rack** - a logical object definition in the xCAT database.

## SYNOPSIS

**rack Attributes:** *displayname, height, num, rackname, room, usercomment*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### rack Attributes:

**displayname** (rack.displayname)

Alternative name for rack. Only used by PCM.

**height** (rack.height)

Number of units which can be stored in the rack.

**num** (rack.num)

The rack number.

**rackname** (rack.rackname)

The rack name.

**room** (rack.room)

The room in which the rack is located.

**usercomment** (rack.comments)

Any user-written notes.

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

## route.7

## NAME

**route** - a logical object definition in the xCAT database.

## SYNOPSIS

**route Attributes:** *gateway, ifname, mask, net, routename, usercomment*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### route Attributes:

**gateway** (routes.gateway)

The gateway that routes the ip traffic from the mn to the nodes. It is usually a service node.

**ifname** (routes.ifname)

The interface name that facing the gateway. It is optional for IPv4 routes, but it is required for IPv6 routes.

**mask** (routes.mask)

The network mask.

**net** (routes.net)

The network address.

**routename** (routes.routename)

Name used to identify this route.

**usercomment** (routes.comments)

Any user-written notes.

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

**site.7**

## NAME

**site** - a logical object definition in the xCAT database.

## SYNOPSIS

**site Attributes:** *installdir*, *master*, *xcatdport*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### site Attributes:

**installdir** (site.value)

The installation directory

**master** (site.value)

The management node

**xcatdport** (site.value)

Port used by xcatd daemon on master

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

**taskstate.7**

## NAME

**taskstate** - a logical object definition in the xCAT database.

## SYNOPSIS

**taskstate Attributes:** *command, disable, node, pid, reserve, state*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### **taskstate Attributes:**

**command** (taskstate.command)

Current command is running

**disable** (taskstate.disable)

Set to 'yes' or '1' to comment out this row.

**node** (taskstate.node)

The node name.

**pid** (taskstate.pid)

The process id of the request process.

**reserve** (taskstate.reserve)

used to lock the node

**state** (taskstate.state)

The task state(callback, running) for the node.

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

**zone.7**

## NAME

**zone** - a logical object definition in the xCAT database.

## SYNOPSIS

**zone Attributes:** *defaultzone, sshbetweennodes, sshkeydir, usercomment, zonenumber*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

### zone Attributes:

**defaultzone** (zone.defaultzone)

If nodes are not assigned to any other zone, they will default to this zone. If value is set to yes or 1.

**sshbetweennodes** (zone.sshbetweennodes)

Indicates whether passwordless ssh will be setup between the nodes of this zone. Values are yes/1 or no/0. Default is yes.

**sshkeydir** (zone.sshkeydir)

Directory containing the shared root ssh RSA keys.

**usercomment** (zone.comments)

Any user-provided notes.

**zonename** (zone.zonenumber)

The name of the zone.

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

**zvmivp.7**

## NAME

**zvmivp** - a logical object definition in the xCAT database.

## SYNOPSIS

**zvmivp Attributes:** \*\*

## DESCRIPTION

Logical objects of this type are stored in the xCAT database in one or more tables. Use the following commands to manipulate the objects: **mkdef**, **chdef**, **lsdef**, and **rmdef**. These commands will take care of knowing which tables the object attributes should be stored in. The attribute list below shows, in parentheses, what tables each attribute is stored in.

**zvmivp Attributes:**

## SEE ALSO

**mkdef(1)**, **chdef(1)**, **lsdef(1)**, **rmdef(1)**

**man8**

**chtab.8**

## NAME

**chtab** - Add, delete or update rows in the database tables.

## SYNOPSIS

**chtab** [-h | --help]

**chtab** [-v | --version]

**chtab** *keycolname=keyvalue[,keycolname=keyvalue,...]* *tablename.colname=newvalue* [*table-name.colname=newvalue ...*]

**chtab** *keycolname=keyvalue[,keycolname=keyvalue,...]* *tablename.colname+=newvalue* [*table-name.colname+=newvalue ...*]

**chtab -d** *keycolname=keyvalue[,keycolname=keyvalue,...]* *tablename* [*tablename ...*]

## DESCRIPTION

The **chtab** command adds, deletes or updates the attribute values in the specified table.column for the specified *keyvalue*. Normally, the given value will completely replace the current attribute value. But if “+=” is used instead of “=”, the specified value will be appended to the comma separated list of attributes, if it is not already there.

The **chtab** does not pass through xcatd, so it is not controlled by the policy mechanism.



## OPTIONS

- h** Display usage message.
- v** Command Version.
- d** Delete option.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## EXAMPLES

1. To add a node=node1 to the nodelist table with groups=all:

```
chtab node=node1 nodelist.groups=all
```

2. To add a keyword (tftpdirc) and value (/tftpboot) to the site table:

```
chtab key=tftpdirc site.value=/tftpboot
```

3. To add node1 to the nodetype table with os=rhel5:

```
chtab node=node1 nodetype.os=rhel5
```

4. To change node1 in nodetype table setting os=sles:

```
chtab node=node1 nodetype.os=sles
```

5. To change node1 by appending otherpkgs to the postbootscripts field in the postscripts table:

```
chtab node=node1 postscripts.postbootscripts+=otherpkgs
```

6. To delete node1 from nodetype table:

```
chtab -d node=node1 nodetype
```

## FILES

/opt/xcat/bin/chtab

## SEE ALSO

tabdump(8)|tabdump.8, tabedit(8)|tabedit.8

## copycds-cdrom.8

## SYNOPSIS

**copycds-cdrom** [*copycds options*] [*drive*]

## DESCRIPTION

**copycds-cdrom** is a wrapper scripts for **copycds** to copy from physical CD/DVD-ROM drives located on the management server.

[*copycds options*] are passed unchanged to copycds.

If [*drive*] is not specified, /dev/cdrom is assumed.

The copycds command copies all contents of Distribution CDs or Service Pack CDs to the install directory as designated in the **site** table attribute: **installdir**.

## SEE ALSO

copycds(8)|copycds.8

## AUTHOR

Isaac Freeman <ifreeman@us.ibm.com>

## copycds.8

## NAME

**copycds** - Copies Linux distributions and service levels from DVDs/ISOs to the xCAT /install directory.

## SYNOPSIS

**copycds** [{-n|--name|--osver} *distroname*] [{-a|--arch} *architecture*] [{-p|--path} *ospkgpath*] [-o | --noosimage] [-w | --nonoverwrite] {*iso* | *device-path*} ...

**copycds** [-i | --inspection] {*iso* | *device-path*}

**copycds** [-h | --help]

## DESCRIPTION

The **copycds** command copies all contents of Distribution DVDs/ISOs or Service Pack DVDs/ISOs to a destination directory. The destination directory can be specified by the **-p** option. If no path is specified, the default destination directory will be formed from the **installdir** site table attribute, distro name and architecture, for example: `/install/rhels6.3/x86_64`. The **copycds** command can copy from one or more ISO files, or the CD/DVD device path.

You can specify **-i** or **--inspection** option to check whether the DVDs/ISOs can be recognized by xCAT. If recognized, the distribution name, architecture and the disc no (the disc sequence number of DVDs/ISOs in multi-disk distribution) of the DVD/ISO is displayed. If xCAT doesn't recognize the DVD/ISO, you must manually specify the distro name and architecture using the **-n** and **-a** options. This is sometimes the case for distros that have very recently been released, and the xCAT code hasn't been updated for it yet.

You can get xCAT to recognize new DVDs/ISOs by adding them to `/opt/xcat/lib/perl/xCAT/data/discinfo.pm` (the key of the hash is the first line of `.discinfo`) and reloading `xcatsd` (**service xcatsd reload**).

## OPTIONS

**-n|--name|--osver** *distroname*

The linux distro name and version that the ISO/DVD contains. Examples: `rhels6.3`, `sles11.2`, `fedora9`. Note the 's' in `rhels6.3` which denotes the Server version of RHEL, which is typically used.

**-a|--arch** *architecture*

The architecture of the linux distro on the ISO/DVD. Examples: `x86`, `x86_64`, `ppc64`, `s390x`.

**-p|--path** *ospkgpath*

The destination directory to which the contents of ISO/DVD will be copied. When this option is not specified, the default destination directory will be formed from the **installdir** site table attribute and the distro name and architecture, for example: `/install/rhel6.3/x86_64`. This option is only supported for distributions of `sles`, `redhat` and `windows`.

**-i|--inspection**

Check whether xCAT can recognize the DVDs/ISOs in the argument list, but do not copy the disc. Displays the os distribution name, architecture and disc no of each recognized DVD/ISO. This option is only supported for distributions of `sles`, `redhat` and `windows`.

**-o|--noosimage**

Do not create the default osimages based on the osdistro copied in. By default, **copycds** will create a set of osimages based on the osdistro.

**-w|--nonoverwrite**

Complain and exit if the os disc has already been copied in. By default, **copycds** will overwrite the os disc already copied in.

## RETURN VALUE

0: The command completed successfully. For the **--inspection** option, the ISO/DVD have been recognized successfully

Nonzero: An Error has occurred. For the **--inspection** option, the ISO/DVD cannot be recognized

## EXAMPLES

1. To copy the RPMs from a set of ISOs that represent the DVDs of a distro:

```
copycds dvd1.iso dvd2.iso
```

2. To copy the RPMs from a physical DVD to /depot/kits/3 directory:

```
copycds -p /depot/kits/3 /dev/dvd
```

3. To copy the RPMs from a DVD ISO of a very recently released distro:

```
copycds -n rhels6.4 -a x86_64 dvd.iso
```

4. To check whether a DVD ISO can be recognized by xCAT and display the recognized disc info:

```
copycds -i /media/RHEL/6.2/RHEL6.2-20111117.0-Server-ppc64-DVD1.iso
```

Output will be similar to:

```
OS Image:/media/RHEL/6.2/RHEL6.2-20111117.0-Server-ppc64-DVD1.iso
DISTNAME:rhels6.2
ARCH:ppc64
DISCNO:1
```

For the attributes that are not recognized, the value will be blank.

5. To copy the packages from a supplemental DVD ISO file:

```
copycds /isodir/RHEL6.5/RHEL6.5-Supplementary-20131114.2-Server-ppc64-DVD1.iso ↵
↵-n rhels6.5-supp
```

Also, remember to add the new directory to your osimage definition:

```
chdef -t osimage myosimage -p pkgdir=/install/rhels6.5-supp/ppc64
```

## SEE ALSO

nodeset(8)|nodeset.8, site(5)|site.5, nodetype(5)|nodetype.5

## makeconservercf.8

### NAME

**makeconservercf** - creates the consver configuration file from info in the xCAT database

### SYNOPSIS

**makeconservercf** [-V|--verbose] [-d|--delete] [*noderange*]

**makeconservercf** [-V|--verbose] [-C|--cleanup]

**makeconservercf** [-V|--verbose] [-l|--local] [*noderange*]

**makeconservercf** [-V|--verbose] [-c|--conserver] [*noderange*]

**makeconservercf** [-V|--verbose] *noderange* [-t|--trust] *hosts*

**makeconservercf** [-h|--help|-v|--version]

### DESCRIPTION

The **makeconservercf** command will write out the `/etc/conserver.cf`, using information from the `nodehm` table and related tables (e.g. `mp`, `ipmi`, `ppc`). Normally, **makeconservercf** will write all nodes to the `/etc/conserver.cf` file. If a *noderange* is specified, it will write only those nodes to the file. In either case, if a node does not have `nodehm.cons` set, it will not be written to the file.

If **-d** is specified, **makeconservercf** will remove specified nodes from `/etc/conserver.cf` file. If *noderange* is not specified, all xCAT nodes will be removed from `/etc/conserver.cf` file.

If **-C|--cleanup** is specified, **makeconservercf** will remove console configuration entries from `/etc/conserver.cf` for the nodes whose definitions have been removed from xCATdb. **Don't** specify any *noderange*.

In the case of a hierarchical cluster (i.e. one with service nodes) **makeconservercf** will determine which nodes will have their consoles accessed from the management node and which from a service node (based on the `nodehm.conserver` attribute). The `/etc/conserver.cf` file will be created accordingly on all relevant management/service nodes. If **-l** is specified, it will only create the local file.

### OPTIONS

#### **-d|--delete**

Delete rather than add or refresh the nodes specified as a *noderange*.

#### **-C|--cleanup**

Remove the entries for the nodes whose definitions have been removed from xCAT db.

#### **-c|--conserver**

Only set up the consver on the consver host. If no consver host is set for nodes, the consver gets set up only on the management node.

#### **-l|--local**

Only run **makeconservercf** locally and create the local `/etc/conserver.cf`. The default is to also run it on all service nodes, if there are any.

**-t|--trust *hosts***

Add additional trusted hosts into `/etc/conserver.cf`. The *hosts* are comma separated list of ip addresses or host names.

**-v|--version**

Display version.

**-V|--verbose**

Verbose mode.

**-h|--help**

Display usage message.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## EXAMPLES

1. To create consver configuration for all the nodes.

```
makeconservercf
```

2. To create consver configuration for nodes node01-node10.

```
makeconservercf node01-node10
```

3. To remove consver configuration for node01.

```
makeconservercf -d node01
```

## SEE ALSO

`rcons(1)`|`rcons.1`

## **makedhcp.8**

## NAME

**makedhcp** - Creates and updates DHCP configuration files.

## SYNOPSIS

```

makedhcp -n [-l | --localonly]
makedhcp -a [-l | --localonly]
makedhcp -a -d [-l | --localonly]
makedhcp -d noderange [-l | --localonly]
makedhcp noderange [-s statements] [-l | --localonly]
makedhcp -q noderange
makedhcp [-h|--help]

```

## DESCRIPTION

The **makedhcp** command creates and updates the DHCP configuration on the management node and service nodes. The **makedhcp** command is supported for both Linux and AIX clusters.

1. Start by filling out the networks(5)|networks.5 table properly.
2. Then use the **makedhcp -n** option to create a new dhcp configuration file. You can set the site table, dhcplease attribute to the lease time for the dhcp client. The default value is 43200.
3. Next, get the node IP addresses and MACs defined in the xCAT database. Also, get the hostnames and IP addresses pushed to /etc/hosts (using makehosts(8)|makehosts.8) and to DNS (using makedns(8)|makedns.8).
4. Then run **makedhcp** with a *noderange* or the **-a** option. This will inject into dhcpd configuration data pertinent to the specified nodes. On linux, the configuration information immediately takes effect without a restart of DHCP.

If you need to delete node entries from the DHCP configuration, use the **-d** flag.

## OPTIONS

### **-n**

Create a new dhcp configuration file with a network statement for each network the dhcp daemon should listen on. (Which networks dhcpd should listen on can be controlled by the dhcpinterfaces attribute in the site(5)|site.5 table.) The **makedhcp** command will automatically restart the dhcp daemon after this operation. This option will replace any existing configuration file (making a backup of it first). For Linux systems the file will include network entries as well as certain general parameters such as a dynamic range and omapi configuration. For AIX systems the file will include network entries. On AIX systems, if there are any non-xCAT entries in the existing configuration file they will be preserved and added to the end of the new configuration file.

### **-a**

Define all nodes to the DHCP server. (Will only add nodes that can be reached, network-wise, by this DHCP server.) The dhcp daemon does not have to be restarted after this. On AIX systems **makedhcp** will not add entries for cluster nodes that will be installed using NIM. The entries for these nodes will be managed by NIM.

### *noderange*

Add the specified nodes to the DHCP server configuration.

### **-s** *statements*

For the input *noderange*, the argument will be interpreted like dhcp configuration file text.

**-d *noderange***

Delete node entries from the DHCP server configuration. On AIX, any entries created by NIM will not be removed.

**-a -d**

Delete all node entries, that were added by xCAT, from the DHCP server configuration.

**-l | --localonly**

Configure dhcpd on the local machine only. Without this option, makedhcp will also send this operation to any service nodes that service the nodes in the noderange.

**-q *noderange***

Query the node entries from the DHCP server configuration. On AIX, any entries created by NIM will not be listed.

**-h|--help**

Display usage message.

**RETURN VALUE**

0. The command completed successfully.
1. An error has occurred.

**EXAMPLES**

1. Create a new DHCP configuration file and add the network definitions:

```
makedhcp -n
```

2. Define all nodes to the dhcp server:

```
makedhcp -a
```

Note: This does not add nodes that will be installed with AIX/NIM.

3. Will cause dhcp on the next request to set root-path appropriately for only node5. Note some characters (e.g. “) must be doubly escaped (once for the shell, and once for the OMAPI layer).

```
makedhcp node5 -s 'option root-path \"172.16.0.1:/install/freebsd6.2/x86_64\"';
```

4. Query a node from the DHCP server.

```
# makedhcp -q node01  
node01: ip-address = 91.214.34.156, hardware-address = 00:00:c9:c6:6c:42
```



## FILES

DHCP configuration files:

[AIX] /etc/dhcpsd.cnf

[SLES] /etc/dhcpd.conf

[RH] /etc/dhcp/dhcpd.conf

## SEE ALSO

noderange(3)|noderange.3

## makedns.8

## NAME

**makedns** - sets up domain name services (DNS).

## SYNOPSIS

**makedns** [-h | --help]

**makedns** [-V | --verbose] [-e | --external] [-n | --new] [*noderange*]

**makedns** [-V | --verbose] [-e | --external] [-d | --delete *noderange*]

## DESCRIPTION

**makedns** configures a DNS server on the system you run it on, which is typically the xCAT management node.

The list of nodes to include comes from either the **noderange** provided on the command line or the entries in the local */etc/hosts* files.

There are several bits of information that must be included in the xCAT database before running this command.

You must set the **forwarders** attributes in the xCAT **site** definition.

The **forwarders** value should be set to the IP address of one or more nameservers at your site that can resolve names outside of your cluster. With this set up, all nodes ask the local nameserver to resolve names, and if it is a name that the MN DNS does not know about, it will try the forwarder names.

An xCAT **network** definition must be defined for each network used in the cluster. The **net** and **mask** attributes will be used by the **makedns** command.

A network **domain** and **nameservers** values must be provided either in the **network** definition corresponding to the node or in the **site** definition.

Only entries in */etc/hosts* or the hosts specified by **noderange** that have a corresponding xCAT network definition will be added to DNS.

By default, **makedns** sets up the **named** service and updates the DNS records on the local system (management node). If the **-e** flag is specified, it will also update the DNS records on any external DNS server that is listed in the */etc/resolv.conf* on the management node. (Assuming the external DNS server can recognize the xCAT key as authentication.)

For more information on Cluster Name Resolution see [https://xcat-docs.readthedocs.io/en/stable/advanced/domain\\_name\\_resolution/domain\\_name\\_resolution.html](https://xcat-docs.readthedocs.io/en/stable/advanced/domain_name_resolution/domain_name_resolution.html)

## OPTIONS

### **-V | --verbose**

Verbose mode.

### **-n | --new**

Use this flag to create new named configuration and db files.

### **-d | --delete**

Remove the DNS records.

### **-e | --external**

Update DNS records to the external DNS server listed in */etc/resolv.conf*.

Enabling the site attribute *externaldns* means use ‘external’ DNS by default. If setting *externaldns* to 1, you need NOT use **-e** flag in every **makedns** call.

### *noderange*

A set of comma delimited node names and/or group names. See the “noderange” man page for details on additional supported formats.

## Examples

1. To set up DNS for all the hosts in */etc/hosts* file.

```
makedns
```

2. To set up DNS for *node1*.

```
makedns node1
```

3. To create a new named configuration and db files for all hosts in */etc/hosts*.

```
makedns -n
```

4. To delete the DNS records for *node1*.

```
makedns -d node1
```

## SEE ALSO

makehosts(8)|makehosts.8

## makegocons.8

### NAME

**makegocons** - Register or unregister the node in the goconserver service

### SYNOPSIS

**makegocons** [-V|--verbose] [-d|--delete] [-q|--query] [*noderange*]

**makegocons** [-V|--verbose] [-C|--cleanup]

### DESCRIPTION

In order to use *goconserver* instead of *conserver*, reference <https://xcat-docs.readthedocs.io/en/latest/advanced/goconserver/quickstart.html> for more information.

The **makegocons** command will start the goconserver service if it is not started, then send the REST request to create or delete the session resource in the goconserver service. The session information including the session command or ssh connection parameters (for openbmc) is generated by xcat based on the records in the related tables (e.g. nodehm, ipmi, ppc, openbmc).

By default **makegocons** will register the session for all of the nodes in xcat.

If a *noderange* is specified, only the session in the specified scope will be affected, goconserver service will not be restarted and the other session will not be disconnected. This is the advantage of goconserver over the consver service with **makeconservercf**.

If **-d** is specified, **makegocons** will remove the session in the goconserver service.

In the case of a hierarchical cluster (i.e. one with service nodes) **makegocons** will determine which nodes will have their consoles accessed from the management node and which from a service node (based on the nodehm.conserver attribute).

To start goconserver on the specified service node, setup goconserver package on that service node, then set the **console** column of **servicenode** table to **2**.

To support diskless service node, a new column **consoleenabled** has been added in **nodehm** table, it is used by **makegocons** command to save the current console state for the node. After reinstalling the service node, the console storage file which maintain the console nodes by goconserver is lost, xCAT would register the console nodes into goconserver based on **consoleenabled** attribute when restarting xcatd service.

For openbmc which uses ssh as the terminal session connection method, goconserver can help save the system resources as goconserver could handle the ssh connection within goroutine which is more light-weighted than the command process.

**Note:** goconserver only support the systemd based systems. It has been integrated with xCAT as a recommended package.

## OPTIONS

### **-d|--delete**

Delete rather than add or refresh the nodes specified as a noderange.

### **-C|--cleanup**

Remove the entries for the nodes whose definitions have been removed from xCAT db.

### **-q|--query**

List the console connection of the nodes. If noderange is not specified, all of the console nodes will be displayed.

### **-v|--version**

Display version.

### **-V|--verbose**

Verbose mode.

### **-h|--help**

Display usage message.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## EXAMPLES

1. To create gocons server configuration for all the nodes.

```
makegocons
```

2. To create gocons server configuration for nodes node01-node10.

```
makegocons node01-node10
```

3. To remove gocons server configuration for node01.

```
makegocons -d node01
```

4. To list console connection for node01.

```
makegocons -q node01
```

## SEE ALSO

rcons(1)|rcons.1

## makehosts.8

## NAME

**makehosts** - sets up /etc/hosts from the xCAT hosts table.

## SYNOPSIS

**makehosts** [-n] [*noderange*] [-l | --longnamefirst] [-d] [-m | --mactolinklocal]

**makehosts** {-h | --help}

## DESCRIPTION

**makehosts** updates the /etc/hosts file based on information stored in the xCAT database object definitions.

The main three bits of information needed are: node hostname, node ip and network domain name.

The hostname and ip address are specified as part of the node definition.

The domain value is taken either from the xCAT network definition associated with the node or from the cluster site definition. If you are using multiple domains in the cluster you should add the domain names to the appropriate xCAT network definition.

Note: If your node hostnames and IP addresses follow a regular pattern, you can use just a few regular expressions to generate /etc/hosts using makehosts. For details on using regular expressions see the “xcatdb” man page.

If you specify additional network interfaces in your xCAT node definitions they will also be added to the /etc/hosts file. You can specify additional network interface information (NICs) using the following node attributes: nicips, nichostnamesuffixes, nictypes, niccustomscripts, nicnetworks. You can get a description of these attributes by running “lsdef -t node -h | more” or “man nics”.

## OPTIONS

### -n

Completely replace the /etc/hosts file, losing any previous content. If this option is not specified, it will only replace the lines in the file that correspond to the nodes in the specified noderange.

### -l | --longnamefirst

The FQDN (Fully Qualified Domain Name) of the host will appear before the PQDN (Partially Qualified Domain Name) for each host in the /etc/hosts file. The default is PQDN first. After xCAT is installed, the attribute name “FQDNfirst” can be added into “site” table manually. If the value is set as “1”, “yes” or “enable”, the /etc/hosts entries generated by “makehosts” will put the FQDN before the PQDN. Otherwise, the original behavior will be performed.

### -m | --mactolinklocal

Updates /etc/hosts file with IPv6 link local addresses, the link local address is generated from the mac address stored in mac table.

**-d**

Delete rather than create records. This will also delete any additional network interfaces (NICs) included in the node definitions.

**EXAMPLES**

1. Add entries to /etc/hosts for all nodes included in the xCAT node group called “compute”.

```
makehosts compute
```

2. If the xCAT hosts table contains:

```
"compute", " |node(\d+) | 1.2.3.($1+0) | ", " |(.*) | ($1).cluster.net | ", ,
```

Assuming the group “compute” contains node01, node02, etc., then in /etc/hosts they will be given IP addresses of 1.2.3.1, 1.2.3.2, etc.

**SEE ALSO**

hosts(5)|hosts.5, makedns(8)|makedns.8

**makeknownhosts.8****NAME**

**makeknownhosts** - Make a known\_hosts file under \$ROOTHOME/.ssh for input noderange.

**SYNOPSIS**

**makeknownhosts** *noderange* [-r | --remove | -d | --delete] [-V | --verbose]

**makeknownhosts** [-h | --help]

**DESCRIPTION**

**makeknownhosts** Replaces or removes entries for the nodes in the known\_hosts file in the \$ROOTHOME/.ssh directory. The known\_hosts file entry is built from the shared ssh host key that xCAT distributes to the installed nodes.

HMCs, AMM, switches, etc., where xCAT does not distribute the shared ssh host key, should not be put in the noderange.

To build the known\_hosts entry for a node, you are only required to have the node in the database, and name resolution working for the node. You do not have to be able to access the node.

Having this file with correct entries, will avoid the ssh warning when nodes are automatically added to the known\_hosts file. The file should be distributed using **xdcp** to all the nodes, if you want node to node communication not to display the warning.

## OPTIONS

### *noderange*

A set of comma delimited node names and/or group names. See the *noderange* man page for details on supported formats.

### **-d|--delete**

Only removes the entries for the nodes from the known\_hosts file.

### **-r|--remove**

Synonymous to **-d|--delete**.

### **-V|--verbose**

Verbose mode.

## EXAMPLES

1. To build the known\_hosts entry for the nodes in the compute group

```
makeknownhosts compute
```

2. To build the known\_hosts entry for the nodes in the lpars and service groups

```
makeknownhosts lpars,service
```

3. To remove the known\_hosts entry for node02

```
makeknownhosts node02 -d
```

## makenetworks.8

## NAME

**makenetworks** - Gather cluster network information and add it to the xCAT database.

## SYNOPSIS

**makenetworks** [-h | --help ]

**makenetworks** [-v | --version]

**makenetworks** [-V | --verbose] [-d | --display]

## DESCRIPTION

The **makenetworks** command can be used to gather network information from an xCAT cluster environment and create corresponding network definitions in the xCAT database.

Every network that will be used to install a cluster node must be defined in the xCAT database.

The default behavior is to gather network information from the management node, and any configured xCAT service nodes, and automatically save that information in the xCAT database.

You can use the **-d** option to display the network information without writing it to the database.

You can also redirect the output to a file that can be used with the xCAT **mkdef** command to define the networks.

For example:

```
makenetworks -d > mynetstanzas  
  
cat mynetstanzas | mkdef -z
```

This features allows you to verify and modify the network information before writing it to the database.

When the network information is gathered a default value is created for the “netname” attribute. This is done to make it possible to use the **mkdef**, **chdef**, **lsdef**, and **rmdef** commands to manage this data.

The default naming convention is to use a hyphen separated “net” and “mask” values with the “.” replaced by “\_”. (ex. “8\_124\_47\_64-255\_255\_255\_0”)

You can also modify the xCAT “networks” database table directly using the xCAT **tabedit** command.

```
tabedit networks
```

Note: The **makenetworks** command is run automatically when xCAT is installed on a Linux management node.

## OPTIONS

**-d|--display** Display the network definitions but do not write to the definitions to the xCAT database. The output will be in stanza file format and can be redirected to a stanza file that can be used with **mkdef** or **chdef** commands to create or modify the network definitions.

**-h | --help** Display usage message.

**-v | --version** Command Version.

**-V | --verbose** Verbose mode.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.



## EXAMPLES

1. Gather cluster network information and create xCAT network definitions.

```
makenetworks
```

2. Display cluster network information but do not write the network definitions to the xCAT database.

```
makenetworks -d
```

The output would be one or more stanzas of information similar to the following. The line that ends with a colon is the value of the “netname” attribute and is the name of the network object to use with the **lsdef**, **mkdef**, **chdef** and **rmdef** commands.

```
9_114_37_0-255_255_255_0:
  objtype=network
  gateway=9.114.37.254
  mask=255.255.255.0
  net=9.114.37.0
  mgtifname=ens3
  mtu=1500
```

## FILES

/opt/xcat/sbin/makenetworks

## SEE ALSO

makedhcp(8)|makedhcp.8

## makeroutes.8

## NAME

**makeroutes** - add or delete routes to/from the os route table on nodes.

## SYNOPSIS

```
makeroutes [-r | --routename r1[,r2...]]
```

```
makeroutes [-d | --delete] [-r | --routenames r1[,r2...]]
```

```
makeroutes noderange [-r | --routename r1[,r2...]]
```

```
makeroutes noderange [-d | --delete] [-r | --routenames r1[,r2...]]
```

```
makeroutes [-h | --help | -v | --version]
```

## DESCRIPTION

The **makeroutes** command adds or deletes routes on the management node or any given nodes. The **noderange** specifies the nodes where the routes are to be added or removed. When the *noderange* is omitted, the action will be done on the management node. The **-r** option specifies the name of routes. The details of the routes are defined in the **routes** table which contains the route name, subnet, net mask and gateway. If **-r** option is omitted, the names of the routes found on **noderes.routenames** for the nodes or on **site.mnroutenames** for the management node will be used.

If you want the routes be automatically setup during node deployment, first put a list of route names to **noderes.routenames** and then add *setroute* script name to the **postscripts.postbootscripts** for the nodes.

## Parameters

*noderange* specifies the nodes where the routes are to be added or removed. If omitted, the operation will be done on the management node.

## OPTIONS

### **-d|--delete**

Specifies to delete the given routes. If not specified, the action is to add routes.

### **-r|--routename**

Specifies a list of comma separated route names defined in the **routes** table. If omitted, all routes defined in **noderes.routenames** for nodes or **site.mnroutenames** for the management node will be used.

### **-h|--help**

Display usage message.

### **-v|--version**

Command Version.

## EXAMPLES

1. To add all routes from the **site.mnroutenames** to the os route table for the management node.

```
makeroutes
```

2. To add all the routes from **noderes.routenames** to the os route table for node1.

```
makeroutes node1
```

3. To add route rr1 and rr2 to the os route table for the management node.

```
makeroutes -r rr1,rr2
```

4. To delete route rr1 and rr2 from the os route table on node1 and node1.

```
makeroutes node1,node2 -d -r rr1,rr2
```

## FILES

/opt/xcat/sbin/makeroutes

## SEE ALSO

**mknb.8**

## NAME

**mknb** - creates a network boot root image for node discovery and flashing

## SYNOPSIS

**mknb** *arch*

## DESCRIPTION

The **mknb** command is run by xCAT automatically when xCAT is installed on the management node. It creates a network boot root image (used for node discovery, BMC programming, and flashing) for the same architecture that the management node is. So you normally do not need to run the **mknb** command yourself.

If you make custom changes to the network boot root image, you will need to run **mknb** again to regenerate the diskless image to include your changes. If you have an xCAT Hierarchical Cluster with Service Nodes having local /tftpboot directories (site.sharedtftp=0), you will need to copy the generated root image to each Service Node.

Presently, the architectures x86\_64 and ppc64 are supported. For ppc64le, use the ppc64 architecture.

## OPTIONS

*arch*

The hardware architecture for which to build the boot image.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## SEE ALSO

makedhcp(8)|makedhcp.8

## nodeadd.8

## NAME

**nodeadd** - Adds nodes to the xCAT cluster database.

## SYNOPSIS

**nodeadd** *noderange* **groups**=*groupnames* [*table.column=value*] [...]

**nodeadd** [-v | --version]

**nodeadd** [-? | -h | --help]

## DESCRIPTION

The **nodeadd** command adds the nodes specified in *noderange* to the xCAT database. It also stores the any additional attributes specified for these nodes. At least one *groupname* must be supplied. You should also consider specifying attributes in at least the following tables: **nodehm**, **noderes**, **nodetype**. See the man page for each of these for details. Also see the **xcatdb** man page for an overview of each table.

The **nodeadd** command also supports some short cut names as aliases to common attributes. See the **models** man page for details.

## OPTIONS

**-v|--version**

Command Version.

**-?|-h|--help**

Display usage message.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## EXAMPLES

1. To add nodes in noderange node1-node4 with group all:

```
nodeadd node1-node4 groups=all
```

2. To add nodes in noderange node1-node4 to the nodetype table with os=rhel5:

```
nodeadd node1-node4 groups=all,rhel5 nodetype.os=rhel5
```

## FILES

/opt/xcat/bin/nodeadd

## SEE ALSO

nodes(1)|nodes.1, nodech(1)|nodech.1, noderange(3)|noderange.3

## nodeset.8

### Name

**nodeset** - set the boot state for a noderange

### Synopsis

**nodeset** *noderange* [**boot** | **stat** [-a]] **offline** | **runcmd**=*command* | **osimage**[=*imagename*] | **shell** | **shutdown**] [-V | --verbose]

**nodeset** *noderange* **osimage**[=*imagename*] [--noupdateinitrd] [--ignorekernelchk]

**nodeset** *noderange* **runimage**=*task*

**nodeset** [-h | --help | -v | --version]

### Description

**nodeset** sets the next boot state for a single or range of nodes or groups. It tells xCAT what you want to happen the next time the nodes are booted up. See noderange(3)|noderange.3. **nodeset** accomplishes this by changing the network boot files. Each xCAT node always boots from the network and downloads a boot file with instructions on what action to take next.

**nodeset** will manipulate the boot configuration files of xnba, grub2, petitboot, yaboot and pxelinux.0.

Assume that /tftpboot is the root for tftpd (set in site(5)|site.5).

**nodeset** for petitboot makes changes to /tftpboot/petitboot/{node name}

**nodeset** for xnba makes changes to /tftpboot/xcat/xnba/nodes/{node name}

**nodeset** for grub2 makes changes to /tftpboot/boot/grub2/{node name}

**nodeset** for pxe makes changes to /tftpboot/pxelinux.cfg/{node hex ip}

**nodeset** for yaboot makes changes to /tftpboot/etc/{node hex ip}

**nodeset** only sets the next boot state, but does not reboot.

**nodeset** is called by **rinstall** and **winstall** and is also called by the installation process remotely to set the boot state back to “boot”.

In a hierarchical cluster managed by service nodes, **nodeset** command is used to make sure compute node states are consistent on service and management nodes. When errors are reported, run the command with verbose mode. And the command will display additional service node information, which might be useful in identifying the problem.

A user can supply their own scripts to be run on the mn or on the service node (if a hierarchical cluster) for a node when the nodeset command is run. Such scripts are called **prescripts**. They should be copied to /install/prescripts directory. A table called *prescripts* is used to specify the scripts and their associated actions. The scripts to be run at the beginning of the nodeset command are stored in the ‘begin’ column of *prescripts* table. The scripts to be run at the end of the nodeset command are stored in the ‘end’ column of *prescripts* table. You can run ‘tabdump -d prescripts’ command for details. The following two environment variables will be passed to each script: NODES contains all the names of the nodes that need to run the script for and ACTION contains the current nodeset action. If *#xCAT setting:MAX\_INSTANCE=number* is specified in the script, the script will get invoked for each node in parallel, but no more than *number* of instances will be invoked at a time. If it is not specified, the script will be invoked once for all the nodes.

## Options

### **boot**

Instruct network boot loader to be skipped, generally meaning boot to hard disk

### **offline**

Cleanup the current pxe/tftp boot configuration files for the nodes requested

### **osimage | osimage=imagename**

Prepare server for installing a node using the specified os image. The os image is defined in the *osimage* table and *linuximage* table. If the <imagename> is omitted, the os image name will be obtained from *nodetype.provmethod* for the node.

### **--nouupdateinitrd**

Skip the rebuilding of initrd when the ‘netdrivers’, ‘driverupdatesrc’ or ‘osupdatename’ were set for injecting new drivers to initrd. But, the **geninitrd** command should be run to rebuild the initrd for new drivers injecting. This is used to improve the performance of **nodeset** command.

### **--ignorekernelchk**

Skip the kernel version checking when injecting drivers from osimage.driverupdatesrc. That means all drivers from osimage.driverupdatesrc will be injected to initrd for the specific target kernel.

### **runimage=task**

If you would like to run a task after deployment, you can define that task with this attribute.

### **stat**

Display the current boot loader config file description for the nodes requested. When **disjointdhcps** is set, using **-a** to display them on all available service nodes.

### **runcmd=command**

This instructs the node to boot to the xCAT genesis environment and specified command to be executed.

### **shell**

This instructs the node to boot to the xCAT genesis environment, and present a shell prompt on console. The node will also be able to be sshed into and have utilities such as wget, tftp, scp, nfs, and cifs. It will have storage drivers available for many common systems.

### shutdown

To make the node to get into power off status. This status only can be used after **runcmd** and **runimage** to power off the node after the performing of operations.

### -V | --verbose

Verbose mode.

### -h | --help

Print help.

### -v | --version

Print version.

## Files

**noderes** table - xCAT node resources file. See `noderes(5)|noderes.5` for further details.

**nodetype** table - xCAT node installation type file. See `nodetype(5)|nodetype.5` for further details. This is used to determine the node installation image type.

**site** table - xCAT main configuration file. See `site(5)|site.5` for further details. This is used to determine the location of the TFTP root directory and the TFTP xCAT subdirectory. `/tftpboot` and `/tftpboot/xcat` is the default.

## Examples

1. To setup to install `mycomputeimage` on the compute node group.

```
nodeset compute osimage=mycomputeimage
```

2. To run `http://protect/T1/textdollar/master/image.tgz` after deployment:

```
nodeset $node runimage=http://$MASTER/image.tgz
```

3. Boot `node1` into xCAT genesis environment and execute `bmcsetup` script. This causes the IP, netmask, gateway, username, and password to be programmed according to the configuration in node object definition.

```
rinstall node1 runcmd=bmcsetup
```

## See Also

`noderange(3)|noderange.3`, `nodels(1)|nodels.1`, `nodestat(1)|nodestat.1`, `rinstall(8)|rinstall.8`, `makedhcp(8)|makedhcp.8`, `osimage(7)|osimage.7`

## rescanplugins.8

### NAME

**rescanplugins** - Notifies xcatd to rescan the plugin directory

### SYNOPSIS

**rescanplugins**

**rescanplugins** [-h | --help]

**rescanplugins** [-v | --version]

**rescanplugins** [-s | --servicenodes]

### DESCRIPTION

**rescanplugins** notifies the xcatd daemon to rescan the plugin directory and update its internal command handlers hash. This command should be used when plugins have been added or removed from the xCAT plugin directory (/opt/xcat/lib/perl/xCAT\_plugin) or if the contents of the handled\_commands subroutine in an existing plugin has changed.

If rescanplugins is called as a subrequest from another command, the xcatd command handlers hash changes will not be available to that command's process. Only subsequent command calls will see the updates.

### OPTIONS

**-h|--help**

Displays the usage message.

**-v|--version**

Displays the release version of the code.

**-s|--servicenodes**

Process the rescanplugins on the management node and on all service nodes. The rescanplugins command will be sent to the xcatd daemon on all nodes defined in the servicenode table. The default is to only run on the management node.

### EXAMPLES

1. To rescan the plugins only on the xCAT Management Node:

```
rescanplugins
```

2. To rescan the plugins on the xCAT Management Node and on all service nodes:

```
rescanplugins -s
```



## rinstall.8

### Name

**rinstall** - Begin OS provision on a noderange

### Synopsis

**rinstall** *noderange* [**boot** | **shell** | **runcmd=command**] [**-c** | **--console**] [**-V** | **--verbose**]

**rinstall** *noderange* [**osimage=imagename**] [**--nouupdateinitrd**][**--ignorekernelchk**] [**-c** | **--console**] [**-u** | **--uefimode**] [**-V** | **--verbose**]

**rinstall** *noderange* **runimage=task**

**rinstall** [**-h** | **--help** | **-v** | **--version**]

### Description

**rinstall** is a convenience command to begin OS provision on a noderange.

If **osimage=imagename** is specified or **osimage** is specified and `nodetype.provmethod=osimage` is set, provision the noderange with the osimage specified/configured. If no task specified, default is **osimage**.

If **-c** is specified, **rinstall** will run **rcons** on the node. This is allowed only if one node is in the noderange. If consoles are needed on multiple nodes, see `winstall(8)|winstall.8`.

### Options

#### **boot**

Instruct network boot loader to be skipped, generally meaning boot to hard disk

#### **osimage=imagename**

Prepare server for installing a node using the specified OS image. The OS image is defined in the *osimage* table and *linuximage* table. If the *imagename* is omitted, the OS image name will be obtained from *nodetype.provmethod* for the node.

#### **--nouupdateinitrd**

Skip the rebuilding of `initrd` when the 'netdrivers', 'driverupdatesrc' or 'osupdatename' were set for injecting new drivers to `initrd`. But, the **geninitrd** command should be run to rebuild the `initrd` for new drivers injecting. This is used to improve the performance of **rinstall** command.

#### **--ignorekernelchk**

Skip the kernel version checking when injecting drivers from `osimage.driverupdatesrc`. That means all drivers from `osimage.driverupdatesrc` will be injected to `initrd` for the specific target kernel.

#### **runimage=task**

If you would like to run a task after deployment, you can define that task with this attribute.

#### **runcmd=command**

This instructs the node to boot to the xCAT genesis environment and specified command to be executed.

#### **shell**

This instructs the node to boot to the xCAT genesis environment, and present a shell prompt on console. The node will also be able to be sshed into and have utilities such as wget, tftp, scp, nfs, and cifs. It will have storage drivers available for many common systems.

**-h | --help**

Display usage message.

**-v | --version**

Display version.

**-u | --uefimode**

For BMC-based servers, to specify the next boot mode to be “UEFI Mode”.

**-V | --verbose**

Verbose output.

**-c | --console**

Requests that **rinstall** runs **rcons** once the provision starts. This will only work if there is only one node in the noderange. See `winstall(8)|winstall.8` for starting consoles on multiple nodes.

## Examples

1. Provision nodes 1 through 20, using their current configuration.

```
rinstall node1-node20
```

2. Provision nodes 1 through 20 with the osimage rhels6.4-ppc64-netboot-compute.

```
rinstall node1-node20 osimage=rhels6.4-ppc64-netboot-compute
```

3. Provision node1 and start a console to monitor the process.

```
rinstall node1 -c
```

4. Boot node1 into xCAT genesis environment and execute bmcsetup script. This causes the IP, netmask, gateway, username, and password to be programmed according to the configuration in node object definition.

```
rinstall node1 runcmd=bmcsetup
```

## See Also

`noderange(3)|noderange.3`, `winstall(8)|winstall.8`, `rcons(1)|rcons.1`

## rmosdistro.8

### SYNOPSIS

```
rmosdistro [-a | --all] [-f | --force] osdistroname [osdistroname2 ...]
```

```
rmosdistro [-h | --help]
```

### DESCRIPTION

The **rmosdistro** command removes the specified OS Distro that was created by **copycds**. To delete all OS Distro entries, please specify [-a|--all]. If the specified OS Distro is referenced by some osimage, [-f|--force] can be used to remove it.

### ARGUMENTS

The OS Distro names to delete, delimited by blank space.

### OPTIONS

**-a | --all**

If specified, try to delete all the OS Distros.

**-f | --force**

Remove referenced OS Distros, never prompt.

**-h | --help**

Show info of rmosdistro usage.

### RETURN VALUE

**Zero:**

The command completed successfully.

**Nonzero:**

An Error has occurred.

### EXAMPLES

1. To remove OS Distro “rhels6.2-ppc64” and “sles11.2-ppc64”:

```
rmosdistro rhels6.2-ppc64 sles11.2-ppc64
```

2. To remove OS Distro “rhels6.2-ppc64”, regardless of whether is referenced by any osimage:

```
rmosdistro -f rhels6.2-ppc64
```

3. To remove all OS Distros:

```
rmosdistro -a
```

## runsqlcmd.8

### NAME

**runsqlcmd** -Runs sql command files against the current xCAT database.

### SYNOPSIS

**runsqlcmd**

**runsqlcmd** [-h | --help]

**runsqlcmd** [-v | --version]

**runsqlcmd** [-d | --dir *directory\_path*] [-V | --verbose]

**runsqlcmd** [-f | --files *list of files*] [-V | --verbose]

**runsqlcmd** [-V | --verbose] [*sql statement*]

### DESCRIPTION

The runsqlcmd routine, runs the sql statements contained in the \*.sql files as input to the command against the current running xCAT database. Only DB2,MySQL and PostgreSQL databases are supported. SQLite is not supported. If no directory or filelist is provided, the default /opt/xcats/lib/perl/xcats\_schema directory is used. If the directory is input with the -d flag, that directory will be used. If a comma separated list of files is input with the -f flag, those files will be used.

### OPTIONS

**-h|--help**

Displays the usage message.

**-v|--version**

Displays current code version.

**-V|--verbose**

Displays extra debug information.

**-d|--dir**

To use a directory other than the default directory, enter the directory path here.

**-f|--files**

Comma separated list of files (full path), wildcard (\*) can be used.

### File format

The files must be of the form <name>.sql or <name>\_<database>.sql where <database> is mysql,pgsql, or db2. Files must have permission 0755.

*sql statement*

Quoted sql statement syntax appropriate for the current database.

## EXAMPLES

1. To run the database appropriate \*.sql files in /opt/xcat/lib/perl/xcAT\_schema :

```
runsqlcmd
```

2. To run the database appropriate \*.sql files in /tmp/mysql:

```
runsqlcmd -d /tmp/mysql
```

3. To run the database appropriate \*.sql files in the input list:

```
runsqlcmd -f "/tmp/mysql/test*,/tmp/mysql/test1*"
```

4. To checkout one DB2 sql file:

```
runsqlcmd -f /tmp/db2/test_db2.sql
```

5. To run the following command to the database:

```
runsqlcmd "Select * from site;"
```

## setupiscsidev.8

### NAME

**setupiscsidev** - creates a LUN for a node to boot up with, using iSCSI

### SYNOPSIS

**setupiscsidev** [-s|--size] *noderange*

**setupiscsidev** [-h|--help|-v|--version]

### DESCRIPTION

The **setupiscsidev** command will create a LUN on the management node (or service node) for each node specified. The LUN device can then be used by the node as an iSCSI device so the node can boot diskless, stateful.

## OPTIONS

### **-s|--size**

The size of the LUN that should be created. Default is 4096.

### **-v|--version**

Display version.

### **-h|--help**

Display usage message.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## SEE ALSO

nodeset(8)|nodeset.8

## tabch.8

## NAME

**tabch** - Add, delete or update rows in the database tables.

## SYNOPSIS

**tabch** [-h | --help]

**tabch** [-v | --version]

**tabch** *keycolname=keyvalue[,keycolname=keyvalue,...]* *tablename.colname=newvalue* [*table-name.colname=newvalue ...*]

**tabch** *keycolname=keyvalue[,keycolname=keyvalue,...]* *tablename.colname+=newvalue* [*table-name.colname+=newvalue ...*]

**tabch -d** *keycolname=keyvalue[,keycolname=keyvalue,...]* *tablename* [*tablename ...*]

## DESCRIPTION

The **tabch** command adds, deletes or updates the attribute values in the specified table.column for the specified key-value. The difference between **tabch** and **chtab** is **tabch** runs as a plugin under the xcatd daemon. This give the additional security of being authorized by the daemon. Normally, the given value will completely replace the current attribute value. But if “+=” is used instead of “=”, the specified value will be appended to the comma separated list of attributes, if it is not already there.

## OPTIONS

**-h|--help** Display usage message.

**-v|--version** Command Version.

**-d** Delete option.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## EXAMPLES

1. To add a node=node1 to the nodelist table with groups=all:

```
tabch node=node1 nodelist.groups=all
```

2. To add a keyword (tftpdir) and value (/tftpboot) to the site table:

```
tabch key=tftpdir site.value=/tftpboot
```

3. To add node1 to the nodetype table with os=rhel5:

```
tabch node=node1 nodetype.os=rhel5
```

4. To change node1 in nodetype table setting os=sles:

```
tabch node=node1 nodetype.os=sles
```

5. To change node1 by appending otherpkgs to the postbootscripts field in the postscripts table:

```
tabch node=node1 postscripts.postbootscripts+=otherpkgs
```

6. To delete node1 from nodetype table:

```
tabch -d node=node1 nodetype
```

## FILES

/opt/xcat/sbin/tabch

## SEE ALSO

tabdump(8)|tabdump.8, tabedit(8)|tabedit.8

## tabdump.8

## NAME

**tabdump** - display an xCAT database table in CSV format.

## SYNOPSIS

**tabdump**

**tabdump** [-d] [*table*]

**tabdump** [*table*]

**tabdump** [-f *filename*] [*table*]

**tabdump** [-n *# of records*] [**auditlog** | **eventlog**]

**tabdump** [-w *attr==val*] [-w *attr=~val*] ...] [*table*]

**tabdump** [-w *attr==val*] [-w *attr=~val*] ...] [-f *filename*] [*table*]

**tabdump** [-v | --version]

**tabdump** [-? | -h | --help]

## DESCRIPTION

The **tabdump** command displays the header and all the rows of the specified table in CSV (comma separated values) format. Only one table can be specified. If no table is specified, the list of existing tables will be displayed.

## OPTIONS

**-?|-h|--help**

Display usage message.

**-d**

Show descriptions of the tables, instead of the contents of the tables. If a table name is also specified, descriptions of the columns (attributes) of the table will be displayed. Otherwise, a summary of each table will be displayed.

**-n**

Shows the most recent number of entries as supplied on the -n flag from the auditlog or eventlog table.



## -f

File name or path to file in which to dump the table. Without this the table is dumped to stdout. Using the -f flag allows the table to be dumped one record at a time. If tables are very large, dumping to stdout can cause problems such as running out of memory.

**-w 'attr==val' -w 'attr=~val' ...**

Use one or multiple -w flags to specify the selection string that can be used to select particular rows of the table. See examples.

Operator descriptions:

<b>==</b>	Select nodes where the attribute value is exactly this value.
<b>!=</b>	Select nodes where the attribute value is <b>not</b> this specific value.
<b>&gt;</b>	Select nodes where the attribute value is greater than this <b>specific value</b> .
<b>&gt;=</b>	Select nodes where the attribute value is greater than <b>or</b> equal to <b>this specific value</b> .
<b>&lt;</b>	Select nodes where the attribute value is less than this <b>specific value</b> .
<b>&lt;=</b>	Select nodes where the attribute value is less than <b>or</b> equal to <b>this specific value</b> .
<b>==~</b>	Select nodes where the attribute value matches the SQL LIKE value.
<b>!~</b>	Select nodes where the attribute value matches the SQL NOT LIKE <b>value</b> .

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## EXAMPLES

1. To display the contents of the site table:

```
tabdump site
```

2. To display the contents of the nodelist table where the groups attribute is compute :

```
tabdump -w 'groups==compute' nodelist
```

3. To display the contents of the nodelist table where the groups attribute is comput% where % is a wildcard and can represent any string and the status attribute is booted :

```
tabdump -w 'groups=~comput%' -w 'status==booted' nodelist
```

4. To display the records of the auditlog on date 2011-04-18 11:30:00 :

```
tabdump -w 'audittime==2011-04-18 11:30:00' auditlog
```

5. To display the records of the auditlog starting on 2011-04-18:

```
tabdump -w 'audittime>2011-04-18 11:30:00' auditlog
```

6. To display the 10 most recent entries in the auditlog:

```
tabdump -n 10 auditlog
```

7. To see what tables exist in the xCAT database:

```
tabdump
```

8. To back up all the xCAT database tables, instead of running **tabdump** multiple times, you can use the **dumpxCATdb** command as follows:

```
dumpxCATdb -p /tmp/xcatbak
```

See the **dumpxCATdb** man page for details.

9. To display a summary description of each table:

```
tabdump -d
```

10. To display a description of each column in the nodehm table:

```
tabdump -d nodehm
```

## FILES

/opt/xcat/sbin/tabdump

## SEE ALSO

tabrestore(8)|tabrestore.8, tabedit(8)|tabedit.8, dumpxCATdb(1)|dumpxCATdb.1

## tabedit.8

## NAME

**tabedit** - view an xCAT database table in an editor and make changes.

## SYNOPSIS

**tabedit** *table*

**tabedit** [-? | -h | --help]

## DESCRIPTION

The `tabedit` command opens the specified table in the user's editor, allows them to edit any text, and then writes changes back to the database table. The table is flattened into a CSV (comma separated values) format file before giving it to the editor. After the editor is exited, the CSV file will be translated back into the database format. You may not `tabedit` the `auditlog` or `eventlog` because indexes will be regenerated. Use `tabprune` command to edit `auditlog` and `eventlog`.

## OPTIONS

`-?|-h|--help`

Display usage message.

## ENVIRONMENT VARIABLES

### TABEDITOR

The editor that should be used to edit the table, for example: `vi`, `vim`, `emacs`, `oocalc`, `pico`, `gnumeric`, `nano`. If **TABEDITOR** is not set, the value from **EDITOR** will be used. If **EDITOR** is not set, it will default to `vi`.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## EXAMPLES

1. To edit the `site` table:

```
tabedit site
```

## FILES

`/opt/xcat/sbin/tabedit`

## SEE ALSO

`tabrestore(8)|tabrestore.8`, `tabdump(8)|tabdump.8`, `chtab(8)|chtab.8`

## tabprune.8

### NAME

**tabprune** - Deletes records from the eventlog,auditlog,isnm\_perf,isnm\_perf\_sum tables.

### SYNOPSIS

**tabprune** [**eventlog** | **auditlog**] [**-V**] [**-i** *recid* | **-n** *number of records* | **-p** *percentage* | **-d** *number of days* | **-a**]

**tabprune** *tablename* **-a**

**tabprune** [**-h** | **--help**] [**-v** | **--version**]

### DESCRIPTION

The tabprune command is used to delete records from the auditlog, eventlog, isnm\_perf, isnm\_perf\_sum tables. As an option, the table header and all the rows pruned from the specified table will be displayed in CSV (comma separated values) format. The all records options (-a) can be used on any xCAT table.

### OPTIONS

#### **-h|--help**

Display usage message.

#### **-V**

Verbose mode. This will cause tabprune to display the records that are being deleted from the table, in case you want to redirect them to a file to archive them.

#### **-a**

Remove all records from the input table name. This option can be used on any xCAT table.

#### **-i** *recid number*

Remove the records whose recid is less than the input recid number.

#### **-n** *number*

Remove the number of records input.

#### **-p** *percent*

Remove the number of records input.

#### **-d** *number of days*

Remove all records that occurred >= than number of days ago.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## EXAMPLES

1. To remove all the records in the eventlog table:

```
tabprune eventlog -a
```

2. To remove all the records in the eventlog table saving the deleted records in eventlog.csv:

```
tabprune eventlog -V -a > eventlog.csv
```

3. To remove all the records before recid=200 in the auditlog table:

```
tabprune auditlog -i 200
```

4. To remove 400 records from the auditlog table and display the remove records:

```
tabprune auditlog -V -n 400
```

5. To remove 50% of the eventlog table:

```
tabprune eventlog -p 50
```

6. To remove all records that occurred >= 5 days ago in the eventlog:

```
tabprune eventlog -d 5
```

## FILES

/opt/xcat/sbin/tabprune

## SEE ALSO

tabrestore(8)|tabrestore.8, tabedit(8)|tabedit.8,tabdump(8)|tabdump.8

## tabrestore.8

## NAME

**tabrestore** - replaces with or adds to a xCAT database table the contents in a csv file.

## SYNOPSIS

**tabrestore** [-a] *table.csv*

**tabrestore** [-? | -h | --help]

**tabrestore** [-v | --version]

## DESCRIPTION

The **tabrestore** command reads the contents of the specified file and puts its data in the corresponding table in the xCAT database. Any existing rows in that table are replaced unless the (-a) flag is used and then the rows in the file are added to the table. The file must be in csv format. It could be created by **tabdump**. Only one table can be specified.

This command can be used to copy the example table entries in /opt/xcat/share/xcat/templates/e1350 into the xCAT database.

## OPTIONS

**-?|-h|--help**

Display usage message.

**-v|--version**

Display version.

**-a|--addrows**

Add rows from the CSV file to the table instead of replacing the table with the CSV file.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## EXAMPLES

1. To replace the rows in the mp table with the rows in the mp.csv file:

```
tabrestore mp.csv
```

The file mp.csv could contain something like:

```
#node,mpa,id,comments,disable
"blade","|\D+(\d+)|amm(($1-1)/14+1)|","|\D+(\d+)|((($1-1)%14+1)|",,
```

2. To add the rows in the mp.csv file to the rows in the mp table:

```
tabrestore -a mp.csv
```

3. To restore database tables from restore\_directory that we dumped with dumpxCATdb:

```
restorexCATdb -p restore_directory
```

## FILES

/opt/xcat/sbin/tabrestore

## SEE ALSO

tabdump(8)|tabdump.8, tabedit(8)|tabedit.8, dumpxCATdb(1)|dumpxCATdb.1

## winstall.8

### Name

**winstall** - Begin OS provision on a noderange

### Synopsis

**winstall** *noderange* [**boot** | **shell** | **runcmd=bmcsetup**] [**runimage=task**] [-V | **--verbose**]

**winstall** *noderange* [**osimage=imagename** | *imagename*] [**--ignorekernelchk**] [-u | **--uefimode**] [-V | **--verbose**]

**winstall** [-h | **--help** | -v | **--version**]

### Description

**winstall** is a convenience command to begin OS provision on a noderange.

If **osimage=imagename** | *imagename* is specified or **nodetype.provmethod=osimage** is set, provision the noderange with the osimage specified/configured.

It will then run **wcons** on the noderange.

### Options

#### boot

Instruct network boot loader to be skipped, generally meaning boot to hard disk

*imagename* | **osimage=imagename**

Prepare server for installing a node using the specified os image. The os image is defined in the *osimage* table and *linuximage* table. If the *imagename* is omitted, the os image name will be obtained from *nodetype.provmethod* for the node.

#### --ignorekernelchk

Skip the kernel version checking when injecting drivers from *osimage.driverupdatesrc*. That means all drivers from *osimage.driverupdatesrc* will be injected to *initrd* for the specific target kernel.

**runimage=task**

If you would like to run a task after deployment, you can define that task with this attribute.

**runcmd=bmcsetup**

This instructs the node to boot to the xCAT nbfs environment and proceed to configure BMC for basic remote access. This causes the IP, netmask, gateway, username, and password to be programmed according to the configuration table.

**shell**

This instructs the node to boot to the xCAT genesis environment, and present a shell prompt on console. The node will also be able to be sshed into and have utilities such as wget, tftp, scp, nfs, and cifs. It will have storage drivers available for many common systems.

**-h | --help**

Display usage message.

**-v | --version**

Display version.

**-u | --uefimode**

For BMC-based servers, to specify the next boot mode to be “UEFI Mode”.

**-V | --verbose**

Verbose output.

**Examples**

1. Provision nodes 1 through 20, using their current configuration.

```
winstall node1-node20
```

2. Provision nodes 1 through 20 with the osimage rhels6.4-ppc64-netboot-compute.

```
winstall node1-node20 osimage=rhels6.4-ppc64-netboot-compute
```

**See Also**

noderange(3)|noderange.3, rinstall(8)|rinstall.8, wcons(1)|wcons.1

**xcatconfig.8****NAME**

**xcatconfig** - Sets up the Management Node during the xCAT install.



## SYNOPSIS

**xcatconfig** [-h | --help]

**xcatconfig** [-v | --version]

**xcatconfig** [-i | --initinstall] [-V | --verbose]

**xcatconfig** [-u | --updateinstall] [-V | --verbose]

**xcatconfig** [-k | --sshkeys] [-s | --sshnodehostkeys] [-c | --credentials] [-d | --database] [-m | --mgtnode] [-t | --tunables] [-V | --verbose]

**xcatconfig** [-f | --force] [-V | --verbose]

## DESCRIPTION

**xcatconfig** Performs basic xCAT setup operations on an xCAT management node. This command should not be run on an xCAT Service Node, unless you are making it a Management Node. See flag description below for more details.

## OPTIONS

### **-h|--help**

Displays the usage message.

### **-v|--version**

Displays the release version of the code.

### **-V|--verbose**

Displays verbose messages.

### **-i|--initialinstall**

The install option is normally run as a post operation from the rpm xCAT.spec file during the initial install of xCAT on the Management Node. It will setup the root ssh keys, ssh node keys, xCAT credentials, initialize the database, export directories, start syslog and other daemons as needed after the initial install of xCAT.

### **-u|--updateinstall**

The update install option is normally run as a post operation from the rpm xCAT.spec file during an update install of xCAT on the Management Node. It will check the setup the root ssh keys, ssh node keys, xCAT credentials, database, exported directories, syslog and the state of daemons needed by xCAT, after the updateinstall of xCAT. If setup is required, it will perform the operation. It will restart the necessary daemons.

### **-k|--sshkeys**

This option will remove and regenerate the root id\_rsa keys. It should only be used, if the keys are deleted or corrupted. The keys must then be distributed to the nodes by installing, running updatenode -k, or using xdsh -K option, for root to be able to ssh to the nodes without being prompted for a password. rspconfig will need to be run to distribute the key to the MM and HMCs. Any device, we need to ssh from the MN to the device will also have to be updated with the new ssh keys.

### **-s|--sshnodehostkeys**

This option will remove and regenerate the node host ssh keys. It should only be used, if the keys are deleted or are corrupted. The keys must then be redistribute to the nodes by installing, running `updatenode -k` or using `xdcp` or `pcp` to copy the keys from `/etc/xcat/hostkeys` directory to the `/etc/ssh` directory on the nodes.

**-c|--credentials**

This option will remove all xcat credentials for root and any userids where credentials have been created. It will regenerate roots credentials, but the admin will have to add back all the userid credentials needed with the `/opt/xcat/share/xcat/scripts/setup-local-client.sh <username>` command. It should only be used, if they are deleted or become corrupted. The root credentials must be redistributed to the service nodes by installing the service node or using `updatenode -k`. `makeconservercf` must be rerun to pick up the new credentials, and `conserver` must be stopped and started.

**-d|--database**

This option will reinitialize the basic xCAT database table setup. It will not remove any new database entries that have been added, but it is strongly suggested that you backup you database (`dumpxCATdb`) before using it.

**-f|--force**

The force option may be used after the install to reinitialize the Management Node. This option will regenerate keys, credential and reinitialize the site table. This option should be used, if keys or credentials become corrupt or lost. Additional action must be taken after using the force options. ssh keys must be redistributed to the nodes, site table attributes might need to be restored, `makeconservercf` needs to be rerun to pick up the new credentials and `conserver` stopped and started, `rspconfig` needs to be rerun to distribute the new keys to the MM and the HMCs. A new set of common ssh host keys will have been generated for the nodes. If you wish your nodes to be able to ssh to each other with out password intervention, then you should redistribute these new keys to the nodes. If the nodes hostkeys are updated then you will need to remove their entries from the `known_hosts` files on the management node before using `ssh`, `xdsh`, `xdcp`. Redistribute credentials and ssh keys to the service nodes and ssh keys to the nodes by using the `updatenode -k` command.

**-m|--mgtnode**

This option will add the Management Node to the database with the correct attributes set to be recognized by xCAT. This should be run after the hostname of the Management Node is set to the name that will resolve to the cluster-facing NIC.

**-t|--tunables**

This option will set tunable parameters on the Management and Service nodes recommended for your Linux cluster. It will only set them during initial install, if you run `xcatconfig -f` or `xcatconfig -t`.

**EXAMPLES**

1. To force regeneration of keys and credentials and reinitialize the site table:

```
xcatconfig -f
```

2. To regenerate root's ssh keys:

```
xcatconfig -k
```

3. To regenerate node host ssh keys:

```
xcatconfig -s
```

4. To regenerate node host ssh keys and credentials:

```
xcatconfig -s -c
```

5. To add the Management Node to the DB:

```
xcatconfig -m
```

## xcatd.8

### NAME

**xcatd** - The xCAT daemon

### SYNOPSIS

**xcatd**

### DESCRIPTION

The heart of the xCAT architecture is the xCAT daemon **xcatd** on the management node. This receives requests from the client, validates the requests, and then invokes the operation. The xcatd daemon also receives status and inventory info from the nodes as they are being discovered and installed/booted.

Errors and information are reported through syslog to the /var/log/messages file. You can search for xCAT in those messages.

See <http://xcat-docs.readthedocs.org/en/latest/overview/index.html#xcat-architecture> for more information.

### EXAMPLES

1. To start/stop/restart xcatd on Linux, enter:

```
service xcatd start  
  
service xcatd stop  
  
service xcatd restart
```

2. To start/stop/restart xcatd on AIX, enter:

```
restartxcatd  
  
or  
  
startsrc -s xcatd  
  
stopsrc -s xcatd
```

## FILES

/opt/xcat/sbin/xcatd

## SEE ALSO

**xcatdebug.8**

## NAME

**xcatdebug** - Enable or disable the trace facilities for xCAT. (Only supports Linux Operating System)

## SYNOPSIS

**xcatdebug** { [-f enable | disable [-c {*configuration file* | *subroutine list*}]] | [-d enable | disable]}

## DESCRIPTION

xCAT offers two trace facilities to debug the xCAT:

### Subroutine calling trace

Display the calling trace for subroutine when it is called.

The trace message includes: The name of the called subroutine; The arguments which passed to the called subroutine; The calling stack of the subroutine. By default, the trace will be enabled to all the subroutines in the xcatd and plugin modules. The target subroutine can be configured by configuration file or through xcatdebug command line.

**The flag -c is used to specify the subroutine list for subroutine calling trace, it can only work with -f. The value of -c can be a configuration file or a subroutine list.**

**configuration file:** a file contains multiple lines of **SUBROUTINE\_DEFINITION** **subroutine list:** **SUBROUTINE\_DEFINITION | SUBROUTINE\_DEFINITION|...**

**SUBROUTINE\_DEFINITION:** is the element for the -c to specify the subroutine list.

The format of **SUBROUTINE\_DEFINITION**: [plugin](subroutine1,subroutine2,...)

**If ignoring the [plugin], the subroutines in the () should be defined in the xcatd.**

e.g. (daemonize,do\_installm\_service,do\_udp\_service)

**Otherwise, the package name of the plugin should be specified.**

e.g. xCAT::Utils(isMN,Version) e.g. xCAT\_plugin::DBObjectdefs(defls,process\_request)

The trace log will be written to /var/log/xcat/subcallingtrace. The log file subcallingtrace will be backed up for each running of the **xcatdebug -f enable**.

### Commented trace log

The trace log code is presented as comments in the code of xCAT. In general mode, it will be kept as comments. But in debug mode, it will be commented back as common code to display the trace log.

**NOTE:** This facility can be enabled by pass the **ENABLE\_TRACE\_CODE=1** global variable when running the xcatd.  
e.g. **ENABLE\_TRACE\_CODE=1 xcatd -f**

This facility offers two formats for the trace log code:

### Trace section

```
## TRACE_BEGIN # print "In the debugn"; ## TRACE_END
```

### Trace in a single line

```
## TRACE_LINE print "In the trace linen";
```

The **commented trace log** can be added in xcatd and plugin modules. But following section has been added into the BEGIN {} section of the target plugin module to enable the facility.

```
if (defined $ENV{ENABLE_TRACE_CODE}) {
    use xCAT::Enabletrace qw(loadtrace filter);
    loadtrace();
}
```

## OPTIONS

### -f

Enable or disable the **subroutine calling trace**.

For **enable**, if ignoring the **-c** flag, all the subroutines in the xcatd and plugin modules will be enabled.

For **disable**, all the subroutines which has been enabled by **-f enable** will be disabled. **-c** will be ignored.

### -c

Specify the configuration file or subroutine list.

#### configuration file

[a file contains multiple lines of **SUBROUTINE\_DEFINITION**]

#### e.g.

```
(plugin_command)          xCAT_plugin::DBObjectdefs(defls,process_request)
xCAT::DBObjUtils(getobjdefs)
```

#### subroutine list

[a string like **SUBROUTINE\_DEFINITION | SUBROUTINE\_DEFINITION|...**]

#### e.g.

```
“(plugin_command)|xCAT_plugin::DBObjectdefs(defls,process_request)|xCAT::DBObjUtils(getobjdefs)”
```

### -d

Enable or disable the **commented trace log**.

Note: The xcatd will be restarted for the performing of **-d**

## EXAMPLES

1. Enable the **subroutine calling trace** for all the subroutines in the xcatd and plugin modules.

```
xcatdebug -f enable
```

2. Enable the **subroutine calling trace** for the subroutines configured in the /opt/xcat/share/xcat/samples/tracelevel0

```
xcatdebug -f enable -c /opt/xcat/share/xcat/samples/tracelevel0
```

3. Enable the **subroutine calling trace** for the `plugin_command` in `xcatd` and `defls,process_request` in the `xCAT_plugin::DBobjectdefs` module.

```
xcatdebug -f enable -c "xCAT_plugin::DBobjectdefs(defls,process_
↪request)|(plugin_command)"
```

4. Disable the **subroutine calling trace** for all the subroutines which have been enabled by **xcatdebug -f enable**.

```
xcatdebug -f disable
```

5. Enable the **commented trace log**

```
xcatdebug -d enable
```

6. Enable both the **subroutine calling trace** and **commented trace log**

```
xcatdebug -f enable -c /opt/xcat/share/xcat/samples/tracelevel0 -d enable
```

## **xcatsetup.8**

### **NAME**

**xcatsetup** - Prime the xCAT database using naming conventions specified in a config file.

### **SYNOPSIS**

**xcatsetup** [-s|--stanzas *stanza-list*] [--yesreallydeletenodes] *cluster-config-file*

**xcatsetup** [-? | -h | --help | -v | --version]

### **DESCRIPTION**

The **xcatsetup** command reads the specified config file that contains general information about the cluster being set up, and naming conventions and IP addresses that you want to use. It then defines the basic objects in the xCAT database representing this cluster configuration. The **xcatsetup** command prepares the database for the step of discovering the hardware that is connected to the service and cluster networks. The typical steps of setting up a system p cluster are:

1. Install the xCAT software on the management node
2. Create the cluster config file and run `xcatsetup`
3. Put hardware control passwords in the `ppchcp` or `ppcdirect` database table
4. Run `makenetworks` and `makedhcp`
5. Run the discovery commands (`lsslp`, `mkhwconn`, `rspconfig`) as described in the System P Hardware Management cookbook.
6. Configure and start services using `makehosts`, `makedns`, `mkconserver.cf`, etc.
7. Create the images that should be installed or booted on the nodes
8. Run `nodeset` and `rpower/rnetboot` to boot up the nodes.

The **xcatssetup** command is intended as a quick way to fill out the database for a cluster that has very regular naming patterns. The only thing it does is fill in database attributes. If your cluster does not follow consistent naming patterns, or has some other special configuration, you should define attribute values manually using `mkdef(1)`|`mkdef.1`, instead of using **xcatssetup**. The cluster config file is meant to be an easy way to prime the database; it is not meant to be a long living file that you update as the cluster changes. If you do want to run `xcatssetup` again at a later time, because, for example, you added a lot of nodes, you should put the total list of nodes in the config file, not just the new ones. This is because `xcatssetup` uses some regular expressions for groups (e.g. `frame`, `cec`, `compute`) that would be calculated incorrectly if the config file told `xcatssetup` about only the new nodes.

Speaking of regular expressions, `xcatssetup` creates some pretty complicated regular expressions in the database. These are useful because they keep most of the tables small, even for large clusters. But if you want to tweak them, they may be hard to understand. If after running `xcatssetup`, you want to convert your database to use individual rows for every node, you can do the following:

```
lsdef -z all >tmp.stanza
cat tmp.stanza | chdef -z
```

Many of the sections and attributes in the configuration file can be omitted, if you have a simple cluster, or if you want to create just 1 or 2 of the object types at this time. See the section **A Simpler Configuration File** for an example of this.

If you want to delete all of the nodes that `xcatssetup` created, and start over, use the **--yesreallydeletenodes** option.

## Restrictions

1. The **xcatssetup** command has only been implemented and tested for system p servers so far.

## Configuration File

The **config file** is organized in stanza format and supports the keywords in the sample file below. Comment lines begin with “#”. Stanzas can be omitted if you do not want to define that type of object. The only hostname formats supported are those shown in this sample file, although you can change the base text and the numbers. For example, `hmc1-hmc3` could be changed to `hwmgmt01-hwmgmt12`. The hostnames specified must sort correctly. I.e. use `node01-node80`, instead of `node1-node80`. This sample configuration file is for a 2 building block cluster.

```
xcat-site:
domain = cluster.com
# currently only direct fsp control is supported
use-direct-fsp-control = 1
# ISR network topology. For example, one of the following: 128D, 64D, 32D, 16D, 8D, 4D,
→ 2D, 1D
topology = 32D
# The nameservers in site table will be set with the value of master automatically.

xcat-service-lan:
# IP range used for DHCP. If you set the entry, the networks table will be filled
# automatically with this range and the dhcp interface will be set in the site table.
dhcp-dynamic-range = 50.0.0.0-50.0.0.200

xcat-hmcs:
hostname-range = hmc1-hmc2
starting-ip = 10.200.1.1
```

(continues on next page)

(continued from previous page)

```
xcat-frames:
# these are the connections to the frames
hostname-range = frame[1-6]
num-frames-per-hmc = 3
# this lists which serial numbers go with which frame numbers
vpd-file = vpd2bb.stanza
# There are two rules of defining FSP/BPAs. The first defining the node's host name by
↳increasing the last bit
# of IP address, while the second defining the node's name by varying the second bit and
↳the third bit of IP.
# This assumes you have 2 service LANs: a primary service LAN 10.230.0.0/255.255.0.0
↳that all of the port 0's
# are connected to, and a backup service LAN 10.231.0.0/255.255.0.0 that all of the
↳port 1's are connected to.
# bpa-a-0-starting-ip = 10.230.1.1
# bpa-b-0-starting-ip = 10.230.2.1
# bpa-a-1-starting-ip = 10.231.1.1
# bpa-b-1-starting-ip = 10.231.2.1
# This assumes you have 2 service LANs: a primary service LAN 40.x.y.z/255.0.0.0 that
↳all of the port 0's
# are connected to, and a backup service LAN 41.x.y.z/255.0.0.0 that all of the port 1's
↳are connected to.
# "x" is the frame number and "z" is the bpa/fsp id (1 for the first BPA/FSP in the
↳Frame/CEC, 2 for the
# second BPA/FSP in the Frame/CEC). For BPAs "y" is always be 0 and for FSPs "y" is the
↳cec id.
vlan-1 = 40
vlan-2 = 41

xcat-cecs:
# These are the connections to the CECs. Either form of hostname is supported.
#hostname-range = cec01-cec64
hostname-range = f[1-6]c[01-12]
# If you use the frame/cec hostname scheme above, but do not have a consistent
# number of cecs in each frame, xcat can delete the cecs that do not get
# supernode numbers assigned to them.
delete-unused-cecs = 1
# lists the HFI supernode numbers for each group of cecs in each frame
supernode-list = supernodelist2bb.txt
# If you do not want to specify the supernode-list at this time and you have a
↳consistent
# number of cecs in each frame, you can instead just use this setting:
num-cecs-per-frame = 12
#fsp-a-0-starting-ip = 10.230.3.1
#fsp-b-0-starting-ip = 10.230.4.1
#fsp-a-1-starting-ip = 10.231.3.1
#fsp-b-1-starting-ip = 10.231.4.1

xcat-building-blocks:
num-frames-per-bb = 3
```

(continues on next page)



(continued from previous page)

```

num-cecs-per-bb = 32

xcat-lpars:
  num-lpars-per-cec = 8
  # If you set these, then do not set the corresponding attributes in the other node
  ↪ stanzas below.
  # Except you still need to set xcat-service-nodes:starting-ip (which is the ethernet
  ↪ adapter)
  #hostname-range = f[1-6]c[01-12]p[1-8]
  hostname-range = f[1-6]c[01-12]p[01,05,09,13,17,21,25,29]
  starting-ip = 10.1.1.1
  aliases = -hf0
  # ml0 is for aix. For linux, use bond0 instead.
  otherinterfaces = -hf1:11.1.1.1,-hf2:12.1.1.1,-hf3:13.1.1.1,-ml0:14.1.1.1

xcat-service-nodes:
  num-service-nodes-per-bb = 2
  # which cecs within the bldg block that the SNs are located in
  cec-positions-in-bb = 1,32
  # this is for the ethernet NIC on each SN
  #hostname-range = sn1-sn4
  starting-ip = 10.10.1.1
  # this value is the same format as the hosts.otherinterfaces attribute except
  # the IP addresses are starting IP addresses
  #otherinterfaces = -hf0:10.10.1.1,-hf1:10.11.1.1,-hf2:10.12.1.1,-hf3:10.13.1.1,-ml0:10.
  ↪ 14.1.1

xcat-storage-nodes:
  num-storage-nodes-per-bb = 3
  # which cecs within the bldg block that the storage nodes are located in
  cec-positions-in-bb = 12,20,31
  #hostname-range = stor1-stor6
  #starting-ip = 10.20.1.1
  #aliases = -hf0
  #otherinterfaces = -hf1:10.21.1.1,-hf2:10.22.1.1,-hf3:10.23.1.1,-ml0:10.24.1.1

xcat-compute-nodes:
  #hostname-range = n001-n502
  #starting-ip = 10.30.1.1
  #aliases = -hf0
  # ml0 is for aix. For linux, use bond0 instead.
  #otherinterfaces = -hf1:10.31.1.1,-hf2:10.32.1.1,-hf3:10.33.1.1,-ml0:10.34.1.1

```

## VPD File for Frames

The **vpd-file** specifies the following vpd table attributes for the frames: node, serial, mtm, side. Use the same stanza format that accepted by the `chdef(1)|chdef.1` command, as documented in `xcatstanzafile(5)|xcatstanzafile.5`. The purpose of this file is to enable xCAT to match up frames found through `lsslp(1)|lsslp.1` discovery with the database objects created by **xcatsetup**. All of the frames in the cluster must be specified.

Here is a sample file:

```
frame1:
  objtype=node
  serial=99200G1
  mtm=9A00-100
frame2:
  objtype=node
  serial=99200D1
  mtm=9A00-100
frame3:
  objtype=node
  serial=99200G1
  mtm=9A00-100
frame4:
  objtype=node
  serial=99200D1
  mtm=9A00-100
frame5:
  objtype=node
  serial=99200G1
  mtm=9A00-100
frame6:
  objtype=node
  serial=99200D1
  mtm=9A00-100
```

## Supernode Numbers for CECs

The **supernode-list** file lists what supernode numbers should be given to each CEC in each frame. Here is a sample file:

```
frame1: 0, 1, 16
frame2: 17, 32
frame3: 33, 48, 49
frame4: 64 , 65, 80
frame5: 81, 96
frame6: 97(1), 112(1), 113(1), 37(1), 55, 71
```

The name before the colon is the node name of the frame. The numbers after the colon are the supernode numbers to assign to the groups of CECs in that frame from bottom to top. Each supernode contains 4 CECs, unless it is immediately followed by “(#)”, in which case the number in parenthesis indicates how many CECs are in this supernode.

## A Simpler Configuration File

This is an example of a simple cluster config file that just defines the frames and CECs for 2 frames, without specifying VPD data or supernode numbers at this time.

```
xcat-site:
  use-direct-fsp-control = 1

xcat-frames:
  hostname-range = frame[1-2]

xcat-cecs:
  #hostname-range = cec[01-24]
  hostname-range = f[1-2]c[01-12]
  num-cecs-per-frame = 12

xcat-lpars:
  hostname-range = f[1-2]c[01-12]p[01,05,09,13,17,21,25,29]
```

## Database Attributes Written

The following lists which database attributes are filled in as a result of each stanza. Note that depending on the values in the stanza, some attributes might not be filled in.

### xcat-site

site table: domain, nameservers, topology

### xcat-hmcs

site table: ea\_primary\_hmc, ea\_backup\_hmc

nodelist table: node, groups (all HMCs (hmc) ), hidden

hosts table: node, ip

ppc table: node, comments

nodetype table: node, nodetype

### xcat-frames

nodelist table: node, groups (all frames (frame) ), hidden

ppc table: node, id, hcp, nodetype, sfp

nodetype table: node, nodetype

nodehm table: node, mgt

vpd table: node, serial, mtm, side

### xcat-bpas

nodelist table: node, groups (bpa,all) , hidden

ppc table: node, id, hcp, nodetype, parent

nodetype table: node, nodetype

nodehm table: node, mgt

vpd table: node, serial, mtm, side

**xcat-cecs**

nodelist table: node, groups (all CECs (cec), all CECs in a frame (<frame>cec) ), hidden

ppc table: node, supernode, hcp, id, parent

nodetype table: node, nodetype

nodehm table: node, mgt

nodegroup table: groupname, grouptype, members, wherevals (all nodes in a CEC (<cec>nodes) )

nodepos: rack, u

**xcat-fsps**

nodelist table: node, groups (fsp,all), hidden

ppc table: node, id, hcp, nodetype, parent

nodetype table: node, nodetype

nodehm table: node, mgt

vpd table: node, serial, mtm, side

**xcat-building-blocks**

site table: sharedtftp, sshbetweennodes(service)

ppc table: node, parent (for frame)

**xcat-service-nodes**

nodelist table: node, groups (all service nodes (service), all service nodes in a BB (bb<num>service) )

hosts table: node, ip, hostnames, otherinterfaces

ppc table: node, id, hcp, parent

nodetype table: node, nodetype, arch

nodehm table: node, mgt, cons

noderes table: netboot

servicenode table: node, nameserver, dhcpserver, tftpserver, nfsserver, conserved, monserver, ftpserver, nimserv, ipforward

nodegroup table: groupname, grouptype, members, wherevals (all nodes under a service node (<servicenode>nodes) )

nodepos: rack, u

**xcat-storage-nodes**

nodelist table: node, groups (all storage nodes (storage), all storage nodes in a BB (bb<num>storage) )

hosts table: node, ip, hostnames, otherinterfaces

ppc table: node, id, hcp, parent

nodetype table: node, nodetype, arch

nodehm table: node, mgt, cons

noderes table: netboot, xcatmaster, servicenode

nodepos: rack, u

### xcat-compute-nodes

odelist table: node, groups (all compute nodes (compute) )

hosts table: node, ip, hostnames, otherinterfaces

ppc table: node, id, hcp, parent

nodetype table: node, nodetype, arch

nodehm table: node, mgt, cons

nodes table: netboot, xcatmaster, servicenode

nodepos: rack, u

### ll-config

postscripts: postscripts

## OPTIONS

### -s|--stanzas *stanza-list*

A comma-separated list of stanza names that **xcatsetup** should process in the configuration file. If not specified, it will process all the stanzas that start with 'xcat' and some other stanzas that give xCAT hints about how to set up the HPC products.

This option should only be specified if you have already run **xcatsetup** earlier with the stanzas that occur before this in the configuration file. Otherwise, objects will be created that refer back to other objects that do not exist in the database.

### -v|--version

Command Version.

### -?|-h|--help

Display usage message.

### --yesreallydeletenodes

Delete the nodes represented in the cluster config file, instead of creating them. This is useful if your first attempt with the cluster config file wasn't quite right and you want to start over. But use this option with extreme caution, because it will potentially delete a lot of nodes. If the only thing you have done so far in your database is add nodes by running **xcatsetup**, then it is safe to use this option to start over. If you have made other changes to your database, you should first back it up using `dumpxCATdb(1)|dumpxCATdb.1` before using this option.

## RETURN VALUE

0. The command completed successfully.
1. An error has occurred.

## EXAMPLES

1. Use the sample config.txt file at the beginning of this man page to create all the objects/nodes for a 2 building block cluster.

```
xcatsetup config.txt
```

The output:

```
Defining site attributes...
Defining HMCs...
Defining frames...
Defining CECs...
Defining building blocks...
Defining LPAR nodes...
```

2. Use the simpler config file shown earlier in this man page to create just the frame and cec objects:

```
xcatsetup config-simple.txt
```

The output:

```
Defining frames...
Defining CECs...
```

## FILES

/opt/xcat/sbin/xcatsetup

## SEE ALSO

mkdef(1)|mkdef.1, chdef(1)|chdef.1, lsdef(1)|lsdef.1, xcatstanzafile(5)|xcatstanzafile.5, noderange(3)|noderange.3, nodeadd(8)|nodeadd.8

## xcatsnap.8

## NAME

**xcatsnap** - Gathers information for service about the current running xCAT environment.

## SYNOPSIS

**xcatsnap**

**xcatsnap** [-h | --help]

**xcatsnap** [-v | --version]

**xcatsnap** [-B | --bypass]

**xcatsnap** [-d | --dir]

## DESCRIPTION

**xcatsnap** - The xcatsnap command gathers configuration, log and trace information about the xCAT components that are installed. This command only collects the data on the local node on which this command is run. This command is typically executed when a problem is encountered with any of these components in order to provide service information to the IBM Support Center.

This command should only be executed at the instruction of the IBM Support Center.

## OPTIONS

### **-h|--help**

Displays the usage message.

### **-v|--version**

Displays the release version of the code.

### **-B|--bypass**

Runs in bypass mode, use if the xcatd daemon is hung.

### **-d|--dir**

The directory to put the snap information. Default is /tmp/xcatsnap.

## ENVIRONMENT VARIABLES

## EXAMPLES

1. Run the xcatsnap routine in bypass mode and put info in /tmp/mydir :

```
xcatsnap -B -d /tmp/mydir
```

2. To run the xcatsnap routine and use default directory /tmp/xcatsnap :

```
xcatsnap
```

## xCAT Tools

*Disclaimer:* **Use at your own risk**

The following tools are shipped with xCAT and have been contributed by various xCAT community users. The tools are located under /opt/xcat/share/xcat/tools/.

## detect\_dhcpd

```
Usage: detect_dhcpd -i interface [-m macaddress] [-t timeout] [-V]
```

This command can be used to detect the dhcp server **in** a network **for** a specific mac\_↵  
↵address.

### Options:

-i interface: The interface which facing the target network.  
-m macaddress: The mac that will be used to detect dhcp server. Recommend to use the\_↵  
↵real mac of the node that will be netboot. If no specified, the mac of\_↵  
↵interface which specified by -i will be used.  
-t timeout: The time to wait to detect the dhcp messages. The default value **is**\_↵  
↵10s.

Author: Wang, Xiao Peng

## mac2linklocal

```
Usage: mac2linklocal -m
```

Determines the IPv6 link local address that **is** appropriate **for** a NIC, based on its MAC.

Author: Li, Guang Cheng

## mktoolscenter

```
Usage: mktoolscenter
```

```
--ph  
--pp  
--puser  
--ppw  
-l  
-s  
--nfserver  
--nfspath  
--profilename  
--help
```

Updates IBM system x server hardware using IBM Bootable Media Creator.

Author: Jim Turner



## nodesw

nodesw changes the vlan of a node to a specified vlan  
 requires xCAT 2.0, Switch configured with SNMP sets, and only tested on SMC8648T  
 nodesw -h|--help  
 nodesw [-v] vlan  
 nodesw [-v] show

Author: Vallard Benincosa

## reorgtbls

DB2 Table Reorganization utility.  
 This script can be set as a cron job or run on the command line to reorg the xcatdb DB2 database tables. It automatically added as a cron job, if you use the db2sqlsetup command to create your DB2 database setup for xCAT.  
 Usage:  
 --V - Verbose mode  
 --h - usage  
 --t -comma delimited list of tables.  
 Without this flag it reorgs all tables in the xcatdb database .

Author: Lissa Valletta

## rmblade

Usage: rmblade [-h|--help]  
 Response to SNMP for monsetting to remove blade from xCAT when trap is recieved.  
 Pipe the MM IP address and blade slot number into this cmd.  
 Example:  
 1. user removes a blade from the chassis  
 2. snmp trap setup to point here  
 3. this script removes the blade configuration from xCAT  
 4. so if blade is placed in new slot or back in then xCAT goes through rediscover process again.

Author: Jarrod Johnson

## rmnodecfg

Usage: rmnodecfg [-h|--help]  
 Removes the configuration of a node so that the next time you reboot it, it forces it to go through the discovery process.  
 This does not remove it completely from xCAT. You may want to do this command before running noderm to completely purge the system of the node

Author: Vallard Benincosa

## test\_hca\_state

test\_hca\_state (part of the BEF\_Scripts **for** xCAT) v3.2.27

Usage: test\_hca\_state NODERANGE [FILTER] | xcoll

--help Display this help output.

### NODERANGE

An xCAT noderange on which to operate.

### FILTER

A string to match **in** the output, filtering out everything **else**. This **is** passed to "egrep" **and** can be a simple string **or** a regular expression.

### Purpose:

This tool provides a quick **and** easily repeatable method of validating key InfiniBand adapter (HCA) **and** node based InfiniBand settings across an entire cluster.

Having consistent OFED settings, **and** even HCA firmware, can be very important **for** a properly functioning InfiniBand fabric. This tool can help you confirm that your nodes are using the settings you want, **and if any** nodes have settings discrepancies.

### Example output:

```
#
# This example shows that all of rack 14 has the same settings.
#
root@mgt1:~ # test_hca_state rack14 | xcoll
=====
rack14
=====
OFED Version: MLNX_OFED_LINUX-2.0-3.0.0.3 (OFED-2.0-3.0.0):
mlx4_0
  PCI: Gen3
  Firmware installed: 2.30.3200
  Firmware active:    2.30.3200
  log_num_mtt:        20
  log_mtt_per_seg:    3
  Port 1: InfiniBand   phys_state: 5: LinkUp
    state: 4: ACTIVE
    rate: 40 Gb/sec (4X FDR10)
    symbol_error: 0
    port_rcv_errors: 0
  Port 2: InfiniBand   phys_state: 3: Disabled
    state: 1: DOWN
    rate: 10 Gb/sec (4X)
    symbol_error: 0
```

(continues on next page)

(continued from previous page)

```

port_rcv_errors: 0

IPoIB
  recv_queue_size: 8192
  send_queue_size: 8192
  ib0:
    Mode: datagram
    MTU: 4092
    Mode: up
  ib1:
    Mode: datagram
    MTU: 4092
    Mode: up

#
# This example uses a FILTER on the word 'firmware'. In this case, we've
# upgraded the firmware across rack11 and rack12.
#
# - On rack11, we've also restarted the IB stack (/etc/init.d/openibd
#   restart) to activate the new firmware.
#
# - Rack 12 has also been updated, as we can see from the 'Firmware
#   installed' line, but it's nodes are still running with their prior
#   level of firmware and must reload the IB stack to have it take effect.
#
root@mgt1:~ # test_hca_state rack11,rack12 firmware | xcoll
=====
rack11
=====
  Firmware installed: 2.30.3200
  Firmware active:    2.30.3200

=====
rack12
=====
  Firmware installed: 2.30.3200
  Firmware active:    2.11.1260

```

Author: Brian Finley

If you encounter any problems with the tools, post a message to the xCAT mailing list for help.

## 1.5 Advanced Topics

### 1.5.1 Chain

The **chain** mechanism is created to allow the administrator to define a series of tasks or operations that will be executed in series on the target node.

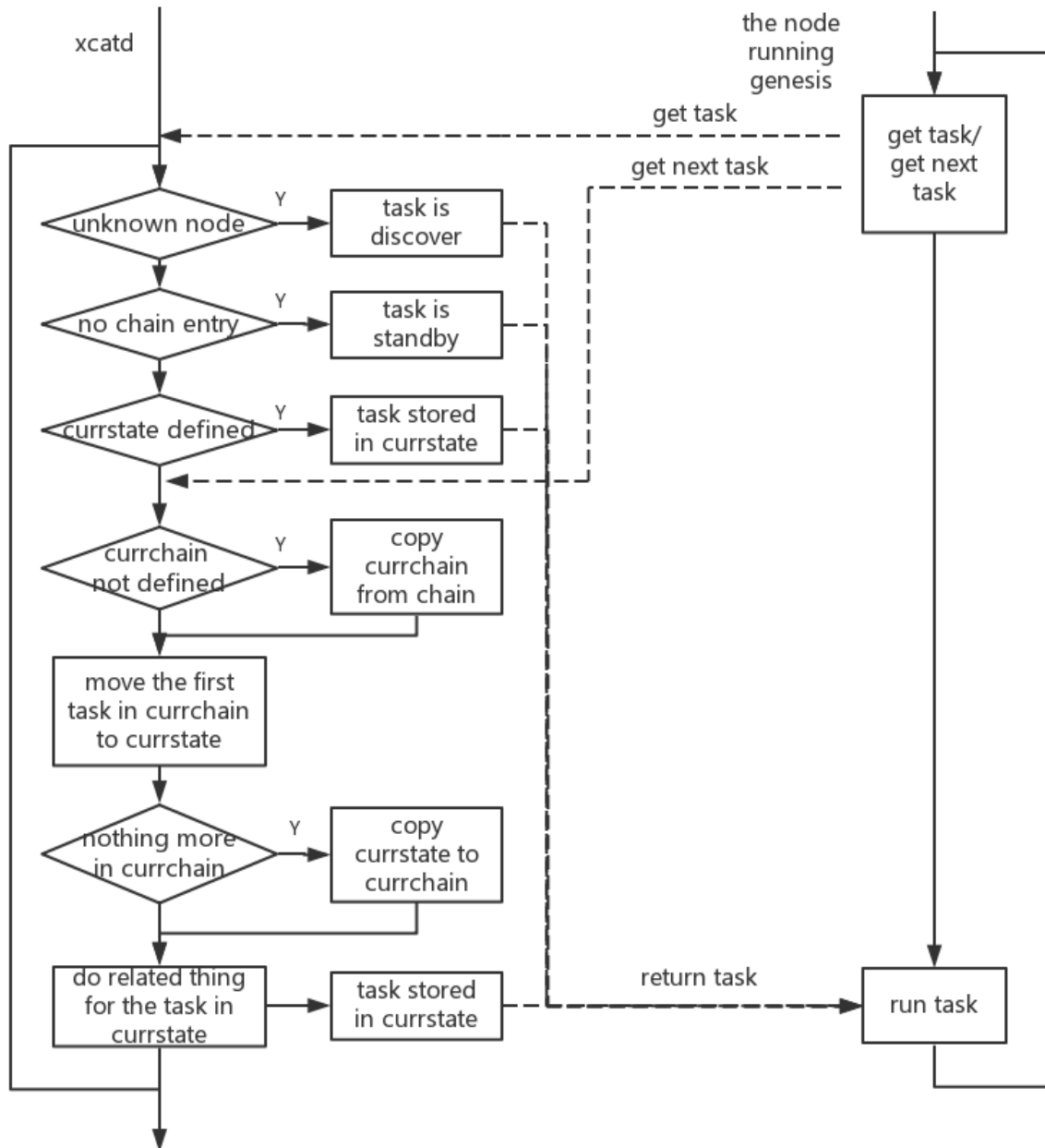
The **chain** mechanism is implemented in xCAT genesis system. The genesis is a customized Linux system, if can be used to do discovery and configuration after booted on the specified node.

#### Understanding chain table

The chain table (`tabdump chain`) is an xCAT database table that holds the chain configuration. The following attributes can be defined to perform the chain function:

```
* currstate
* currchain
* chain
```

To know how are those three attributes used, reference the picture:



## Task Type

xCAT supports following types of task which could be set in the chain:

- `runcmd`

```
runcmd=<cmd>
```

Currently only the `bmcsetup` command is officially supplied by xCAT to run to configure the bmc of the compute node. You can find the `bmcsetup` in `/opt/xcat/share/xcat/netboot/genesis/<arch>/fs/bin/`. You also could create your command in this directory and adding it to be run by `runcmd=<you cmd>`.

```
runcmd=bmcsetup
```

---

**Note:** The command `mknb <arch>` is needed before reboot the node.

---

- `runimage`

```
runimage=<URL>
```

**URL** is a string which can be run by `wget` to download the image from the URL. The example could be:

```
runimage=http://<IP of xCAT Management Node>/<dir>/image.tgz
```

**The `image.tgz` must have the following properties:**

- Created using the `tar zcvf` command
- The tarball must include a `runme.sh` script to initiate the execution of the `runimage`

To create your own image, reference creating image for `runimage`.

**Tip:** You could try to run `wget http://<IP of xCAT Management Node>/<dir>/image.tgz` manually to make sure the path has been set correctly.

- `osimage`

```
osimage=<image name>
```

This task is used to specify the image that should be deployed onto the compute node.

- `shell`

Causes the genesis kernel to create a shell for the administrator to log in and execute commands.

- `standby`

Causes the genesis kernel to go into standby and wait for tasks from the chain. ...

## Run Task List During Discovery

To run a list of tasks during the discovery, set the tasks in the chain table by using the `chdef` command to change the chain attribute, before powering on the nodes. For example:

```
chdef <noderange> chain='runcmd=bmcsetup,osimage=<osimage name>'
```

These tasks will be run after the discovery.

## Run Task List to Configure a Node

Run the `nodeset` command to set the tasks for the compute node and `rpower <noderange> reset` to initiate the running of tasks.

```
nodeset <noderange> runimage=http://<IP of xCAT Management Node>/image.tgz,osimage=
↪<image_name>
rpower <noderange> reset
```

In this example, the `runimage` will be run first, and then the image `<image_name>` will be deployed to the node.

## 1.5.2 Cluster Maintenance

### Compute Node

#### Changing the Hostname/IP address

#### Background

If the hostname or IP address has already been modified on compute nodes, follow the steps to change the configuration in `xcat`.

#### Remove Old Provision Environment

1. Remove the nodes from DNS configuration

```
makedns -d <noderange>
```

2. Remove the nodes from the DHCP configuration

```
makedhcp -d <noderange>
```

3. Remove the nodes from the goconserv server configuration

```
makegocons -d <noderange>
```

## Change Definition

1. Change networks table definitions

```
lsdef -t network -l
```

The output may be like

```
10_0_0_0-255_0_0_0 (network)
192_168_122_0-255_255_255_0 (network)
```

Change the networks table definitions, For example 192\_168\_122\_0-255\_255\_255\_0 is a original network configuration which should be modified to 192\_168\_123\_0-255\_255\_255\_0:

```
rmdef -t network 192_168_122_0-255_255_255_0
mkdef -t network 192_168_123_0-255_255_255_0 net=192.168.123.0 mask=255.255.255.0
```

2. Change the hostname in the xCAT database (This command only supports one node at a time). For many nodes you will have to write a script.

```
# changes node1 to node2 in the database
chdef -t node -o node1 -n node2
```

3. Change the hostname and IP address in the /etc/hosts file

- If you do not use the hosts table in xCAT to create the /etc/hosts file, edit the /etc/hosts file and change your hostname and IP address entries directly.
- If you use the xCAT hosts table, and your nodes are defined by name in the hosts table, the hosts table must be updated with the new names when we changed the node name using chdef command. If the hosts tables contains regular expression, you have to rewrite the regular expression to match your new hostname and IP address.
- If these is no regular expression in the hosts table, you can run

```
# change the IP address for the new hostname in the hosts table.
nodech <newnodename> hosts.ip="x.xx.xx.xx"
# add hostname/IP records in /etc/hosts from the definition in the xCAT hosts
# table for the <noderange>
makehosts <noderange>
```

## Update The Provision Environment

1. Configure the new names in DNS

```
makedns -n
```

2. Configure the new names in DHCP

```
makedhcp -a
```

3. Configure the new names in goconservser

```
makegocons
```



## Replacing Nodes

### OpenPOWER Nodes

When compute nodes are physically replaced in the frame, leverage xCAT to re-discover the compute nodes. The following guide can be used for:

- IBM OpenPOWER S822LC for HPC

1. Identify the machine(s) to be replaced: `frame10cn02`.
2. **[Optional]** It's recommended to set the BMC IP address back to DHCP, if it was set to STATIC.

```
rspconfig frame10cn02 ip=dhcp
```

3. Set the outgoing machine to offline and remove attributes of the machine:

```
nodeset frame10cn02 offline
chdef frame10cn02 mac=""
```

4. If using **MTMS**-based discovery, fill in the Model-Type and Serial Number for the machine:

```
chdef frame10cn02 mtm=8335-GTB serial=<NEW SERIAL NUMBER>
```

5. If using **SWITCH**-based discovery, go on to the next step. The switch and switch-port should already be set in the compute node definition.

Node attributes will be replaced during the discovery process (mtm, serial, mac, etc.)

6. Search for the new BMC in the open range:

```
bmcdiscover --range <IP open range> -w -z
```

7. When the BMC is found, start the discovery with the following commands:

```
rsetboot /node-8335.* net
rpower /node-8335.* boot
```

## Management Node

### Changing the hostname/IP address

#### Overview

This document is intended to describe the steps that must be taken if you need to change your Linux Management Node's hostname and/or IP address after the cluster is installed and configured by xCAT. This documentation will only cover the changes by xCAT and will not try to cover any other changes by any other tools.

## Backup your xCAT data

Clean up the database by running `tabprune` command:

```
tabprune -a auditlog
tabprune -a eventlog
```

Now take a snapshot of the Management Node. This will also create a database backup. You can use this data as reference if needed.

```
xcatsnap -d
```

## Stop xCAT

You need to stop the `xcat` daemon and any other applications that are using the xCAT database on the Management Node and the Service Nodes. To determine your database, run

```
lsxcatd -a | grep dbengine
```

To stop xCAT:

```
service xcatd stop
```

## Stop The Database

For all databases except SQLite, you should stop the database. For example

```
service postgresql stop
service mysqld stop
```

## Change the Management Hostname

- `hostname` command

```
hostname <new_MN_name>
```

- Update the hostname configuration files:

Add hostname in `/etc/hostname`

Add HOSTNAME attribute in `/etc/sysconfig/network` (only for [RHEL])

## Update Database Files

You need to update the new MN hostname or IP address in several database configuration files.

### SQLite

Nothing to do.

### PostgreSQL

- Edit `/etc/xcat/cfgloc` file...  
Replace `Pg:dbname=xcatdb;host=<old_MN_ip>|xcatadm|xcat20` with `Pg:dbname=xcatdb;host=<new_MN_ip>|xcatadm|xcat20`.
- Edit config database config file `/var/lib/pgsql/data/pg_hba.conf`...  
Replace `host all all <old_MN_ip>/32 md5` with `host all all <new_MN_ip>/32 md5`

### MySQL

- Edit `/etc/xcat/cfglooc`...  
Replace `mysql:dbname=xcatdb;host=<old_MN_ip>|xcatadmin|xcat20` with `mysql:dbname=xcatdb;host=<new_MN_ip>|xcatadmin|xcat20`

## Start the database

```
service postgresql start
service mysqld start
```

Start xCAT

```
service xcatd start
```

Verify your new database setup

```
lsxcatd -a | grep dbengine
tabdump site # if output exists
```

## Change The Definition In xCAT Database

### Change the site table master attribute

```
chdef -t site master=<new_MN_ip>
```

## Change all IP address attribute relevant to the MN IP address

For example, the old IP address was “10.6.0.1”

- Query all the attributes with old address

```
lsdef -t node -l | grep "10.6.0.1"
...
conserver=10.6.0.1
conserver=10.6.0.1
conserver=10.6.0.1
conserver=10.6.0.1
nfsserver=10.6.0.1
servicenode=10.6.0.1
xcatmaster=10.6.0.1
kcmdline=quiet repo=http://10.6.0.1/install/rhels6/ppc64/ ks=http://10.6.0.1/
↪install/autoinst
/slessn ksdevice=d6:92:39:bf:71:05
nfsserver=10.6.0.1
servicenode=10.6.0.1
tftpserver=10.6.0.1
xcatmaster=10.6.0.1
servicenode=10.6.0.1
xcatmaster=10.6.0.1
```

- Looking at the list above, taking `conserver` as an example, query the nodes with `conserver=10.6.0.1`:

```
lsdef -t node -w consver="10.6.0.1"
...
cn1 (node)
cn2 (node)
cn3 (node)
cn4 (node)
```

- Change the consver address for nodes `cn1`, `cn2`, `cn3`, `cn4`

```
chdef -t node cn1-cn4 consver=<new_ip_address>
```

- Repeat the same process for the other attributes containing the old IP address.

## Change networks table

Check your networks table to see if the network definitions are still correct, if not edit accordingly

```
lsdef -t network -l
chdef -t network <key=value>
```

## Check Result

You can check whether all the old address has been changed using

```
dumpxCATdb -P <new database backup path>
cd <new database backup path>
fgrep "10.6.0.1" *.csv
```

If the old address still exists in the \*.csv file, you can edit this file, then use the following command to restore the records

```
tabrestore <xxx.csv>
```

## Generate SSL credentials(optional)

Use the following command to generate new SSL credentials: `xcatconfig -c`.

Then update the following in xCAT:

- Update the policy table with new management node name and replace:

```
"1.4","old_MN_name",,,,,,"trusted",,
```

with:

```
"1.4","new_MN_name",,,,,,"trusted",``
```

- Setup up goconserver with new credentials

```
makegocons
```

## External DNS Server Changed

- Update nameserver entries in `/etc/resolv.conf`
- Update nameserver attribute in site table

```
chdef -t site -o clustersite nameservers="new_ip_address1,new_ip_address2"
```

- Update site forwarders in DB

```
chdef -t site -o clustersite forwarders="new_ip_address1,new_ip_address2"
```

- Run command `makedns -n`

## Domain Name Changed

Change the entries in `/etc/hosts`.

Change the `/etc/resolv.conf`, forwarders attribute in site table.

```
lsdef -t site -o clustersite -i forwarders
chdef -t site -o clustersite forwarders <new list>
```

Change the domain name in the xCAT database site table.

```
chdef -t site -o clustersite domain=<new_domainname>
```

From xCAT 2.8, multiple domains is supported in the cluster. Update the networks table definition.

```
lsdef -t network -l
chdef -t network -o <network_name> ddnsdomain=<new_domainname1,new_domainname2>
```

## Update the Provision Environment

Determine if the Management node is defined in the database, assuming it was done correctly using `xcatconfig -m`, by running:

```
lsdef __mgmtnode
```

If it exists, then use the return name and do the following:

- Remove the MN from DNS configuration

```
makedns -d <old_MN_name>
```

- Remove the MN from the DHCP configuration

```
makedns -d <old_MN_name>
```

- Remove the MN from the conserver configuration

```
makedns -d <old_MN_name>
```

- Change the MN name in the xCAT database

```
chdef -t node -o <old_MN_name> -n <new_MN_name>
```

- Add the new MN to DNS

```
makedns -n
```

- Add the MN to dhcp

```
makedhcp -a
```

- Add the MN to goconserver

```
makegocons
```

## Update the genesis packages

Run `mknb <arch>` after changing the ip of MN.

## Service Node

### Changing the Hostname/IP address

#### Change compute node definition relevant to the service node

Change the settings in database. Below shows a method to find out where the old IP address settings (take 10.6.0.1 as a example) are used in Hierarchy environment.

- Query the old attribute

```
lsdef -t node -l | grep "10.6.0.1"
# below is output of the above command. We can find out that nfsserver
# and servicenode are using the old IP address setting.
nfsserver=10.6.0.1
servicenode=10.6.0.1
```

- Query the nodes whose nfsserver is 10.6.0.1

```
lsdef -w nfsserver==10.6.0.1
# below is output of the above command
cn1 (node)
cn2 (node)
cn3 (node)
cn4 (node)
```

- Change the nfsserver address for cn1,cn2,cn3,cn4 by running the following command:

```
chdef -t node cn1-cn4 nfsserver=<new service node IP address>
```

## Database Connection Changes

Granting or revoking access privilege in the database for the service node.

- For MySQL, refer to *Granting/Revoking access to the database for Service Node Clients*.

## Update Provision Environment on Service Node

If you are using service nodes to install the nodes and using `/etc/hosts` for hostname resolution, you need to copy the new `/etc/hosts` from the management node to the service nodes, then run `makedns -n` on the service nodes. For example:

```
xdcp <servicenodes> /etc/hosts /etc/hosts
xdsh <servicenodes> makedns -n
```

Reinstall the nodes to pick up all changes

```
nodeset <noderange> osimage=<osimagename>
```

Then use your normal command to install the nodes like `rinstall`, `rnetboot`, etc.

## Software and Firmware Inventory

xCAT provides a command `sinv` that checks the software and firmware inventory in this cluster.

The command creates an inventory of the input software/firmware check, comparing to other machines in the cluster and produces an output of node that are installed the same and those that are not.

This command uses the `xdsh` parallel command, so it is in itself a parallel command, and thus can be run on multiple cluster nodes at one time and is hierarchical.

The `sinv` command is designed to check the configuration of the nodes in a cluster. The command takes as input command line flags, and one or more templates which will be compared against the output of the `xdsh` command, designated to be run on the nodes in the `noderange`.

The nodes will then be grouped according to the template they match and a report returned to the administrator in the output file designated or to `stdout`.

`sinv` supports checking the output from the `rinv` or `xdsh` command.

For example, if you wanted to check the `ssh` level on all the nodes and make sure they were the same as on the service node, you would first generate a template from the “good” service node (`sn1`) by running the following:

```
xdsh sn1 "rpm -qa | grep ssh" | xdshcoll > /tmp/sinv/sinv.template
```

To execute `sinv` using the `sinv.template` generated above on the `nodegroup`, `testnodes`, writing output report to `/tmp/sinv.output`, enter:

```
sinv -c "xdsh testnodes rpm -qa | grep ssh" -p /tmp/sinv/sinv.template -o /tmp/sinv.  
↪output
```

The report will look something like this, if every node matches:

Command started with following input:

```
xdsh cmd:xdsh testnodes rpm -qa | grep ssh.  
Template path:/tmp/sinv/sinv.template.  
Template cnt:0.  
Remove template:NO.  
Output file:/tmp/sinv/sinv.output.  
Exactmatch:NO.  
Ignorefirst:NO.  
Seed node:None.  
file:None.  
The following nodes match /tmp/lissav/sinv.template:  
testnodes
```

There are many options for matching and reporting supported by the `sinv` command, including support to run `rinv` and generate reports on firmware inventory.



### 1.5.3 Migrate xCat Management node

#### xCAT Management Node Migration

This document describes how to migrate xCAT Management node to a new node. The following example describes a typical scenario, this example is verified on redhat7.3.

1. Initially, the first xcat management node is active, and the second node is passive.
2. Backup all useful xCAT data from xCAT Management node to back-up server at regular intervals.
3. When the first xCAT management node is broken, use backup to restore original xCAT data to the second node with the same host name and ip.

#### Backup Old xCAT Management Node

Backup xCAT management node data to backup server:

##### 1.1 Backup xCAT important files and directories:

1. Get installdir from site table, backup installdir directory, in this case, back up install directory:

```
lsdef -t site clustersite -i installdir
  Object name: clustersite
  installdir=/install
```

2. Backup these two xCAT directories:

```
~/ .xcat
/etc/xcat
```

---

**Note:** Backing up ~/ .xcat is for all users who have xCAT client certs.

---

3. If there are customized files and directories for otherpkgdir, pkgdir, pkglist or template in some *osimage* definitions, backup these files and directories. for example:

```
lsdef -t osimage customized_rhels7.4-x86_64-install-compute -i otherpkgdir,pkgdir,
→pkglist,template
  Object name: customized_rhels7.4-x86_64-install-compute
  otherpkgdir=/<customized_dir>/post/otherpkgs/rhels7.4/x86_64
  pkgdir=/<customized_pkgdir>/rhels7.4/x86_64
  pkglist=/<customized_pkglist_dir>/compute.rhels7.pkglist
  template=/<customized_temp_dir>/compute.rhels7.tmpl
```

##### 1.2 Backup ssh related files:

```
/etc/ssh
~/ .ssh
```

##### 1.3 Backup host files:

```
/etc/resolv.conf
/etc/hosts
```

(continues on next page)

(continued from previous page)

```
/etc/passwd
/etc/group
```

1.4 Backup yum resource files:

```
/etc/yum.repos.d
```

1.5 Backup conserver conf files:

```
/etc/conserver.cf
```

1.6 Backup DNS related files:

```
/etc/named
/etc/named.conf
/etc/named.iscdlv.key
/etc/named.root.key
/etc/rndc.key
/etc/sysconfig/named
/var/named
```

1.7 Backup dhcp files:

```
/etc/dhcp
/var/lib/dhcpd
/etc/sysconfig/dhcpd
/etc/sysconfig/dhcpd6
```

1.8 Backup apache:

```
/etc/httpd
/var/www
```

1.9 Backup tftp files:

```
/tftpboot
```

1.10 Backup NTP configure file:

```
/etc/ntp.conf
```

1.11 Backup database configure files (optional):

- **[PostgreSQL]**

```
/var/lib/pgsql/data/pg_hba.conf
/var/lib/pgsql/data/postgresql.conf
```

1.12 Backup NFS (optional):

```
/etc/exports
/var/lib/nfs
/etc/sysconfig/nfs
```

### 1.13 (optional)

Besides the files mentioned above, there may be some additional customization files and production files that need to be backup, depending on your local unique requirements. Here are some example files that can be considered:

```
/.profile
/.rhosts
/etc/auto_master
/etc/auto/maps/auto.u
/etc/motd
/etc/security/limits
/etc/netscvc.conf
/etc/inetd.conf
/etc/security/passwd
/etc/security/group
/etc/services
/etc/inittab(andmore)
```

1.14 Backup the xCAT database tables for the current configuration, using command:

```
dumpxCATdb -p <your_backup_dir>
```

1.15 Save all installed xCAT RPM names into a file:

```
rpm -qa|grep -i xCAT > xcat_rpm_names
```

1.16 (Optional) Find customization made to files installed from packages, backup these files. For example

```
rpm -q --verify -a conserver-xcat
rpm -q --verify -a xCAT-server
rpm -q --verify -a syslinux-xcat
rpm -q --verify -a xCAT-client
rpm -q --verify -a xCAT
```

## Restore xCAT management node

2.1 Power off old xCAT management server before configuring new xCAT management server

2.2 Configure new xCAT management server using the same ip and hostname as old xCAT management server. Configure the same additional network for hardware management network if needed, for example, bmc network or hmc network. xCAT management server setup refer to [Prepare the Management Node](#)

2.3 Overwrite files/directories mentioned in above 1.2, 1.3, 1.4 from backup server to new xCAT management server

2.4 Download xcat-core and xcat-dep tar ball, then install xCAT in new xCAT management server, refer to [install xCAT](#)

2.5 Use `rpm -qa|grep -i xCAT` to list all xCAT RPMs in new xCAT management node, compare these RPMs base name with those in `xcat_rpm_names` from above 1.15. If some RPMs are missing, use `yum install <rpm_package_basename>` to install missing RPMs.

2.6 If use MySQL/MariaDB/PostgreSQL, migrate xCAT to use MySQL/MariaDB/PostgreSQL refer to [Configure a Database](#)

2.7 To restore the xCAT database

- a. Restore xCAT database from the `/dbbackup/db` directory without `auditlog` and `eventlog`, enter:

```
restorexCATdb -p /dbbackup/db
```

- b. Restore the xCAT database including auditlog and eventlog from the /dbbackup/db directory, enter:

```
restorexCATdb -a -p /dbbackup/db
```

- c. (optional) Overwrite files in above 1.11, restart PostgreSQL:

```
service postgresql restart
```

2.8 Overwrite remaining files/directories mentioned in above 1.1, 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 1.12; If needed, check if files exist based on above 1.13 and 1.16.

2.9 Verify xCAT:

```
tabdump site
```

2.10 Restart named, use nslookup to check DNS:

```
service named restart  
nslookup <cn1>
```

2.11 Restart conserver, use rcons to check console:

```
service conserver restart  
rcons <cn1>
```

2.12 Configure DHCP:

```
makedhcp -n  
makedhcp -a
```

2.13 Restart httpd for REST API, for more information refer to [Rest API](#):

```
service httpd restart
```

## 1.5.4 Confluent

Confluent is a new codebase with a few goals in mind:

- Augment xCAT 2.X series
- Potentially serve in place of xCAT-server for the next generation of xCAT

**Disclaimer:** *Confluent code in conjunction with xCAT 2.X is currently BETA, use at your own risk*

## confluent-server

### Getting Started

Confluent is intended to be used in conjunction with xCAT. The following documentation assumes that xCAT is already installed and configured on the management node.

### Download confluent

To build from source, ensure your machine has the correct development packages to build rpms, then execute the following:

- Clone the git repo:

```
git clone https://github.com/xcat2/confluent.git
```

- Build the confluent-server and confluent-client packages:

```
cd confluent/confluent_server ; ./buildrpm ; cd -  
cd confluent/confluent_client ; ./buildrpm ; cd -
```

## Install

### dependency

The following example describes the steps for **rhels7.5** on **ppc64le**:

```
yum install libffi-devel.ppc64le  
yum install openssl-devel  
pip install crypto pyasn1 pycrypto eventlet pyparsing netifaces scrapy pysnmp paramiko_  
↪ pyghmi pyte
```

## confluent

Installing xCAT-confluent via rpm:

```
rpm -ivh /root/rpmbuild/RPMS/noarch/confluent_server-*.noarch.rpm --nodeps  
rpm -ivh /root/rpmbuild/RPMS/noarch/confluent_client-*.noarch.rpm --nodeps
```

You may find it helpful to add the confluent paths into your system path:

```
CONFLUENTROOT=/opt/confluent  
export PATH=$CONFLUENTROOT/bin:$PATH  
export MANPATH=$CONFLUENTROOT/share/man:$MANPATH
```

## Configuration

### Starting/Stopping confluent

To start confluent:

```
service confluent start
```

To stop confluent:

```
service confluent stop
```

If you want confluent daemon to start automatically at bootup, add confluent service to chkconfig:

```
chkconfig confluent on
```

### Replacing conserver with confluent

A new keyword, `consoleservice`, has been added to the xCAT site table to allow the system administrator to control between **conserver** and **confluent**. If `consoleservice` is not set, default behavior is to use **conserver**.

Set the `consoleservice` to confluent:

```
chdef -t site consoleservice='confluent'
```

Run `makeconfluentcfg` to create the confluent configuration files:

```
makeconfluentcfg
```

Use `rcons` as before to start the console session.:

```
rcons <singlenode>
```

### Web Browser access

Confluent-api and confluent-consoles are able to be accessed from the browser. It is **highly** recommended that you create a non-root user to access the sessions:

Create the non-root user on the management node

```
# useradd -m xcat
```

Create a non-root user **in** confetty

```
# /opt/confluent/bin/confetty create users/xcat
```

Set the password **for** the non-root user

```
# /opt/confluent/bin/confetty set users/xcat password="mynewpassword"
password="*****"
```

## Rest Explorer

Configure the httpd configuration for confluent-api by creating a `confluent.conf` file under `/etc/httpd/conf.d/` directory:

```
The example uses server ip: 10.2.5.3 and port 4005

# cat /etc/httpd/conf.d/confluent.conf
LoadModule proxy_http_module modules/mod_proxy_http.so
<Location /confluent-api>
    ProxyPass http://10.2.5.3:4005
</Location>

# restart httpd
service httpd restart
```

Now point your browser to: `http://<server ip>:<port>` and log in with the non-root user and password created above.

## Confluent consoles

confluent-web is provided in a subdirectory under the confluent project `confluent_web`

Download the content of that directory to `/var/www/html/confluent` and point your browser to:

```
http://<server ip>/confluent/consoles.html
```

## confluent-client

### Starting the confetty client

As the root user, running `/opt/confluent/bin/confetty` will open the confetty prompt

```
[root@cf910f02c05p03 ~]# /opt/confluent/bin/confetty
/ ->
```

## Creating a non root user

It's recommended to create a non root user to use to connect to confetty

1. Create a non-root user on the management node:

```
useradd -m xcat
```

2. As root, create a non-root user in confetty:

```
/opt/confluent/bin/confetty create users/xcat
```

3. Set the password for the non-root user:

```
/opt/confluent/bin/confetty set users/xcat password="mynewpassword"
password="*****"
```

## Connecting to a remote server

In order to do remote sessions, keys must first be added to `/etc/confluent`

- `/etc/confluent/privkey.pem` - private key
- `/etc/confluent/srvcert.pem` - server cert

If you want to use the xCAT Keys, you can simple copy them into `/etc/confluent`

```
cp /etc/xcat/cert/server-key.pem /etc/confluent/privkey.pem
cp /etc/xcat/cert/server-cert.pem /etc/confluent/srvcert.pem
```

The user and password may alternatively be provided via environment variables:

```
CONFLUENT_USER=xcat
CONFLUENT_PASSPHRASE="mynewpassword"
export CONFLUENT_USER CONFLUENT_PASSPHRASE
```

Start confetty, specify the server IP address:

```
confetty -s <remote_ip>
```

If you want to run a confluent command against another host, could set the `CONFLUENT_HOST` variable:

```
CONFLUENT_HOST=<remote_ip>
export CONFLUENT_HOST
```

## 1.5.5 Go Conserver

`goconserver` is a conserver replacement written in [Go](https://github.com/xcat2/goconserver/) programming language. For more information, see <https://github.com/xcat2/goconserver/>

### Quickstart

1. For refresh xCAT installation, run the command below to start and configure `goconserver`

```
makegocons
```

The new console logs will start logging to `/var/log/consoles/<node>.log`

1. For xCAT updating, and use `conserver` before, following the step below to enable `goconserver`

1. stop `conserver` on management node

```
systemctl stop conserver.service
```

2. For hierarchical cluster, shall also stop `conserver` on **service nodes**, and config `goconserver` as console server:

```
xdsh service 'systemctl stop conserver.service'
```

```
chdef -t group -o service setupconserver=2
```

3. start and configure `goconserver`

```
makegocons
```

The new console logs will start logging to `/var/log/consoles/<node>.log`



2. To check the console status of nodes, use:

```
makegocons -q
```

## Configuration

### Location

The configuration file for goconserver is located at `/etc/goconserver/server.conf`. When the configuration is changed, reload using: `systemctl restart goconserver.service`. An example for the configuration could be found from [Example Conf](#).

### Tag For xCAT

xCAT generates a configuration file that includes a identifier on the first line. For example:

```
#generated by xcat Version 2.13.10 (git commit 7fcd37ffb7cec37c021ab47d4baec151af547ac0,
↳built Thu Jan 25 07:15:36 EST 2018)
```

makegocons checks for this token and will not make changes to the configuration file if it exists. This gives the user the ability to customize the configuration based on their specific site configuration.

## Multiple Output Plugins

goconserver support console redirection to multiple targets with `file`, `tcp` and `udp` logger plugins. The entry could be found like below:

```
console:
# the console session port for client(congo) to connect.
port: 12430

logger:
# for file logger
file:
# multiple file loggers could be specified
# valid fields: name, logdir
- name: default
  logdir: /var/log/goconserver/nodes/
- name: xCAT
  logdir: /var/log/containers

tcp:
- name: logstash
  host: briggs01
  port: 9653
  ssl_key_file: /etc/xcat/cert/server-cred.pem
  ssl_cert_file: /etc/xcat/cert/server-cred.pem
  ssl_ca_cert_file: /etc/xcat/cert/ca.pem

- name: rsyslog
  host: sn02
```

(continues on next page)

(continued from previous page)

```
port: 9653

udp:
- name: filebeat
  host: 192.168.1.5
  port: 512
```

With the configuration above, the console log files for each node would be written in both `/var/log/goconserver/nodes/<node>.log` and `/var/log/containers/<node>.log`. In addition, console log content will be redirected into remote services specified in the `tcp` and `udp` sections.

## Verification

To check if `goconserver` works correctly, see the log file `/var/log/goconserver/server.log`.

1. Check if TCP logger has been activated.

When starting `goconserver`, if the log message is like below, it means the TCP configuration has been activated.

```
{ "file": "github.com/xcat2/goconserver/console/logger/tcp.go (122)", "level": "info",
  → "msg": "Starting TCP publisher: logstash", "time": "2018-03-02T21:15:35-05:00" }
{ "file": "github.com/xcat2/goconserver/console/logger/tcp.go (122)", "level": "info",
  → "msg": "Starting TCP publisher: sn02", "time": "2018-03-02T21:15:35-05:00" }
```

2. Debug when encounter error about TCP logger

If the remote service is not started or the network is unreachable, the log message would be like below.

```
{ "file": "github.com/xcat2/goconserver/console/logger/tcp.go (127)", "level": "error",
  → "msg": "TCP publisher logstash: dial tcp 10.6.27.1:9653: getsockopt: connection_
  → refused", "time": "2018-03-07T21:12:58-05:00" }
```

Check the service status and the network configuration including the `selinux` and `iptables` rules. When the remote service works correctly, TCP or UDP logger of `goconserver` would recover automatically.

## Reconnect Interval

If console node is defined with `ondemand=false`, when the console connection could not be established, `goconserver` would reconnect automatically. The interval time could be specified at

```
console:
# retry interval in second if console could not be connected.
reconnect_interval: 10
```

## Performance Tuning

Adjust the worker numbers to leverage multi-core processor performance based on the site configuration.

```
global:
# the max cpu cores for workload
worker: 4
```

## Debug

The log level for goconserver is defined in `/etc/goconserver/server.conf`

```
global:
# debug, info, warn, error, fatal, panic
log_level: info
```

## REST API

goconserver provides REST API interface to manage the node sessions. For detail, see [REST](#).

### 1.5.6 Docker

#### Quick Start to Use xCAT Docker Image

A new Docker image will be published for each new release of xCAT. Use `docker search xcat2` to list all Docker images xCAT has released. xCAT Docker image official organization is `xcat`, repository is `xcat2`.

```
[dockerhost]# sudo docker search xcat2
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
xcat/xcat2	...	...	...	...

The xCAT Docker images are tagged to match the xCAT releases, If you want to deploy the xCAT 2.14.6 version, pull down the `xcat/xcat2:2.14.6` image. xCAT Docker image also has a `latest` tag to point to the latest release. Currently xCAT Docker images are based on CentOS.

**Attention:** To do discovery for POWER9 bare metal server, please refer to *xCAT Genesis Base*

#### Prerequisite for Docker Host

- To run xCAT under Docker, the services SELinux and AppArmor on Docker host must be disabled.

SELinux can be disabled with:

```
echo 0 > /selinux/enforce
sed -i 's/^SELINUX=.*$/SELINUX=disabled/' /etc/selinux/config
```

AppArmor can be disabled with:

```
/etc/init.d/apparmor teardown
```

- To run xCAT under Docker the ports described in *document* should be available.

For Linux user, use the following command to verify ports are not used

```
netstat -nlp |grep -E ":(3001|3002|68|53|873|80|69|12429|12430|67) "
```

## Pull the xCAT Docker Image from DockerHub

To pull the latest xCAT Docker image, run

```
[dockerhost]# sudo docker pull xcat/xcat2:latest
```

## Run xCAT in Docker Container

Run the xCAT Docker container with the Docker image xCAT/xCAT2:latest

```
[dockerhost]# sudo docker run -d \  
  --name xcatmn \  
  --network=host \  
  --hostname xcatmn \  
  --privileged \  
  -v /sys/fs/cgroup:/sys/fs/cgroup:ro \  
  -v /xcatdata:/xcatdata \  
  -v /var/log/xcat:/var/log/xcat \  
  -v /customer_data:/customer_data \  
  xcat/xcat2:latest
```

The descriptions:

### **name**

Assign a name to the container, this name can be used to manipulate the container on docker host.

### **--network=host**

Use the host network driver for a container, that container network stack is not isolated from the docker host.

### **hostname**

Specify the hostname of container, which is available inside the container.

### **--privileged=true**

Give extended privileges to this container.

### **-v /sys/fs/cgroup:/sys/fs/cgroup:ro**

Is **mandatory** configuration to enable systemd in container.

### **-v /xcatdata:/xcatdata**

xCAT container will create /xcatdata volume to store configuration and OS distro data. I.e. xCAT important directories /install, /tftpboot and /etc will be saved under /xcatdata. If user does not explicitly mount this directory to docker host, this directory will be mounted under /var/lib/docker/volumes/.

**-v /var/log/xcat:/var/log/xcat**

All xCAT running logs are saved under /var/log/xcat. Use this setting to export them to Docker host.

**-v /customer\_data:/customer\_data**

**Is optional.** Use this setting to transfer user data between Docker host and container.

## Run xCAT Command in Docker Container

To enter xCAT Docker container

```
[dockerhost]# sudo docker exec -it xcatmn bash
[xcatmn]#
```

Also can enter xCAT Docker container through ssh

```
[anynode]# ssh <docker_container_ip> -p 2200
```

**Attention:** Need to set site table depending on your own environment.

For example

```
[xcatmn]# chtag key=master site.value=<docker_host_ip>
```

Now container xcatmn will work as a normal xCAT management node, can run xCAT commands directly. For example

```
[xcatmn]# lsxcatd -a
```

**Attention:** Use of NFS outside of xCAT Docker container is recommended. For NFS service set up inside of xCAT Docker container, mount the shared directory with -v option when starting xCAT container.

## 1.5.7 Domain Name Resolution

### Cluster Name Resolution

Setting up name resolution and having the nodes be resolved to IP addresses are required in xCAT clusters.

There are many different ways to configure name resolution in your cluster. Four of the common choices will be described in this section (look for Option #: headings):

1. In a basic (non-hierarchical) cluster, point **all** nodes to a DNS server running on the **management** node. This **is** the most common setup.
2. In a basic (non-hierarchical) cluster, point **all** nodes to an external DNS running at **your** site. This requires that **all** of your nodes have network connectivity to your site.
3. In a hierarchical cluster, point **all** compute nodes to their service node. Make service **node** **as** the compute nodes' DNS server.
4. Don't use DNS, just distribute the /etc/hosts file to every node. If you choose this, you will have to distribute new versions of the /etc/hosts file to all the cluster nodes whenever you add new nodes to the cluster, and you will have to specify site, master and all other server attributes in the database as IP addresses.

But before any of those options are chosen, there are some things that must be done for all of the options.

### Set the site domain Attribute

First set the domain attribute in the xCAT site table to the hostname domain you want to use for your cluster nodes:

```
chdef -t site domain=mycluster.com
```

### Create xCAT Network Definitions

When you installed xCAT, it ran `makenetworks` to create network definitions for the networks that the management node is connected to (i.e. has NICs configured for). If the cluster-facing NICs were not configured when xCAT was installed, or if there are more networks in the cluster that are only available via the service nodes or compute nodes, create the new network definitions now.

Use the `mkdef` command to add additional networks to the xCAT database. (See the network for information about each attribute.) For example:

```
mkdef -t network clusternet net=11.0.0.0 mask=255.255.0.0 gateway=11.0.0.254 domain=app.  
↪cluster.com
```

---

**Note:** The `makedns` command (mentioned below) will only add nodes into the DNS configuration if the network for the node is defined.

---

If you want to use a different hostname domain or a different set of nameservers for nodes that are on a particular network, set those attributes in the corresponding network object:

```
mkdef -t network clusternet domain=app.cluster.com nameservers=1.2.3.4
```

### Populate the `/etc/hosts` File

All of the management node interfaces and all of the nodes need to be added to the `/etc/hosts` file on the xCAT management node (whether you are using the DNS option or not). You can either edit the `/etc/hosts` file by hand, or use `makehosts`.

If you edit the file by hand, it should look similar to:

```
127.0.0.1 localhost localhost.localdomain  
50.1.2.3 mgmtnode-public mgmtnode-public.cluster.com  
10.0.0.100 mgmtnode mgmtnode.cluster.com  
10.0.0.1 node1 node1.cluster.com  
10.0.0.2 node2 node2.cluster.com
```

Verify that your `/etc/hosts` file contains entries for all of your management node interfaces. Manually add any that are missing.

If your node names and IP addresses follow a regular pattern, you can easily populate `/etc/hosts` by putting a regular expression in the xCAT hosts table and then running `makehosts`. To do this, you need to first create an initial definition of the nodes in the database, if you haven't done that already:

```
mkdef node[01-80] groups=compute,all
```

Next, put a regular expression in the hosts table. The following example will associate IP address 10.0.0.1 with node1, 10.0.0.2 with node2, etc:

```
chdef -t group compute ip='|node(\d+)|10.0.0.($1+0)|'
```

(For an explanation of the regular expressions, see the `man xcatdb`.) Then run:

```
makehosts compute
```

and the following entries will be added to `/etc/hosts`:

```
10.0.0.1 node01 node01.cluster.com
10.0.0.2 node02 node02.cluster.com
10.0.0.3 node03 node03.cluster.com
```

This information is used by the `makehosts` command to add the additional interface hostnames etc. to the `/etc/hosts` file. It is also used by xCAT adapter configuration postscripts to automatically configure the additional network interfaces on the node. See the section (refer to *Specifying additional network interfaces for cluster nodes*).

**Note:** It is a convention of xCAT that for Linux systems the short hostname is the primary hostname for the node, and the long hostname is an alias. To have the long hostname be the primary hostname, you can use the `-l` option on the `makehosts` command.

## Preparing for Using a DNS

If you are choosing any of the options for using DNS, follow these steps:

**NOTE:** This documentation only applies to the xCAT `makedns` command using the `ddns.pm` plugin. The `ddns.pm` plugin is based on `named9/bind9`, and can not support `named8/bind8` due to syntax difference.

- Set the **nameservers** and **forwarders** attributes in the xCAT site table. The **nameservers** attribute identifies the DNS server hostname/ip that the nodes point to in their `/etc/resolv.conf` files. The **forwarders** attribute are the DNS server's ip that can resolve external hostnames. If you are running a DNS on the xCAT MN, it will use the **forwarders** DNS server to resolve any hostnames it can't.

For example:

```
chdef -t site nameservers=10.0.0.100 forwarders=9.14.8.1,9.14.8.2
```

\* Create an `/etc/resolv.conf` file on the management node

Edit `/etc/resolv.conf` to contain the cluster domain value you set in the site table's **domain** attribute above, and to point to the same DNS server you will be using for your nodes (if you are using DNS).

### Option #1: Running DNS on Your Management Node

This is the most common set up. In this configuration, a DNS running on the management node handles all name resolution requests for cluster node names. A separate DNS in your site handles requests for non-cluster hostnames.

There are several bits of information that must be included in the xCAT database before running the `makedns` command.

You must set the **forwarders** attribute in the xCAT cluster **site** definition. The **forwarders** value should be set to the IP address of one or more **nameservers** at your site that can resolve names outside of your cluster. With this set up, all nodes ask the local nameserver to resolve names, and if it is a name that the management node DNS does not know about, it will try the forwarder names.

An xCAT **network** definition must be defined for each management network used in the cluster. The **net** and **mask** attributes will be used by the `makedns` command.

A network **domain** and **nameservers** value must be provided either in the network definition corresponding to the nodes or in the site definition.

For example, if the cluster domain is **mycluster.com**, the IP address of the management node, (as known by the cluster nodes), is **100.0.0.41** and the site DNS servers are **50.1.2.254,50.1.3.254** then you would run the following command.

```
chdef -t site domain=mycluster.com nameservers=100.0.0.41 forwarders=50.1.2.254,50.1.3.254
```

Once `/etc/hosts` is populated with all of the nodes' hostnames and IP addresses, configure DNS on the management node and start it:

```
makedns -n
```

The **resolv.conf** files for the compute nodes will be created automatically using the **domain** and **nameservers** values set in the xCAT **network** or **site** definition.

If you add nodes or change node names or IP addresses later on, rerun `makedns` which will automatically restart `named`.

To verify the DNS service on management node is working or not:

```
nslookup <host> <mn's ip>
```

For example:

```
nslookup node1 100.0.0.41
```

### Option #2: Use a DNS That is Outside of the Cluster

If you already have a DNS on your site network and you want to use it to solve the node name in your cluster, follow the steps in this section to configure your external dns (against your local dns on xCAT MN/SN).

- Set the site **nameservers** value to the IP address of the external name server.

```
chdef -t site nameservers=<external dns IP>
```

- Set the correct information of external dns into the `/etc/resolv.conf` on your xCAT MN.

The **domain** and **nameservers** values must be set correctly in `/etc/resolv.conf`. Which should have the same values with the ones your set in the site table.

- Manually set up your external dns server with correct `named.conf` and correct zone files
- Add the TSIG to the `named.conf` of your external dns for `makedns` command to update external dns



```
tabdump -w key==omapi passwd
get the key like "omapi","xcat_key",
↪ "MFRCeHJybnJxeVBNaE1YT1BFtFJzN2JuREFMeEMwU0U=",,,
Add it to your named.conf
key xcat_key {
    algorithm hmac-md5;
    secret "MFRCeHJybnJxeVBNaE1YT1BFtFJzN2JuREFMeEMwU0U=";
};
```

- Then change each zone to make your zones to allow this key to update.

```
zone "1.168.192.IN-ADDR.ARPA." in {
    type master;
    allow-update {
        key xcat_key;
    };
    file "db.192.168.1";
};
```

- To update the name resolution entries from /etc/hosts or hosts table of xCAT MN to external DNS, run `makedns -e`

Alternatively, you can set `site.externaldns=1` and run `makedns`

### Option #3: Run DNS on Management Node and Service Nodes

When you have service nodes, the recommended configuration is to run DNS on the management node and all of the service nodes. Two choices are available:

#### Option #3.1: Using the management node as DNS server, the service nodes as forwarding/caching servers.

This means the **DNS** server on the management node is the only one configured with all of the node hostname/IP address pairs. The DNS servers on the service nodes are simply forwarding/caching the DNS requests to the management node.

#### Option #3.2: Using the management node as DNS master, the service nodes as DNS slaves.

This means the **DNS** server on the management node is configured with all of the node hostname/IP address pairs, and allowed to transfer DNS zones to the service nodes. The DNS servers on the service nodes are DNS slaves, so that if the management node goes down for some reason, then you still have the service nodes to be able to do name resolution.

The configurations are described below for the two options, note the differences marked as Option #3.x.

**Note:** for Option #3.1, only the DNS on the management node will use the **forwarders** setting. The DNS servers on the service nodes will always forward requests to the management node.

**Note:** for Option #3.2, make sure **servicenode.nameserver=2** before you run `makedns -n`.

Once `/etc/hosts` is populated with all of the nodes' hostnames and IP addresses, configure DNS on the management node and start it:

```
makedns -n
```

When the `/etc/resolv.conf` files for the compute nodes are created the value of the **nameserver** in `/etc/resolv.conf` is gotten from **site.nameservers** or **networks.nameservers** if it's specified.

For example:

```
chdef -t site nameservers="<xcatmaster>"          # for Option #3.1
OR
chdef -t network <my_network> nameservers="<xcatmaster>"  # for Option #3.1

chdef -t site nameservers="<xcatmaster>, MN_IP"          # for Option #3.2
OR
chdef -t network <my_network> nameservers="<xcatmaster>, MN_IP"  # for Option #3.2
```

The **<xcatmaster>** keyword will be interpreted as the value of the **<xcatmaster>** attribute of the node definition. The **<xcatmaster>** value for a node is the name of it's server as known by the node. This would be either the cluster-facing name of the service node or the cluster-facing name of the management node.

---

**Note:** The site **nameservers** value must be set to **<xcatmaster>** before you run `makedhcp`.

---

Make sure that the DNS service on the service nodes will be set up by xCAT.

Assuming you have all of your service nodes in a group called "service" you could run a command similar to the following.

```
chdef -t group service setupnameserver=1          # for Option #3.1
chdef -t group service setupnameserver=2          # for Option #3.2
```

For Linux systems, make sure DHCP is set up on the service nodes.

```
chdef -t group service setupdhcp=1
```

If you have not yet installed or diskless booted your service nodes, xCAT will take care of configuring and starting DNS on the service nodes at that time. If the service nodes are already running, restarting `xcated` on them will cause xCAT to recognize the above setting and configure/start DNS:

```
xdsh service 'service xcatd restart'  # linux
```

If you add nodes or change node names or IP addresses later on, rerun `makedns`. The DNS on the service nodes will automatically pick up the new information.

## Specifying additional network interfaces for cluster nodes

You can specify additional interface information as part of an xCAT node definition. This information is used by xCAT to populate the `/etc/hosts` file with the extra interfaces (using the `makehosts` command) and providing xCAT adapter configuration scripts with the information required to automatically configure the additional interfaces on the nodes.

To use this support you must set one or more of the following node definition attributes.

```
nicips - IP addresses for additional interfaces (NIC). (Required)
nichostnamesuffixes - Hostname suffixes per NIC. This is a suffix to add to the node_
↳ name to use for the hostname of the additional interface. (Optional)
nictypes - NIC types per NIC. The valid "nictypes" values are: "ethernet", "infiniband",_
↳ and "bmc". (Optional)
niccustomscripts - The name of an adapter configuration postscript to be used to_
```

(continues on next page)

(continued from previous page)

```

↪ configure the interface. (Optional)
nicnetworks - xCAT network definition names corresponding to each NIC. (ie. the network_
↪ that the nic ip resides on.) (Optional)
nicaliases - Additional aliases to set for each additional NIC.
(These are added to the /etc/hosts file when using makehosts). (Optional)

```

The additional NIC information may be set by directly editing the xCAT **nics** table or by using the xCAT **\*defs** commands to modify the node definitions.

The details for how to add the additional information is described below. As you will see, entering this information manually can be tedious and error prone. This support is primarily targeted to be used in conjunction with other IBM products that have tools to fill in this information in an automated way.

## Managing additional interface information using the xCAT **\*defs** commands

The xCAT **\*defs** commands (**mkdef**, **chdef**, and **lsdef**) may be used to manage the additional NIC information in the xCAT database.

When using the these commands the expanded nic\* attribute format will always be used.

### Expanded format for nic\* attributes

The expanded format will be the nics attribute name and the nic name, separated by a “.” (dot).(ie. <nic attr=”” name=””>.<nic name=””> )

For example, the expanded format for the **nicips** and **nichostnamesuffixes** attributes for a nic named **eth1** might be:

```

nicips.eth1=10.1.1.6
nichostnamesuffixes.eth1=-eth1

```

If we assume that your xCAT node name is **compute02** then this would mean that you have an additional interface (“**eth1**”) and that the hostname and IP address are **compute02-eth1** and **10.1.1.6**.

A “[” delimiter is used to specify multiple values for an interface. For example:

```

nicips.eth2=60.0.0.7|70.0.0.7
nichostnamesuffixes.eth2='-eth2|-eth2-lab'

```

This indicates that **eth2** gets two hostnames and two IP addresses. ( **compute02-eth2** gets **60.0.0.7** and **compute02-eth2-lab** gets “**70.0.0.7**”.)

For the **nicaliases** attribute a list of additional aliases may be provided.

```

nicaliases.eth1='alias1 alias2'
nicaliases.eth2='alias3|alias4'

```

This indicates that the **compute02-eth1** hostname would get the additional two aliases, **alias1** **alias2**, included in the **/etc/hosts** file, (when using the **makehosts** command).

The second line indicates that **compute02-eth2** would get the additional alias **alias3** and that **compute02-eth-lab** would get **alias4**

## Setting individual nic attribute values

The nic attribute values may be set using the `chdef` or `mkdef` commands. You can specify the `nic*` values when creating an xCAT node definition with `mkdef` or you can update an existing node definition using `chdef`.

**Note:** `chdef` does not support using the “-m” and “-p” options to modify the `nic*` attributes.

`nicips` example:

```
chdef -t node -o compute02 nicips.eth1=11.10.1.2 nicips.eth2='80.0.0.2|70.0.0.2'
```

NOTE: The management interface (**eth0**), that the **compute02** IP is configured on, is not included in the list of additional nics. Although adding it to the list of nics would do no harm.

This **nicips** value indicates that there are two additional interfaces to be configured on node `compute02`, `eth1` and `eth2`. The **eth1** interface will get the IP address **11.10.1.2**. The **eth2** interface will get two IP addresses, “**80.0.0.2**” and “**70.0.0.2**”.

`nichostnamesuffixes` example:

```
chdef -t node -o compute02 nichostnamesuffixes.eth1=-eth1 nichostnamesuffixes.eth2='-  
→eth2|-eth2-lab'
```

This value indicates that the hostname for “**eth1**” should be “**compute02-eth1**”. For “**eth2**” we had two IP addresses so now we need two suffixes. The hostnames for “**eth2**” will be “**compute02-eth2**” and “**compute02-eth2-lab**”. The IP for “**compute02-eth2**” will be “**80.0.0.2**” and the IP for “**compute02-eth2-lab**” will be “**70.0.0.2**”.

The suffixes provided may be any string that will conform to the DNS naming rules.

**Warning:** According to DNS rules a hostname must be a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9), minus sign (-), and period (.). When you are specifying “**nichostnamesuffixes**” or “**nicaliases**” make sure the resulting hostnames will conform to this naming convention.

`nictypes` example:

```
chdef -t node -o compute02 nictypes.eth1=ethernet nictypes.eth2='ethernet|ethernet'
```

This value indicates that all the nics are ethernet. The valid “**nictypes**” values are: “**ethernet**”, “**infiniband**”, and “**bmc**”.

`niccustomscripsts` example:

```
chdef -t node -o compute02 niccustomscripsts.eth1=cfgeth niccustomscripsts.eth2=  
→'cfgeth|cfgeth'
```

In this example “**cfgeth**” is the name of an adapter configuration postscript to be used to configure the interface.

`nicnetworks` example:

```
chdef -t node -o compute02 nicnetworks.eth1=clstrnet11 nicnetworks.eth2=  
→'clstrnet80|clstrnet-lab'
```

In this example we are saying that the IP address of “**eth1**” (ie. `compute02-eth1` -> 11.10.1.2) is part of the xCAT network named “**clstrnet11**”. “**compute02-eth2**” is in network “**clstrnet80**” and “**compute02-eth2-lab**” is in “**clstrnet-lab**”.

By default the xCAT code will attempt to match the interface IP to one of the xCAT network definitions. An xCAT network definition must be created for all networks being used in the xCAT cluster environment. nicaliases example:

```
chdef -t node -o compute02 nicaliases.eth1="moe larry"
```

In this example it specifies that, (when running `makehosts`), the “`compute02-eth1`” entry in the `/etc/hosts` file should get the additional aliases “`moe`” and “`larry`”.

### Add additional NIC information for a single cluster node

In this example we assume that we have already designated that node “`compute01`” get IP address “`60.0.0.1`” which will be configured on interface “`eth0`”. This will be the xCAT management interface for the node. In addition to the management interface we also wish to include information for the “`eth1`” interface on node “`compute01`”. To do this we must set the additional nic information for this node. For example:

```
chdef -t node -o compute01 nicips.eth1='80.0.0.1' nichostnamesuffixes.eth1='-eth1'
↪ nictypes.eth1='ethernet' nicnetworks.eth1='clstrnet80'
```

This information will be used to configure the “`eth1`” interface, (in addition to the management interface (`eth0`)), during the boot of the node.

Also, if you were to run “`makehosts compute01`” at this point you would see something like the following entries added to the `/etc/hosts` file.

```
60.0.0.1 compute01 compute01.cluster60.com
80.0.0.1 compute01-eth1 compute01-eth1.cluster80.com
```

The domain names are found by checking the xCAT network definitions to see which one would include the IP address. The domain for the matching network is then used for the long name in the `/etc/hosts` file.

NOTE: If you specify the same IP address for a nic as you did for the management interface then the nic hostname will be considered an alias of the xCAT node hostname. For example, if you specified “`60.0.0.1`” for the eth1 “`nicips`” value then the `/etc/hosts` entry would be:

```
60.0.0.1 compute01 compute01.cluster60.com compute01-eth1
```

### Add additional NIC information for a group of nodes

In this example we’d like to configure additional “`eth1`” interfaces for a group of cluster nodes.

The basic approach will be to create an xCAT node group containing all the nodes and then use a regular expression to determine the actual “`nicips`” to use for each node.

For this technique to work you must set up the hostnames and IP address to have a regular pattern. For more information on using regular expressions in the xCAT database see the `xcatdb` man page.

In the following example, the xCAT node group “`compute`” was defined to include all the computational nodes: `compute01`, `compute02`, `compute03` etc. (These hostnames/IPs will be mapped to the “`eth0`” interfaces.)

For the “`eth1`” interfaces on these nodes we’d like to have “`compute01-eth1`” map to “`80.0.0.1`”, and “`compute02-eth1`” to map to “`80.0.0.2`” etc.

To do this we could define the “`compute`” group attributes as follows:

```
chdef -t group -o compute nicips='|\D+(\d+)|eth1!80.0.0.($1+0)|' nichostnamesuffixes=
↳ 'eth1!-eth1' nictypes='eth1!ethernet'
```

These values will be applied to each node in the “**compute**” group. So, for example, if I list the attributes of “**compute08**” I’d see the following **nic\*** attribute values set.

```
lsdef compute08
Object name: compute08
. . .
nicips.eth1=80.0.0.8
nichostnamesuffixes.eth1=-eth1
nictypes.eth1=ethernet
. . .
```

Here is a second example of using regular expressions to define multiple nodes:

```
chdef -t group -o nictest nicips='|\D+(\d+)|ib0!10.4.102.($1*1)|' nichostnamesuffixes=
↳ 'ib0!-ib' nictypes='ib0!Infiniband' nicnetworks='ib0!barcoo_infiniband'

lsdef nictest
Object name: node01
groups=nictest
nichostnamesuffixes.ib0=-ib
nicips.ib0=10.4.102.1
nicnetworks.ib0=barcoo_infiniband
nictypes.ib0=Infiniband
postbootscripts=otherpkgs
postscripts=syslog,remoteshell
```

NOTE: Make sure you haven’t already set **nic\*** values in the individual node definitions since they would take precedence over the group value.

## Using expanded stanza file format

The xCAT stanza file supports the expanded **nic\*** attribute format.

It will contain the **nic\*** attributes as described above.

Example:

```
compute01:
  objtype=node
  arch=x86_64
  mgt=ipmi
  cons=ipmi
  bmc=10.1.0.12
  nictypes.etn0=ethernet
  nicips.eth0=11.10.1.3
  nichostnamesuffixes.eth0=-eth0
  nicnetworks.eth0=clstrnet1
  nictypes.eth1=ethernet
  nicips.eth1=60.0.0.7|70.0.0.7
  nichostnamesuffixes.eth1=-eth1|-eth1-lab
```

(continues on next page)

(continued from previous page)

```
nicnetworks.eth1=clstrnet2|clstrnet3
nicaliases.eth0="alias1 alias2"
nicaliases.eth1="alias3|alias4"
```

The `lsdef` command may be used to create a stanza file in this format and the `chdef/mkdef` commands will read a stanza file in this format.

## Using `lsdef` to display `nic*` attributes

If a node has any `nic` attributes set they will be displayed along with the node definition. The `nic` attribute values are displayed in the expanded format.

```
lsdef compute02
```

If you would only like to see the `nic*` attributes for the node you can specify the “`--nics`” option on the command line.

```
lsdef compute02 --nics
```

If you would like to display individual `nic*` attribute values you can use the “`-i`” option.

You can either specify the base `nic*` attribute name or the expanded name for a specific NIC.

```
lsdef compute05 -i nicips,nichostnamesuffixes
Object name: compute05
  nicips.eth1=11.1.89.7
  nicips.eth2=12.1.89.7
  nichostnamesuffixes.eth1=-lab
  nichostnamesuffixes.eth2=-app

lsdef compute05 -i nicips.eth1,nichostnamesuffixes.eth1
Object name: compute05
  nicips.eth1=11.1.89.7
  nichostnamesuffixes.eth1=-lab
```

## Setting addition interface information using the `xCAT tabedit` command

Another option for setting the `nic` attribute values is to use the `tabedit` command. All the `nic` attributes for a node or group are stored in the xCAT database table named “`nics`”. You can edit the table directly using the `xCAT tabedit` command.

Example:

```
tabedit nics
```

For a description of the `nic*` table attributes see the `nics` table man page.

Sample table contents:

```
#node,nicips,nichostnamesuffixes,nictypes,niccustomscripts,nicnetworks,nicaliases,
↪ comments,disable
"compute03","eth0!11.10.1.3,eth1!60.0.0.7","eth0!-eth0,eth1!-eth1","eth0!ethernet,eth1!
↪ ethernet",,
```

(continues on next page)

(continued from previous page)

```
"eth0!clstrnet11,eth1!clstrnet60",eth0!moe,,
. . . . .
```

## Limited support for user application networks

In some cases you may have additional user application networks in your site that are not specifically used for cluster management. If desired you can create xCAT network definitions for these networks. This not only provides a convenient way to keep track of the network details but the information can also be used to help set up name resolution for these networks on the cluster nodes. When you add a network definition that includes a “**domain**” value then that domain is automatically included the xCAT name resolution set up. This will enable the nodes to be able to resolve hostnames from the other domains.

For example, when you run `makedhcp -n` it will list all domains defined in the xCAT “**site**” definition and xCAT “**network**” definitions in the “**option domain-search**” entry of the shared-network stanza in the dhcp configuration file. This will cause dhcp to put these domains in the compute nodes’ `/etc/resolv.conf` file every time it gets a dhcp lease.

## hostname setting on compute node

After compute node is deployed, its hostname is coming from DHCP, the default hostname is the same with the node name. If you want to have persistent hostname, you can use `confignetwork -s` to configure the install NIC with static IP address, at the same time, it persists hostname on the compute node.

Execute `confignetwork -s` to configure provision IP address as static IP address:

- a. Add `confignetwork -s` into postscript list to execute on reboot

```
chdef cn1 -p postscripts="confignetwork -s"
```

- b. If the compute node is already running, use `updatenode` command to run `confignetwork -s` postscript without rebooting the node

```
updatenode cn1 -P "confignetwork -s"
```

## 1.5.8 GPUs

### NVIDIA CUDA

CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model created by NVIDIA. It can be used to increase computing performance by leveraging the Graphics Processing Units (GPUs).

For more information, see NVIDIA's website: <https://developer.nvidia.com/cuda-zone>

xCAT supports CUDA installation for Ubuntu 14.04.3 and RHEL 7.5 on PowerNV (Non-Virtualized) for both diskful and diskless nodes.

Within the NVIDIA CUDA Toolkit, installing the `cuda` package will install both the `cuda-runtime` and the `cuda-toolkit`. The `cuda-toolkit` is intended for developing CUDA programs and monitoring CUDA jobs. If your particular installation requires only running GPU jobs, it's recommended to install only the `cuda-runtime` package.



## Create CUDA software repository

The NVIDIA CUDA Toolkit is available to download at <http://developer.nvidia.com/cuda-downloads>.

Download the toolkit and prepare the software repository on the xCAT Management Node to server the NVIDIA CUDA files.

### RHEL 7.5

1. Create a repository on the MN node installing the CUDA Toolkit:

```
# For cuda toolkit name: /path/to/cuda-repo-rhel7-9-2-local-9.2.64-1.ppc64le.rpm
# extract the contents from the rpm
mkdir -p /tmp/cuda
cd /tmp/cuda
rpm2cpio /path/to/cuda-repo-rhel7-9-2-local-9.2.64-1.ppc64le.rpm | cpio -i -d

# Create the repo directory under xCAT /install dir for cuda 9.2
mkdir -p /install/cuda-9.2/ppc64le/cuda-core
cp /tmp/cuda/var/cuda-repo-9-2-local/*.rpm /install/cuda-9.2/ppc64le/cuda-core

# Create the yum repo files
createrepo /install/cuda-9.2/ppc64le/cuda-core
```

2. The NVIDIA CUDA Toolkit contains rpms that have dependencies on other external packages (such as DKMS). These are provided by EPEL. It's up to the system administrator to obtain the dependency packages and add those to the cuda-deps directory:

```
mkdir -p /install/cuda-9.2/ppc64le/cuda-deps

# Copy the DKMS rpm to this directory
cp /path/to/dkms-2.4.0-1.20170926git959bd74.el7.noarch.rpm /install/cuda-9.2/
  ↳ ppc64le/cuda-deps

# Execute createrepo in this directory
createrepo /install/cuda-9.2/ppc64le/cuda-deps
```

## Ubuntu 14.04.3

NVIDIA supports two types of debian repositories that can be used to install Cuda Toolkit: **local** and **network**. You can download the installers from <https://developer.nvidia.com/cuda-downloads>.

### Local

A local package repo will contain all of the CUDA packages. Extract the CUDA packages into `/install/cuda-repo/ppc64le`:

```
# For CUDA toolkit: /root/cuda-repo-ubuntu1404-7-5-local_7.5-18_ppc64le.deb

# Create the repo directory under xCAT /install dir
mkdir -p /install/cuda-repo/ppc64le

# extract the package
dpkg -x /root/cuda-repo-ubuntu1404-7-5-local_7.5-18_ppc64le.deb /install/cuda-repo/
↪ppc64le
```

### Network

The online package repo provides a source list entry pointing to a URL containing the CUDA packages. This can be used directly on the Compute Nodes.

The `sources.list` entry may look similar to:

```
deb http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1410/ppc64le /
```

### Authorize the CUDA repo

In order to access the CUDA repository you must import the CUDA GPGKEY into the `apt_key` trust list. xCAT provides a sample postscript `/install/postscripts/addcudakey` to help with this task:

```
chdef -t node -o <noderange> -p postscripts=addcudakey
```

### Create osimage definitions

Generate osimage definitions to provision the compute nodes with the NVIDIA CUDA toolkit installed.

## RHEL 7.5

xCAT provides a sample package list (pkglist) files for CUDA. You can find them:

- Diskful: `/opt/xcat/share/xcat/install/rh/cuda*`
- Diskless: `/opt/xcat/share/xcat/netboot/rh/cuda*`

## Diskful images

The following examples will create diskful images for `cudafull` and `cudaruntime`. The `osimage` definitions will be created from the base `rhels7.5-ppc64le-install-compute` `osimage`.

**[Note]:** There is a requirement to reboot the machine after the CUDA drivers are installed. To satisfy this requirement, the CUDA software is installed in the `pkglist` attribute of the `osimage` definition where a reboot will happen after the Operating System is installed.

### cudafull

1. Create a copy of the `install-compute` image and label it `cudafull`:

```
lsdef -t osimage -z rhels7.5-ppc64le-install-compute \
| sed 's/install-compute:/install-cudafull:/' \
| mkdef -z
```

2. Add the CUDA repo created in the previous step to the `pkgdir` attribute:

```
chdef -t osimage -o rhels7.5-ppc64le-install-cudafull -p \
pkgdir=/install/cuda-9.2/ppc64le/cuda-core,/install/cuda-9.2/ppc64le/cuda-deps
```

3. Use the provided `cudafull` `pkglist` to install the CUDA packages:

```
chdef -t osimage -o rhels7.5-ppc64le-install-cudafull \
pkglist=/opt/xcat/share/xcat/install/rh/cudafull.rhels7.ppc64le.pkglist
```

### cudaruntime

1. Create a copy of the `install-compute` image and label it `cudaruntime`:

```
lsdef -t osimage -z rhels7.5-ppc64le-install-compute \
| sed 's/install-compute:/install-cudaruntime:/' \
| mkdef -z
```

2. Add the CUDA repo created in the previous step to the `pkgdir` attribute:

```
chdef -t osimage -o rhels7.5-ppc64le-install-cudaruntime -p \
pkgdir=/install/cuda-9.2/ppc64le/cuda-core,/install/cuda-9.2/ppc64le/cuda-deps
```

3. Use the provided `cudaruntime` `pkglist` to install the CUDA packages:

```
chdef -t osimage -o rhels7.5-ppc64le-install-cudaruntime \
pkglist=/opt/xcat/share/xcat/install/rh/cudaruntime.rhels7.ppc64le.pkglist
```

## Diskless images

The following examples will create diskless images for `cudafull` and `cudaruntime`. The `osimage` definitions will be created from the base `rhels7.5-ppc64le-netboot-compute` `osimage`.

**[Note]:** For diskless, the install of the CUDA packages **MUST** be done in the `otherpkglist` and **NOT** the `pkglist` as with diskful. The requirement for rebooting the machine is not applicable in diskless nodes because the image is loaded on each reboot.

### cudafull

1. Create a copy of the `netboot-compute` image and label it `cudafull`:

```
lsdef -t osimage -z rhels7.5-ppc64le-netboot-compute \
| sed 's/netboot-compute:/netboot-cudafull:/' \
| mkdef -z
```

2. Verify that the CUDA repo created in the previous step is available in the directory specified by the `otherpkgdir` attribute.

The `otherpkgdir` directory can be obtained by running `lsdef` on the `osimage`:

```
# lsdef -t osimage rhels7.5-ppc64le-netboot-cudafull -i otherpkgdir
Object name: rhels7.5-ppc64le-netboot-cudafull
otherpkgdir=/install/post/otherpkgs/rhels7.5/ppc64le
```

Create a symbolic link of the CUDA repository in the directory specified by `otherpkgdir`

```
ln -s /install/cuda-9.2 /install/post/otherpkgs/rhels7.5/ppc64le/cuda-9.2
```

3. Change the `rootimgdir` for the `cudafull` `osimage`:

```
chdef -t osimage -o rhels7.5-ppc64le-netboot-cudafull \
rootimgdir=/install/netboot/rhels7.5/ppc64le/cudafull
```

4. Create a custom `pkglist` file to install additional operating system packages for your CUDA node.

1. Copy the default compute `pkglist` file as a starting point:

```
mkdir -p /install/custom/netboot/rh/

cp /opt/xcat/share/xcat/netboot/rh/compute.rhels7.ppc64le.pkglist \
/install/custom/netboot/rh/cudafull.rhels7.ppc64le.pkglist
```

2. Edit the `pkglist` file and append any packages you desire to be installed. For example:

```
vi /install/custom/netboot/rh/cudafull.rhels7.ppc64le.pkglist
...
# Additional packages for CUDA
pciutils
```

3. Set the new file as the `pkglist` attribute for the `cudafull` `osimage`:

```
chdef -t osimage -o rhels7.5-ppc64le-netboot-cudafull \
pkglist=/install/custom/netboot/rh/cudafull.rhels7.ppc64le.pkglist
```

5. Create the `otherpkg.pkglist` file to do the install of the CUDA full packages:

1. Create the `otherpkg.pkglist` file for `cudafull`:

```
vi /install/custom/netboot/rh/cudafull.rhels7.ppc64le.otherpkgs.pkglist
# add the following packages
cuda-9.2/ppc64le/cuda-deps/dkms
cuda-9.2/ppc64le/cuda-core/cuda
```

2. Set the `otherpkg.pkglist` attribute for the `cudafull` osimage:

```
chdef -t osimage -o rhels7.5-ppc64le-netboot-cudafull \
    otherpkglist=/install/custom/netboot/rh/cudafull.rhels7.ppc64le.otherpkgs.
    pkglist
```

6. Generate the image:

```
genimage rhels7.5-ppc64le-netboot-cudafull
```

7. Package the image:

```
packimage rhels7.5-ppc64le-netboot-cudafull
```

## cudaruntime

1. Create a copy of the `netboot-compute` image and label it `cudaruntime`:

```
lsdef -t osimage -z rhels7.5-ppc64le-netboot-compute \
| sed 's/netboot-compute:/netboot-cudaruntime:/' \
| mkdef -z
```

2. Verify that the CUDA repo created previously is available in the directory specified by the `otherpkgdir` attribute.

1. Obtain the `otherpkgdir` directory using the `lsdef` command:

```
# lsdef -t osimage rhels7.5-ppc64le-netboot-cudaruntime -i otherpkgdir
Object name: rhels7.5-ppc64le-netboot-cudaruntime
otherpkgdir=/install/post/otherpkgs/rhels7.5/ppc64le
```

2. Create a symbolic link to the CUDA repository in the directory specified by `otherpkgdir`

```
ln -s /install/cuda-9.2 /install/post/otherpkgs/rhels7.5/ppc64le/cuda-9.2
```

3. Change the `rootimgdir` for the `cudaruntime` osimage:

```
chdef -t osimage -o rhels7.5-ppc64le-netboot-cudaruntime \
    rootimgdir=/install/netboot/rhels7.5/ppc64le/cudaruntime
```

4. Create the `otherpkg.pkglist` file to do the install of the CUDA runtime packages:

1. Create the `otherpkg.pkglist` file for `cudaruntime`:

```
vi /install/custom/netboot/rh/cudaruntime.rhels7.ppc64le.otherpkgs.pkglist

# Add the following packages:
cuda-9.2/ppc64le/cuda-deps/dkms
cuda-9.2/ppc64le/cuda-core/cuda-runtime-9-2
```

2. Set the otherpkg.pkglist attribute for the cudaruntime osimage:

```
chdef -t osimage -o rhels7.5-ppc64le-netboot-cudaruntime \
    otherpkglist=/install/custom/netboot/rh/cudaruntime.rhels7.ppc64le.otherpkgs.
    ↪pkglist
```

5. Generate the image:

```
genimage rhels7.5-ppc64le-netboot-cudaruntime
```

6. Package the image:

```
packimage rhels7.5-ppc64le-netboot-cudaruntime
```

## POWER9 Setup

NVIDIA POWER9 CUDA driver need some additional setup. Refer the URL below for details.

<http://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#power9-setup>

xCAT includes a script, `cuda_power9_setup` as example, to help user handle this situation.

## Diskful osimage

For diskful deployment, there is no need to change the osimage definition. Instead, add this postscript to your compute node postscripts list.

```
chdef p9compute -p postscripts=cuda_power9_setup
```

## Diskless osimage

For diskless deployment, the script need to add to the postinstall script of the osimage. And it should be run in the chroot environment. Please refer the following commands as an example.

```
mkdir -p /install/custom/netboot/rh
cp /opt/xcat/share/xcat/netboot/rh/compute.rhels7.ppc64le.postinstall /install/custom/
    ↪netboot/rh/cudafull.rhels7.ppc64le.postinstall

cat >>/install/custom/netboot/rh/cudafull.rhels7.ppc64le.postinstall <<-EOF

/install/postscripts/cuda_power9_setup
EOF

chdef -t osimage rhels7.5-ppc64le-netboot-cudafull postinstall=/install/custom/netboot/
    ↪rh/cudafull.rhels7.ppc64le.postinstall
```

## Ubuntu 14.04.3

### Diskful images

The following examples will create diskful images for `cudafull` and `cudaruntime`. The `osimage` definitions will be created from the base `ubuntu14.04.3-ppc64el-install-compute` `osimage`.

xCAT provides a sample package list files for CUDA. You can find them at:

- `/opt/xcat/share/xcat/install/ubuntu/cudafull.ubuntu14.04.3.ppc64el.pkglist`
- `/opt/xcat/share/xcat/install/ubuntu/cudaruntime.ubuntu14.04.3.ppc64el.pkglist`

**[diskful note]:** There is a requirement to reboot the machine after the CUDA drivers are installed. To satisfy this requirement, the CUDA software is installed in the `pkglist` attribute of the `osimage` definition where the reboot happens after the Operating System is installed.

### cudafull

1. Create a copy of the `install-compute` image and label it `cudafull`:

```
lsdef -t osimage -z ubuntu14.04.3-ppc64el-install-compute \
| sed 's/install-compute:/install-cudafull:/' \
| mkdef -z
```

2. Add the CUDA repo created in the previous step to the `pkgdir` attribute.

If your Management Node IP is 10.0.0.1, the URL for the repo would be `http://10.0.0.1/install/cuda-repo/ppc64el/var/cuda-repo-7-5-local`, add it to the `pkgdir`:

```
chdef -t osimage -o ubuntu14.04.3-ppc64el-install-cudafull \
-p pkgdir=http://10.0.0.1/install/cuda-repo/ppc64el/var/cuda-repo-7-5-local
```

**TODO:** Need to add Ubuntu Port? “`http://ports.ubuntu.com/ubuntu-ports` `trusty` `main,http://ports.ubuntu.com/ubuntu-ports` `trusty-updates` `main`”

3. Use the provided `cudafull` `pkglist` to install the CUDA packages:

```
chdef -t osimage -o ubuntu14.04.3-ppc64el-install-cudafull \
pkglist=/opt/xcat/share/xcat/install/ubuntu/cudafull.ubuntu14.04.3.ppc64el.pkglist
```

### cudaruntime

1. Create a copy of the `install-compute` image and label it `cudaruntime`:

```
lsdef -t osimage -z ubuntu14.04.3-ppc64el-install-compute \
| sed 's/install-compute:/install-cudaruntime:/' \
| mkdef -z
```

2. Add the CUDA repo created in the previous step to the `pkgdir` attribute:

If your Management Node IP is 10.0.0.1, the URL for the repo would be `http://10.0.0.1/install/cuda-repo/ppc64el/var/cuda-repo-7-5-local`, add it to the `pkgdir`:

```
chdef -t osimage -o ubuntu14.04.3-ppc64el-install-cudaruntime \
-p pkgdir=http://10.0.0.1/install/cuda-repo/ppc64el/var/cuda-repo-7-5-local
```

**TODO:** Need to add Ubuntu Port? “<http://ports.ubuntu.com/ubuntu-ports> trusty main,<http://ports.ubuntu.com/ubuntu-ports> trusty-updates main”

3. Use the provided cudaruntime pkglist to install the CUDA packages:

```
chdef -t osimage -o ubuntu14.04.3-ppc64el-install-cudaruntime \
pkglist=/opt/xcat/share/xcat/install/ubuntu/cudaruntime.ubuntu14.04.3.ppc64el.
->pkglist
```

## Diskless images

The following examples will create diskless images for `cudafull` and `cudaruntime`. The `osimage` definitions will be created from the base `ubuntu14.04.3-ppc64el-netboot-compute` `osimage`.

xCAT provides a sample package list files for CUDA. You can find them at:

- `/opt/xcat/share/xcat/netboot/ubuntu/cudafull.ubuntu14.04.3.ppc64el.pkglist`
- `/opt/xcat/share/xcat/netboot/ubuntu/cudaruntime.ubuntu14.04.3.ppc64el.pkglist`

**[diskless note]:** For diskless images, the requirement for rebooting the machine is not applicable because the images is loaded on each reboot. The install of the CUDA packages is required to be done in the `otherpkglist` **NOT** the `pkglist`.

## cudafull

1. Create a copy of the `netboot-compute` image and label it `cudafull`:

```
lsdef -t osimage -z ubuntu14.04.3-ppc64el-netboot-compute \
| sed 's/netboot-compute:/netboot-cudafull:/' \
| mkdef -z
```

2. Add the CUDA repo created in the previous step to the `otherpkgdir` attribute.

If your Management Node IP is 10.0.0.1, the URL for the repo would be `http://10.0.0.1/install/cuda-repo/ppc64el/var/cuda-repo-7-5-local`, add it to the `otherpkgdir`:

```
chdef -t osimage -o ubuntu14.04.3-ppc64el-netboot-cudafull \
otherpkgdir=http://10.0.0.1/install/cuda-repo/ppc64el/var/cuda-repo-7-5-local
```

3. Add the provided `cudafull` `otherpkg.pkglist` file to install the CUDA packages:

```
chdef -t osimage -o ubuntu14.04.3-ppc64el-netboot-cudafull \
otherpkglist=/opt/xcat/share/xcat/netboot/ubuntu/cudafull.otherpkgs.pkglist
```

**TODO:** Need to add Ubuntu Port? “<http://ports.ubuntu.com/ubuntu-ports> trusty main,<http://ports.ubuntu.com/ubuntu-ports> trusty-updates main”

4. Verify that `acpid` is installed on the Management Node or on the Ubuntu host where you are generating the diskless image:



```
apt-get install -y acpid
```

5. Generate the image:

```
genimage ubuntu14.04.3-ppc64el-netboot-cudafull
```

6. Package the image:

```
packimage ubuntu14.04.3-ppc64el-netboot-cudafull
```

## cudaruntime

1. Create a copy of the netboot-compute image and label it cudaruntime:

```
lsdef -t osimage -z ubuntu14.04.3-ppc64el-netboot-compute \
| sed 's/netboot-compute:/netboot-cudaruntime:/' \
| mkdef -z
```

2. Add the CUDA repo created in the previous step to the otherpkgdir attribute.

If your Management Node IP is 10.0.0.1, the URL for the repo would be `http://10.0.0.1/install/cuda-repo/ppc64el/var/cuda-repo-7-5-local`, add it to the otherpkgdir:

```
chdef -t osimage -o ubuntu14.04.3-ppc64el-netboot-cudaruntime \
otherpkgdir=http://10.0.0.1/install/cuda-repo/ppc64el/var/cuda-repo-7-5-local
```

3. Add the provided cudaruntime otherpkg.pkglist file to install the CUDA packages:

```
chdef -t osimage -o ubuntu14.04.3-ppc64el-netboot-cudaruntime \
otherpkglist=/opt/xcat/share/xcat/netboot/ubuntu/cudaruntime.otherpkgs.pkglist
```

**TODO:** Need to add Ubuntu Port? “`http://ports.ubuntu.com/ubuntu-ports` trusty main,`http://ports.ubuntu.com/ubuntu-ports` trusty-updates main”

4. Verify that acpid is installed on the Management Node or on the Ubuntu host where you are generating the diskless image:

```
apt-get install -y acpid
```

5. Generate the image:

```
genimage ubuntu14.04.3-ppc64el-netboot-cudaruntime
```

6. Package the image:

```
packimage ubuntu14.04.3-ppc64el-netboot-cudaruntime
```

## Postscripts

The following sections demonstrates how to use xCAT to configure post-installation steps

### Setting PATH and LD\_LIBRARY\_PATH

NVIDIA recommends various post-installation actions that should be performed to properly configure the nodes. A sample script is provided by xCAT for this purpose `config_cuda` and can be modified to fit your specific installation.

Add this script to your node object using the `chdef` command:

```
chdef -t node -o <noderange> -p postscripts=config_cuda
```

### Setting GPU Configurations

NVIDIA allows for changing GPU attributes using the `nvidia-smi` commands. These settings do not persist when a compute node is rebooted. One way set these attributes is to use an xCAT postscript to set the values every time the node is rebooted.

- Set the power limit to 175W:

```
# set the power limit to 175W
nvidia-smi -pl 175
```

- Set the GPUs to persistence mode to increase performance:

```
# nvidia-smi -pm 1
Enabled persistence mode for GPU 0000:03:00.0.
Enabled persistence mode for GPU 0000:04:00.0.
Enabled persistence mode for GPU 0002:03:00.0.
Enabled persistence mode for GPU 0002:04:00.0.
All done.
```

### Deploy CUDA nodes

#### Diskful

- To provision diskful nodes using `osimage=rhels7.5-ppc64le-install-cudafull`:

```
nodeset <noderange> osimage=rhels7.5-ppc64le-install-cudafull
rsetboot <noderange> net
rpower <noderange> boot
```

## Diskless

- To provision diskless nodes using osimage rhels7.5-ppc64le-netboot-cudafull:

```
nodeset <noderange> osimage=rhels7.5-ppc64le-netboot-cudafull
rsetboot <noderange> net
rpower <noderange> boot
```

## Verify CUDA Installation

The following verification steps only apply to the ``cudafull`` installations.

1. Verify driver version by looking at: /proc/driver/nvidia/version:

```
# cat /proc/driver/nvidia/version
NVRM version: NVIDIA UNIX ppc64le Kernel Module 352.39 Fri Aug 14 17:10:41 PDT
2015
GCC version: gcc version 4.8.5 20150623 (Red Hat 4.8.5-4) (GCC)
```

2. Verify the CUDA Toolkit version

```
# nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2015 NVIDIA Corporation
Built on Tue_Aug_11_14:31:50_CDT_2015
Cuda compilation tools, release 7.5, V7.5.17
```

3. Verify running CUDA GPU jobs by compiling the samples and executing the deviceQuery or bandwidthTest programs.

- Compile the samples:

**[RHEL]:**

```
cd ~/
cuda-install-samples-7.5.sh .
cd NVIDIA_CUDA-7.5_Samples
make
```

**[Ubuntu]:**

```
cd ~/
apt-get install cuda-samples-7-0 -y
cd /usr/local/cuda-7.0/samples
make
```

- Run the deviceQuery sample:

```
# ./bin/ppc64le/linux/release/deviceQuery
./deviceQuery Starting...
CUDA Device Query (Runtime API) version (CUDART static linking)
Detected 4 CUDA Capable device(s)
Device 0: "Tesla K80"
    CUDA Driver Version / Runtime Version      7.5 / 7.5
```

(continues on next page)

(continued from previous page)

```

    CUDA Capability Major/Minor version number:    3.7
    Total amount of global memory:                  11520 MBytes (12079136768
↳bytes)
    (13) Multiprocessors, (192) CUDA Cores/MP:      2496 CUDA Cores
    GPU Max Clock rate:                             824 MHz (0.82 GHz)
    Memory Clock rate:                              2505 Mhz
    Memory Bus Width:                               384-bit
    L2 Cache Size:                                  1572864 bytes
    .....
    deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 7.5, CUDA Runtime
↳Version = 7.5, NumDevs = 4, Device0 = Tesla K80, Device1 = Tesla K80, Device2
↳= Tesla K80, Device3 = Tesla K80
    Result = PASS

```

- Run the bandwidthTest sample:

```

# ./bin/ppc64le/linux/release/bandwidthTest
[CUDA Bandwidth Test] - Starting...
Running on...
Device 0: Tesla K80
Quick Mode
Host to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
  Transfer Size (Bytes)      Bandwidth(MB/s)
  33554432                  7765.1
Device to Host Bandwidth, 1 Device(s)
PINNED Memory Transfers
  Transfer Size (Bytes)      Bandwidth(MB/s)
  33554432                  7759.6
Device to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
  Transfer Size (Bytes)      Bandwidth(MB/s)
  33554432                  141485.3
Result = PASS

```

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.

## GPU Management and Monitoring

The `nvidia-smi` command provided by NVIDIA can be used to manage and monitor GPUs enabled Compute Nodes. In conjunction with the xCAT `xdsh`` command, you can easily manage and monitor the entire set of GPU enabled Compute Nodes remotely from the Management Node.

Example:

```

# xdsh <noderange> "nvidia-smi -i 0 --query-gpu=name,serial,uuid --format=csv,noheader"
node01: Tesla K80, 0322415075970, GPU-b4f79b83-c282-4409-a0e8-0da3e06a13c3
...

```

**Warning:** The following commands are provided as convenience. Always consult the `nvidia-smi` manpage for the latest supported functions.

## Management

Some useful `nvidia-smi` example commands for management.

- Set persistence mode, When persistence mode is enabled the NVIDIA driver remains loaded even when no active clients, DISABLED by default:

```
nvidia-smi -i 0 -pm 1
```

- Disabled ECC support for GPU. Toggle ECC support, A flag that indicates whether ECC support is enabled, need to use `-query-gpu=ecc.mode.pending` to check [Reboot required]:

```
nvidia-smi -i 0 -e 0
```

- Reset the ECC volatile/aggregate error counters for the target GPUs:

```
nvidia-smi -i 0 -p 0/1
```

- Set MODE for compute applications, query with `-query-gpu=compute_mode`:

```
nvidia-smi -i 0 -c 0/1/2/3
```

- Trigger reset of the GPU

```
nvidia-smi -i 0 -r
```

- Enable or disable Accounting Mode, statistics can be calculated for each compute process running on the GPU, query with `-query-gpu=accounting.mode`:

```
nvidia-smi -i 0 -am 0/1
```

- Specifies maximum power management limit in watts, query with `-query-gpu=power.limit`

```
nvidia-smi -i 0 -pl 200
```

## Monitoring

Some useful `nvidia-smi` example commands for monitoring.

- The number of NVIDIA GPUs in the system

```
nvidia-smi --query-gpu=count --format=csv,noheader
```

- The version of the installed NVIDIA display driver

```
nvidia-smi -i 0 --query-gpu=driver_version --format=csv,noheader
```

- The BIOS of the GPU board

```
nvidia-smi -i 0 --query-gpu=vbios_version --format=csv,noheader
```

- Product name, serial number and UUID of the GPU:

```
nvidia-smi -i 0 --query-gpu=name,serial,uuid --format=csv,noheader
```

- Fan speed:

```
nvidia-smi -i 0 --query-gpu=fan.speed --format=csv,noheader
```

- The compute mode flag indicates whether individual or multiple compute applications may run on the GPU. (known as exclusivity modes)

```
nvidia-smi -i 0 --query-gpu=compute_mode --format=csv,noheader
```

- Percent of time over the past sample period during which one or more kernels was executing on the GPU:

```
nvidia-smi -i 0 --query-gpu=utilization.gpu --format=csv,noheader
```

- Total errors detected across entire chip. Sum of device\_memory, register\_file, l1\_cache, l2\_cache and texture\_memory

```
nvidia-smi -i 0 --query-gpu=ecc.errors.corrected.aggregate.total --format=csv,  
↪noheader
```

- Core GPU temperature, in degrees C:

```
nvidia-smi -i 0 --query-gpu=temperature.gpu --format=csv,noheader
```

- The ECC mode that the GPU is currently operating under:

```
nvidia-smi -i 0 --query-gpu=ecc.mode.current --format=csv,noheader
```

- The power management status:

```
nvidia-smi -i 0 --query-gpu=power.management --format=csv,noheader
```

- The last measured power draw for the entire board, in watts:

```
nvidia-smi -i 0 --query-gpu=power.draw --format=csv,noheader
```

- The minimum and maximum value in watts that power limit can be set to

```
nvidia-smi -i 0 --query-gpu=power.min_limit,power.max_limit --format=csv
```

## Update NVIDIA Driver

If the user wants to update the newer NVIDIA driver on the system, follow the [Create CUDA software repository](#) document to create another repository for the new driver.

The following example assumes the new driver is in `/install/cuda-9.2/ppc64le/nvidia_new`.

## Diskful

1. Change pkgdir for the cuda image:

```
chdef -t osimage -o rhels7.5-ppc64le-install-cudafull \
  pkgdir=/install/cuda-9.2/ppc64le/nvidia_new,/install/cuda-9.2/ppc64le/cuda-deps
```

2. Use xdsh command to remove all the NVIDIA rpms:

```
xdsh <noderange> "yum remove *nvidia* -y"
```

3. Run updatenode command to update NVIDIA driver on the compute node:

```
updatenode <noderange> -S
```

4. Reboot compute node:

```
rpower <noderange> off
rpower <noderange> on
```

5. Verify the newer driver level:

```
nvidia-smi | grep Driver
```

## Diskless

To update a new NVIDIA driver on diskless compute nodes, re-generate the osimage pointing to the new NVIDIA driver repository and reboot the node to load the diskless image.

Refer to [Create osimage definitions](#) for specific instructions.

### 1.5.9 High Availability

The xCAT management node plays an important role in the cluster, if the management node is down for whatever reason, the administrators will lose the management capability for the whole cluster, until the management node is back up and running. In some configurations, like the Linux NFSROOT-based statelite in a non-hierarchy cluster, the compute nodes may not be able to run at all without the management node. So, it is important to consider the high availability for the management node.

The goal of the HAMN (High Availability Management Node) configuration is, when the primary xCAT management node fails, the standby management node can take over the role of the management node, either through automatic failover or through manual procedure performed by the administrator, and thus avoid long periods of time during which your cluster does not have active cluster management function available.

The following pages describes ways to configure the xCAT Management Node for High Availability.

## Configuration considerations

xCAT provides several configuration options for the HAMN, you can select one of the option based on your failover requirements and hardware configuration, the following configuration considerations should be able to help you to make the decision.

## Data synchronization mechanism

The data synchronization is important for any high availability configuration. When the xCAT management node failover occurs, the xCAT data needs to be exactly the same before failover, and some of the operating system configuration should also be synchronized between the two management nodes. To be specific, the following data should be synchronized between the two management nodes to make the xCAT HAMN work:

- xCAT database
- xCAT configuration files, like `/etc/xcat`, `~/ .xcat`, `/opt/xcat`
- The configuration files for the services that are required by xCAT, like `named`, `DHCP`, `apache`, `nfs`, `ssh`, etc.
- The operating systems images repository and users customization data repository, the `/install` directory contains these repositories in most cases.

There are a lot of ways for data synchronization, but considering the specific xCAT HAMN requirements, only several of the data synchronization options are practical for xCAT HAMN.

**1. Move physical disks between the two management nodes:** if we could physically move the hard disks from the failed management node to the backup management node, and bring up the backup management node, then both the operating system and xCAT data will be identical between the new management node and the failed management node. RAID1 or disk mirroring could be used to avoid the disk be a single point of failure.

**2. Shared data:** the two management nodes use the single copy of xCAT data, no matter which management node is the primary MN, the cluster management capability is running on top of the single data copy. The access to the data could be done through various ways like shared storage, NAS, NFS, samba etc. Based on the protocol being used, the data might be accessible only on one management node at a time or be accessible on both management nodes in parallel. If the data could only be accessed from one management node, the failover process need to take care of the data access transition; if the data could be accessed on both management nodes, the failover does not need to consider the data access transition, it usually means the failover process could be faster.

Warning: Running database through network file system has a lot of potential problems and is not practical, however, most of the database system provides database replication feature that can be used to synchronize the database between the two management nodes.

**3. Mirroring:** each of the management node has its own copy of the xCAT data, and the two copies of data are synchronized through mirroring mechanism. DRBD is used widely in the high availability configuration scenarios, to provide data replication by mirroring a whole block device via network. If we put all the important data for xCAT onto the DRBD devices, then it could assure the data is synchronized between the two management nodes. Some parallel file system also provides capability to mirror data through network.



## Manual vs. Automatic Failover

When the primary management node fails, the backup management node could automatically take over, or the administrator has to perform some manual procedure to finish the failover. In general, the automatic failover takes less time to detect the failure and perform and failover, comparing with the manual failover, but the automatic failover requires more complex configuration. We could not say the automatic failover is better than the manual failover in all cases, the following factors should be considered when deciding the manual failover or automatic failover:

### 1. How long the cluster could survive if the management node is down?

If the cluster could not survive for more than several minutes, then the automatic failover might be the only option; if the compute nodes could run without the management node, at least for a while, then the manual failover could be an option.

From xCAT perspective, if the management node needs to provide network services like DHCP, named, ntp or nfs to the compute nodes, then the cluster probably could not survive too long if the management node is down; if the management node only performs hardware control and some other management capabilities, then the failed management node may not cause too much trouble for the cluster. xCAT provides various options for configuring if the compute nodes rely on the network services on the management node.

### 2. Configuration complexity

The configuration for the high availability applications is usually complex, it may take a long time to configure, debug and stabilize the high availability configuration.

### 3. Maintenance effort

The automatic failover brings in several high availability applications, after the initial configuration is done, additional maintenance effort will be needed. For example, taking care of the high availability applications during cluster update, the updates for the high availability applications themselves, troubleshooting any problems with the high availability applications. A simple question may be able to help you to decide: could you get technical support if some of the high availability applications run into problems? All software has bugs.

## Configuration Options

The combinations of data synchronization mechanism and manual/automatic failover indicates different HAMN configuration options, the table below list all the combinations (the bold numbers are the combinations xCAT has documented and tested):

#	Move physical disks	Shared data	Mirroring
Manual Failover	<b>1</b>	<b>2</b>	3
Automatic Failover	4	<b>5</b>	<b>6</b>

Option 1, setup\_ha\_mgmt\_node\_with\_raid1\_and\_disks\_move

Option 2, setup\_ha\_mgmt\_node\_with\_shared\_data

Option 3, it is doable but not currently supported.

Option 4, it is not practical.

Option 5, setup\_xcat\_high\_available\_management\_node\_with\_nfs

Option 6, setup\_ha\_mgmt\_node\_with\_drbd\_pacemaker\_corosync

### 1.5.10 Hierarchical Clusters / Large Cluster Support

xCAT supports management of very large sized cluster by creating a **Hierarchical Cluster** and the concept of **xCAT Service Nodes**.

When dealing with large clusters, to balance the load, it is recommended to have more than one node (Management Node, “MN”) handling the installation and management of the Compute Nodes (“CN”). These additional *helper* nodes are referred to as **Service Nodes** (“SN”). The Management Node can delegate all management operational needs to the Service Node responsible for a set of compute node.

**The following configurations are supported:**

- Each service node installs/manages a specific set of compute nodes
- Having a pool of service nodes, any of which can respond to an installation request from a compute node (*Requires service nodes to be aligned with networks broadcast domains, compute node chooses service nodes based on who responds to DHCP request first.*)
- A hybrid of the above, where each specific set of compute nodes have 2 or more service nodes in a pool

The following documentation assumes an xCAT cluster has already been configured and covers the additional steps needed to support xCAT Hierarchy via Service Nodes.

#### Service Nodes

Service Nodes are similar to the xCAT Management Node in that each service Nodes runs an instance of the xCAT daemon: `xcatd`. `xcatd`'s communicate with each other using the same XML/SSL protocol that the xCAT client uses to communicate with `xcatd` on the Management Node.

The Service Nodes need to communicate with the xCAT database running on the Management Node. This is done using the remote client capabilities of the database. This is why the default SQLite database cannot be used.

The xCAT Service Nodes are installed with a special xCAT package `xcatsn` which tells `xcatd` running on the node to behave as a Service Node and not the Management Node.

#### Configure a Database

##### SQLite

xCAT uses the SQLite database (<https://www.sqlite.org/>) as the default database and it is initialized during xCAT installation of the Management Node. If using Service Nodes, SQLite **cannot** be used because Service Nodes require remote access to the xCAT database. One of the following databases should be used:

- *MySQL/MariaDB*
- *PostgreSQL*

## MySQL/MariaDB

### Install MySQL/MariaDB

The MySQL database is supported by xCAT since xCAT 2.1. MariaDB is a fork of the MySQL project which was released around 2009 and is a drop-in replacement for MySQL. MariaDB support within xCAT started in version 2.8.5 and is currently fully supported moving forward.

Database	MySQL	MariaDB
xCAT 2.1+	Yes	No
xCAT 2.8.5	Yes	RHEL 7
xCAT 2.9	Yes	SLES 12
xCAT 2.10+	Yes	Yes

MySQL/MariaDB packages are shipped as part of most Linux Distributions.

### Red Hat Enterprise Linux

- For RHEL 6 and prior, MySQL is shipped. Using yum, ensure that the following packages are installed on the management node:

```
perl-DBD-MySQL*
mysql-server-5.*
mysql-5.*
mysql-connector-odbc-*
```

- For RHEL 7, MariaDB is shipped. Using yum, ensure that the following packages are installed on the management node:

```
mariadb-server-5.*
mariadb-5.*
perl-DBD-MySQL*
mysql-connector-odbc-*
```

- For RHEL 8, MariaDB is shipped. Using dnf, ensure that the following packages are installed on the management node:

```
mariadb-server-10.*
mariadb-10.*
perl-DBD-MySQL*
mariadb-connector-odbc-*
```

## Suse Linux Enterprise Server

- MySQL - Using `zypper`, ensure that the following packages are installed on the management node:

```
mysql-client-5*
libmysqlclient_r15*
libqt4-sql-mysql-4*
libmysqlclient15-5*
perl-DBD-mysql-4*
mysql-5*
```

- MariaDB - Using `zypper`, ensure that the following packages are installed on the management node:

```
mariadb-client-10.*
mariadb-10.*
mariadb-errormessages-10.*
libqt4-sql-mysql-*
libmysqlclient18-*
perl-DBD-mysql-*
```

- For SLE15, MariaDB is shipped. Using `zypper`, ensure that the following packages are installed on the management node:

```
mariadb
mariadb-client
mariadb-errormessages
perl-DBD-mysql
libmariadb-devel
mariadb-tools
libmariadb_plugins
```

## Debian/Ubuntu

- MySQL - Using `apt-get`, ensure that the following packages are installed on the management node:

```
mysql-server
mysql-common
libdbd-mysql-perl
libmysqlclient*
mysql-client-5*
mysql-client-core-5*
mysql-server-5*
mysql-server-core-5*
```

- MariaDB - Using `apt-get`, ensure that the following packages are installed on the management node. `apt-get install mariadb-server` will pull in all required packages. For Ubuntu 16.04, it no longer required `libmariadbclient18`.

```
libmariadbclient18
mariadb-client
mariadb-common
mariadb-server
```

## Configure MySQL/MariaDB

### Migrate xCAT to use MySQL/MariaDB

The following utility is provided to migrate an existing xCAT database from SQLite to MySQL/MariaDB.

```
mysqlsetup -i
```

If you need to update the database at a later time to give access to your service nodes, you can use the `mysqlsetup -u -f` command. A file needs to be provided with all the hostnames and/or IP addresses of the servers that need to access the database on the Management node. Wildcards can be used.

```
mysqlsetup -u -f /tmp/servicenodes
```

where the `/tmp/servicenodes` contains a host per line:

```
cat /tmp/servicenodes
node1
1.115.85.2
10.%.%.%
node2.cluster.net
```

**While not recommended**, if you wish to manually migrate your xCAT database, see the following documentation: [Manually set up MySQL](#)

### Granting/Revoking access to the database for Service Node Clients

- Log into the MySQL interactive program.

```
/usr/bin/mysql -u root -p
```

- Granting access to the xCAT database. Service Nodes are required for xCAT hierarchical support. Compute nodes may also need access that depends on which application is going to run. (xcat201 is xcatadmin's password for following examples)

```
MariaDB > GRANT ALL on xcatdb.* TO xcatadmin@<servicenode(s)> IDENTIFIED BY 'xcat201'
↪;
```

Use the wildcards to do a GRANT ALL to every ipaddress or nodename that need to access the database.

```
MariaDB > GRANT ALL on xcatdb.* TO xcatadmin@'%.cluster.net' IDENTIFIED BY 'xcat201'
↪;
MariaDB > GRANT ALL on xcatdb.* TO xcatadmin@'8.113.33.%' IDENTIFIED BY 'xcat201';
```

- To revoke access, run the following:

```
MariaDB > REVOKE ALL on xcatdb.* FROM xcatadmin@'8.113.33.%;
```

- To Verify the user table was populated.

```
MariaDB > SELECT host, user FROM mysql.user;
```

## Using MySQL/MariaDB

### Start/Stop MySQL/MariaDB service

[RHEL] for MariaDB:

```
service mariadb start
service mariadb stop
```

[RHEL] for MySQL:

```
service mysqld start
service mysqld stop
```

[SLES] and [Ubuntu]:

```
service mysql start
service mysql stop
```

[SLE15] :

```
systemctl start mariadb
systemctl stop mariadb
```

### Basic MySQL/MariaDB commands

Refer to <https://www.mariadb.org/> for the latest documentation.

- Using mysql, connect to the xcat database:

```
mysql -u root -p
```

- List the hosts and users which managed by this xcat MN:

```
MariaDB> SELECT host, user FROM mysql.user;
```

- List the databases:

```
MariaDB> SHOW DATABASES;
```

- Use the xcatdb:

```
MariaDB> use xcatdb;
```

- List all the tables:

```
MariaDB [xcatdb]> SHOW TABLES;
```

- Show the entries in the nodelist table:

```
MariaDB [xcatdb]> select * from nodelist;
```

- Quit mysql:

```
MariaDB [xcatdb]> quit;
```

## Removing xcatdb from MySQL/MariaDB

If you no longer want to use MySQL/MariaDB to maintain xcatdb, and like to switch to PostgreSQL or just default SQLite ( **Note:** SQLite does not support xCAT Hierarchy (has service nodes)), use the following documentation as guide to remove xcatdb.

- Run a backup of the database to save any information that is needed (optional):

```
mkdir -p ~/xcat-dbback
dumpxCATdb -p ~/xcat-dbback
```

If you want to restore this database later:

```
XCATBYPASS=1 restorexCATdb -p ~/xcat-dbback
```

- To switch to PostgreSQL, follow: [Install PostgreSQL](#)
- To switch to default xCAT database, SQLite (**Note:** xCAT Hierarchy cluster will no longer work):

1. Stop the xcatd daemon on the management node.

```
service xcatd stop
```

2. Remove the xatdb from MySQL/MariaDB (optional):

```
/usr/bin/mysql -u root -p
```

drop the xcatdb:

```
mysql> drop database xcatdb;
```

remove the xcatadm database owner :

```
mysql> drop user xcatadm;
```

3. Move, or remove, the /etc/xcat/cfgloc file as it points xCAT to MySQL/MariaDB. (without this file, xCAT defaults to SQLite):

```
rm /etc/xcat/cfgloc
```

4. Restart xcatd:

```
service xcatd start
```

## PostgreSQL

### Install PostgreSQL

PostgreSQL packages are shipped as part of most Linux Distributions.

### Red Hat Enterprise Linux

Using yum, install the following rpms:

```
yum install postgresql*  
yum install perl-DBD-Pg
```

### Suse Linux Enterprise Server

**Note:** On SLES, perl-DBD packages are provided on the SDK iso images.

Using zypper, install the following rpms:

```
zypper install postgresql*  
zypper install perl-DBD-Pg
```

### Debian/Ubuntu

Using apt, install the following packages:

```
apt install postgresql libdbd-pg-perl
```

### Configure PostgreSQL

### Migrate xCAT to use PostgreSQL

A utility is provided to migrate an existing xCAT database from SQLite to PostgreSQL.

```
pgsqlsetup -i -V
```

**While not recommended**, if you wish to manually migrate your xCAT database, see the following documentation: [Manually set up PostgreSQL](#)



## Setting up the Service Nodes

For service nodes, add the IP address of each service nodes to the postgres configuration file: `/var/lib/pgsql/data/pg_hba.conf`

If you had the following two service nodes:

```
sn10, ip: 192.168.1.10 with netmask 255.255.255.0
sn11, ip: 192.168.1.11 with netmask 255.255.255.0
```

You would add the following to `/var/lib/pgsql/data/pg_hba.conf`

```
host    all             all             192.168.1.10/32      md5
host    all             all             192.168.1.11/32      md5
```

Restart PostgreSQL after editing the file:

```
service postgresql restart
```

For more information about changing the `pg_hba.conf` file and `postgresql.conf` files, see the following documentation: [Setup the PostgreSQL Configuration Files](#)

## Modify PostgreSQL database directory

1. Check the xcatdb have been switched to pgsql:

```
lsxcatd -a
Version 2.13.6 (git commit f8c0d11ff2c7c97d6e62389c0aafcdfa06cee1f6, built Mon_
Aug 7 07:15:47 EDT 2017)
This is a Management Node
cfgloc=Pg:dbname=xcatdb;host=10.3.5.100|xcatadm
dbengine=Pg
dbname=xcatdb
dbhost=10.3.5.100
dbadmin=xcatadm
```

2. Check the current working directory:

```
sudo -u postgres psql
could not change directory to "/root"
psql (9.2.18)
Type "help" for help.

postgres=# SHOW data_directory;
 data_directory
-----
/var/lib/pgsql/data
(1 row)

postgres=# \q
```

3. Stop postgresql service and modify the configuration files:

```
systemctl stop postgresql
mkdir /install/pgsql_db
rsync -av /var/lib/pgsql/ /install/pgsql_db/

cat /usr/lib/systemd/system/postgresql.service | grep -i PGDATA=
    Environment=PGDATA=**/install/pgsql_db/data/**

cat /install/pgsql_db/data/postgresql.conf | grep data_directory
#data_directory = 'ConfigDir'          # use data in another directory
**data_directory = '/install/pgsql_db/data/'**
```

4. Reload, start postgresql service and check the working directory:

```
systemctl daemon-reload
systemctl start postgresql
sudo -u postgres psql
psql (9.2.18)
Type "help" for help.

postgres=# SHOW data_directory;
data_directory
-----
/install/pgsql_db/data
(1 row)

postgres=#
```

## Using PostgreSQL

Refer to <http://www.postgresql.org/> for the latest documentation.

Using psql, connect to the xcat database:

```
su - postgres
psql -h <hostname> -U xcatadm -d xcatdb (default pw: cluster)
```

list the xCAT tables:

```
xcatdb=> \dt
```

show the entries in the nodelist table:

```
xcatdb=> select * from nodelist;
```

quit postgres:

```
xcatdb=> \q
```

## Useful Commands

Show the SQL create statement for a table:

```
/usr/bin/pg_dump_xcatdb -U xcatadm -t <table_name>

# example, for prescripts table:
/usr/bin/pg_dump xcatdb -U xcatadm -t prescripts
```

List all databases in postgres:

```
su - postgres
psql -l
```

## Removing xcatdb from PostgreSQL and restoring data into SQLite

To remove xcatdb completely from the PostgreSQL database and restore xCAT data into SQLite:

1. Run a backup of the database to save any information that is needed:

```
mkdir -p ~/xcat-dbback
dumpxCATdb -p ~/xcat-dbback
```

2. Stop the xcatd daemon on the management node.

```
service xcatd stop
```

3. Remove the xatdb from PostgreSQL:

```
su - postgres
```

drop the xcatdb:

```
dropdb xcatdb
```

remove the xcatadm database owner :

```
dropuser xcatadm
```

clean up the postgresql files (necessary if you want to re-create the database):

```
cd /var/lib/pgsql/data
rm -rf *
exit
```

4. Move, or remove, the /etc/xcat/cfgloc file as it points xCAT to PostgreSQL. (without this file, xCAT defaults to SQLite):

```
mv /etc/xcat/cfgloc /etc/xcat/cfgloc.postgres
```

5. Restore the PostgreSQL database into SQLite:

```
XCATBYPASS=1 restorexCATdb -p ~/xcat-dbback
```

6. Restart xcatd:

```
service xcatd start
```

## Define Service Nodes

This next part shows how to configure a xCAT Hierarchy and provision xCAT service nodes from an existing xCAT cluster.

**Note:** The document assumes that the compute nodes that are part of your cluster have already been defined into the xCAT database and you have successfully provisioned the compute nodes using xCAT

The following table illustrates the cluster being used in this example:

Operating System	rhels7.1
Architecture	ppc64le
xCAT Management Node	xcat01
Compute Nodes (group=rack1)	r1n01 r1n02 r1n03 ... r1n10
Compute Nodes (group=rack2)	r2n01 r2n02 r2n03 ... r2n10

1. Modify site table attribute to include **service** group's postscripts in compute node definition:

```
chdef -t site hierarchicalattrs="postscripts"
```

2. Select the compute nodes that will become service nodes

The first node in each rack, **r1n01** and **r2n01**, is selected to become the xCAT service nodes and manage the compute nodes in that rack

3. Change the attributes for the compute node to make them part of the **service** group:

```
chdef -t node -o r1n01,r2n01 -p groups=service
```

4. When copycds was run against the ISO image, several osimages were created into the osimage table. The ones named \*-service are provided to easily help provision xCAT service nodes.

```
# lsdef -t osimage | grep rhels7.1
rhels7.1-ppc64le-install-compute (osimage)
rhels7.1-ppc64le-install-service (osimage) <=====
rhels7.1-ppc64le-netboot-compute (osimage)
```

5. Add some common service node attributes to the service nodegroup:

```
chdef -t group -o service setupnfs=1 \
                           setupdhcp=1 \
                           setupftp=1 \
                           setupnameserver=1 \
                           setupconserver=2
```

### Tips/Hint

- Even if you do not want xCAT to configure any services, you must define the service nodes in the servicenode table with at least one attribute, set to 0, otherwise xCAT will not recognize the node as a service node
- See the setup\* attributes in the node definition man page for the list of available services: `man node`

- For clusters with subnetted management networks, you might want to set `setupipforward=1`
- For the `setupconserver` attribute, if `conserver` is used, set to 1, if `goconserver` is used, set to 2

#### 6. Add additional postscripts for Service Nodes (optional)

By default, xCAT will execute the `servicenode` postscript when installed or diskless booted. This postscript will set up the necessary credentials and install the xCAT software on the Service Nodes. If you have additional postscripts that you want to execute on the service nodes, copy to `/install/postscripts` and run the following:

```
chdef -t group -o service -p postscripts=<mypostscript>
```

#### 7. Assigning Compute Nodes to their Service Nodes

The node attributes `servicenode` and `xcatmaster`, define which Service node will serve the particular compute node.

- `servicenode` - defines which Service Node the **Management Node** should send commands to (e.g `xdsh`) and should be set to the hostname or IP address of the service node that the management node can contact it by.
- `xcatmaster` - defines which Service Node the **Compute Node** should boot from and should be set to the hostname or IP address of the service node that the compute node can contact it by.

You must set both `servicenode` and `xcatmaster` regardless of whether or not you are using service node pools. For most scenarios, the value will be identical.

```
chdef -t group -o rack1 servicenode=r1n01 xcatmaster=r1n01
chdef -t group -o rack2 servicenode=r2n01 xcatmaster=r2n01
```

#### 8. Set the `conserver` and `monserver` attributes

Set which service node should run the `conserver` (console) and `monserver` (monitoring) daemon for the nodes in the group. The most typical setup is to have the service node also act as it's `conserver` and `monserver`.

```
chdef -t group -o rack1 conserver=r1n01 monserver=r1n01
chdef -t group -o rack2 conserver=r2n01 monserver=r2n01
```

#### 9. Choose location of `/install` and `/tftpboot` directories (optional).

The `site` table attributes `installloc` and `sharedtftp` control mounting of `/install` and `/tftpboot` directories from Management Node to Service Node.

To mount `/install` and `/tftpboot` directories from Management node to each Service Node:

```
chdef -t site clustersite sharedtftp=1
chdef -t site clustersite installloc="/install"
```

To make `/install` and `/tftpboot` directories local on each Service Node, set `site` table attributes and “sync” `/install` and `/tftpboot` directory contents from Management Node to Service Nodes:

```
chdef -t site clustersite sharedtftp=0
chdef -t site clustersite installloc=
rsync -auv --exclude 'autoinst' /install r1n01:/
rsync -auv --exclude 'autoinst' /install r2n01:/
rsync -auv /tftpboot r1n01:/
rsync -auv /tftpboot r2n01:/
```

**Note:** If `/install` and `/tftpboot` directories local to each Service Node are used and `mknb` command is executed to generate a diskless network boot image with custom changes, it will not be automatically copied to the Service Node. Make sure to run the above `rsync` commands after executing the `mknb`. Verify `/tftpboot/xcat` directory on Service node contains `genesis.kernel.<arch>` and `genesis.fs.<arch>.gz` files.

---

## Configure DHCP

Add the relevant networks into the DHCP configuration, refer to: [Configure xCAT](#)

Add the defined nodes into the DHCP configuration, refer to [Manually define nodes](#)

In the large cluster, the size of dhcp lease file `/var/lib/dhcpd/dhcpd.leases` on the DHCP server will grow over time. At around 100MB in size, the DHCP server will take a long time to respond to DHCP requests from clients and cause DHCP timeouts:

```
...
Mar  2 01:59:10 c656ems2 dhcpd: DHCPDISCOVER from 00:0a:f7:73:7d:d0 via eth0
Mar  2 01:59:10 c656ems2 dhcpd: DHCPOFFER on 9.114.39.101 to 00:0a:f7:73:7d:d0 via eth0
Mar  2 01:59:10 c656ems2 dhcpd: DHCPREQUEST for 9.114.39.101 (9.114.39.157) from_
↪00:0a:f7:73:7d:d0 via eth0
Mar  2 01:59:10 c656ems2 dhcpd: DHCPACK on 9.114.39.101 to 00:0a:f7:73:7d:d0 via eth0
Mar  2 01:59:10 c656ems2 dhcpd: DHCPDECLINE of 9.114.39.101 from 00:0a:f7:73:7d:d0 via_
↪eth0: not found
Mar  2 01:59:10 c656ems2 dhcpd: DHCPDISCOVER from 00:0a:f7:73:7d:d0 via eth0
Mar  2 01:59:10 c656ems2 dhcpd: DHCPOFFER on 9.114.39.101 to 00:0a:f7:73:7d:d0 via eth0
Mar  2 01:59:10 c656ems2 dhcpd: DHCPREQUEST for 9.114.39.101 (9.114.39.157) from_
↪00:0a:f7:73:7d:d0 via eth0
Mar  2 01:59:10 c656ems2 dhcpd: DHCPACK on 9.114.39.101 to 00:0a:f7:73:7d:d0 via eth0
Mar  2 01:59:10 c656ems2 dhcpd: DHCPDECLINE of 9.114.39.101 from 00:0a:f7:73:7d:d0 via_
↪eth0: not found
Mar  2 01:59:10 c656ems2 dhcpd: DHCPDISCOVER from 00:0a:f7:73:7d:d0 via eth0
Mar  2 01:59:10 c656ems2 dhcpd: DHCPOFFER on 9.114.39.101 to 00:0a:f7:73:7d:d0 via eth0
Mar  2 01:59:10 c656ems2 dhcpd: DHCPREQUEST for 9.114.39.101 (9.114.39.157) from_
↪00:0a:f7:73:7d:d0 via eth0
Mar  2 01:59:10 c656ems2 dhcpd: DHCPACK on 9.114.39.101 to 00:0a:f7:73:7d:d0 via eth0
Mar  2 01:59:10 c656ems2 dhcpd: DHCPDECLINE of 9.114.39.101 from 00:0a:f7:73:7d:d0 via_
↪eth0: not found
...
```

The solution is simply to restart the dhcpd service or run `makedhcp -n`.

## Provision Service Nodes

### Diskful (Stateful) Installation

Any cluster using statelite compute nodes must use a stateful (diskful) Service Nodes.

---

**Note:** All xCAT Service Nodes must be at the exact same xCAT version as the xCAT Management Node.

---

## Configure otherpkgdir and otherpkglist for service node osimage

- Create a subdirectory `xcat` under a path specified by `otherpkgdir` attribute of the service node os image, selected during the *Define Service Nodes* step.

For example, for osimage `rhels7-x86_64-install-service`

```
[root@fs4 xcat]# lsdef -t osimage rhels7-x86_64-install-service -i otherpkgdir
Object name: rhels7-x86_64-install-service
otherpkgdir=/install/post/otherpkgs/rhels7/x86_64
[root@fs4 xcat]# mkdir -p /install/post/otherpkgs/rhels7/x86_64/xcat
```

- Download or copy `xcat-core` and `xcat-dep` .bz2 files into that `xcat` directory

```
wget https://xcat.org/files/xcat/xcat-core/<version>_Linux/xcat-core/xcat-core-
-><version>-linux.tar.bz2
wget https://xcat.org/files/xcat/xcat-dep/<version>_Linux/xcat-dep-<version>-linux.
->tar.bz2
```

- untar the `xcat-core` and `xcat-dep` .bz2 files

```
cd /install/post/otherpkgs/<os>/<arch>/xcat
tar jxvf core-rpms-snap.tar.bz2
tar jxvf xcat-dep-*.tar.bz2
```

- Verify the following entries are included in the package file specified by the `otherpkglist` attribute of the service node osimage.

```
xcat/xcat-core/xCATsn
xcat/xcat-dep/<os>/<arch>/conserver-xcat
xcat/xcat-dep/<os>/<arch>/perl-Net-Telnet
xcat/xcat-dep/<os>/<arch>/perl-Expect
```

For example, for the osimage `rhels7-x86_64-install-service`

```
lsdef -t osimage rhels7-x86_64-install-service -i otherpkglist
Object name: rhels7-x86_64-install-service
otherpkglist=/opt/xcat/share/xcat/install/rh/service.rhels7.x86_64.otherpkgs.
->pkglist

cat /opt/xcat/share/xcat/install/rh/service.rhels7.x86_64.otherpkgs.pkglist
xcat/xcat-core/xCATsn
xcat/xcat-dep/rh7/x86_64/conserver-xcat
xcat/xcat-dep/rh7/x86_64/perl-Net-Telnet
xcat/xcat-dep/rh7/x86_64/perl-Expect
```

**Note:** You will be installing the xCAT Service Node RPM `xCATsn-<version>` on the Service Node, not the xCAT Management Node RPM `xCAT-<version>`. Do not install xCAT Management Node RPM `xCAT-<version>` on the Service Node.

## Update the rhels6 RPM repository (rhels6 only)

- This section could be removed after the powerpc-utils-1.2.2-18.el6.ppc64.rpm is built in the base rhels6 ISO.
- The direct rpm download link is: <ftp://linuxpatch.ncsa.uiuc.edu/PERCS/powerpc-utils-1.2.2-18.el6.ppc64.rpm>
- The update steps are as following: - put the new rpm in the base OS packages

```
cd /install/rhels6/ppc64/Server/Packages
mv powerpc-utils-1.2.2-17.el6.ppc64.rpm /tmp
cp /tmp/powerpc-utils-1.2.2-18.el6.ppc64.rpm .
# make sure that the rpm is be readable by other users
chmod +r powerpc-utils-1.2.2-18.el6.ppc64.rpm
```

- create the repodata

```
cd /install/rhels6/ppc64/Server
ls -al repodata/
total 14316
dr-xr-xr-x 2 root root 4096 Jul 20 09:34 .
dr-xr-xr-x 3 root root 4096 Jul 20 09:34 ..
-r--r--r-- 1 root root 1305862 Sep 22 2010_
↪20dfb74c144014854d3b16313907ebcf30c9ef63346d632369a19a4add8388e7-other.sqlite.bz2
-r--r--r-- 1 root root 1521372 Sep 22 2010_
↪57b3c81512224bbb5cebbfcb6c7fd1f7eb99cca746c6c6a76fb64c64f47de102-primary.xml.gz
-r--r--r-- 1 root root 2823613 Sep 22 2010_
↪5f664ea798d1714d67f66910a6c92777ecbbe0bf3068d3026e6e90cc646153e4-primary.sqlite.
↪bz2
-r--r--r-- 1 root root 1418180 Sep 22 2010_
↪7cec82d8ed95b8b60b3e1254f14ee8e0a479df002f98bb557c6ccad5724ae2c8-other.xml.gz
-r--r--r-- 1 root root 194113 Sep 22 2010_
↪90cbb67096e81821a2150d2b0a4f3776ab1a0161b54072a0bd33d5cadd1c234a-comps-rhel6-
↪Server.xml.gz
*-r--r--r-- 1 root root 1054944 Sep 22 2010_
↪98462d05248098ef1724eddb2c0a127954aade64d4bb7d4e693cff32ab1e463c-comps-rhel6-
↪Server.xml**
-r--r--r-- 1 root root 3341671 Sep 22 2010_
↪bb3456b3482596ec3aa34d517affc42543e2db3f4f2856c0827d88477073aa45-filelists.sqlite.
↪bz2
-r--r--r-- 1 root root 2965960 Sep 22 2010_
↪eb991fd2bb9af16a24a066d840ce76365d396b364d3cdc81577e4cf6e03a15ae-filelists.xml.gz
-r--r--r-- 1 root root 3829 Sep 22 2010 repomd.xml
-r--r--r-- 1 root root 2581 Sep 22 2010 TRANS.TBL
createrepo \
-g repodata /98462d05248098ef1724eddb2c0a127954aade64d4bb7d4e693cff32ab1e463c-comps-
↪rhel6-Server.xml
```

---

**Note:** You should use comps-rhel6-Server.xml with its key as the group file.

---



## Install Service Nodes

```
rinstall <service_node> osimage="<osimagenam>"
```

For example

```
rinstall <service_node> osimage="rhels7-x86_64-install-service"
```

## Monitor the Installation

Watch the installation progress using either `wcons` or `rcons` and monitor log messages:

```
wcons service      # make sure DISPLAY is set to your X server/VNC or
rcons <node_name>
tail -f /var/log/messages
```

**Note:** We have experienced one problem while trying to install RHEL6 diskful Service Node working with SAS disks. The Service Node cannot reboot from SAS disk after the RHEL6 operating system has been installed. We are waiting for the build with fixes from RHEL6 team, once meet this problem, you need to manually select the SAS disk to be the first boot device and boots from the SAS disk.

## Update Service Node Diskful Image

To update the xCAT software on the Service Node:

1. Remove previous `xcat-core`, `xcat-dep`, and tar files in the NFS mounted `/install/post/otherpkgs/` directory:

```
rm -rf /install/post/otherpkgs/<os>/<arch>/xcat/xcat-core
rm -rf /install/post/otherpkgs/<os>/<arch>/xcat/xcat-dep
rm /install/post/otherpkgs/<os>/<arch>/xcat/<xcat-core.tar>
rm /install/post/otherpkgs/<os>/<arch>/xcat/<xcat-dep.tar>
```

2. Download the desired tar files from <http://xcat.org/download.html> to the Management Node, and untar them in the same NFS mounted `/install/post/otherpkgs/` directory:

```
cd /install/post/otherpkgs/<os>/<arch>/xcat/
tar jxvf <new-xcat-core.tar>
tar jxvf <new-xcat-dep.tar>
```

3. On the Service Node, run the package manager commands relative to the OS to update xCAT. For example, on RHEL, use the following `yum` commands:

```
yum clean metadata # or yum clean all
yum update '*xCAT*'
```

## Diskless (Stateless) Installation

---

**Note:** The stateless Service Node is not supported in Ubuntu hierarchy cluster. For Ubuntu, skip this section.

---

If you want, your Service Nodes can be stateless (diskless). The Service Node must contain not only the OS, but also the xCAT software and its dependencies. In addition, a number of files are added to the Service Node to support the PostgreSQL, or MySQL database access from the Service Node to the Management node, and ssh access to the nodes that the Service Nodes services. The following sections explain how to accomplish this.

### Build the Service Node Diskless Image

This section assumes you can build the stateless image on the management node because the Service Nodes are the same OS and architecture as the management node. If this is not the case, you need to build the image on a machine that matches the Service Node's OS architecture.

- Create an osimage definition.

When you run `copycds`, xCAT will only create a Service Node stateful osimage definitions for that distribution. For a stateless Service Node osimage, you may create it from a stateless Compute Node osimage definition.

```
lsdef -t osimage | grep -i netboot
rhels7.3-ppc64le-netboot-compute (osimage)

mkdef -t osimage -o rhels7.3-ppc64le-netboot-service \
      --template rhels7.3-ppc64le-netboot-compute \
      profile=service provmethod=netboot postscripts=servicenode

lsdef -t osimage rhels7.3-ppc64le-netboot-service
Object name: rhels7.3-ppc64le-netboot-service
  exlist=/opt/xcat/share/xcat/netboot/rh/compute.rhels7.ppc64le.exlist
  imagetype=linux
  osarch=ppc64le
  osdistrname=rhels7.3-ppc64le
  osname=Linux
  osvers=rhels7.3
  otherpkgdir=/install/post/otherpkgs/rhels7.3/ppc64le
  pkgdir=/install/rhels7.3/ppc64le
  pkglist=/opt/xcat/share/xcat/netboot/rh/compute.rhels7.ppc64le.pkglist
  postinstall=/opt/xcat/share/xcat/netboot/rh/compute.rhels7.ppc64le.postinstall
  postscripts=servicenode
  profile=service
  provmethod=netboot
  rootingdir=/install/netboot/rhels7.3/ppc64le/compute
```

- Configure mandatory attributes for Service Node osimage definition.

The following attributes must be modified to suitable value:

```
exlist
otherpkglist
pkglist
postinstall
rootingdir
```

### 1. Create the `exlist`, `pkglist` and `otherpkglist` files.

xCAT ships a basic requirements lists that will create a fully functional Service Node. However, you may want to customize your service node by adding additional operating system packages or modifying the files excluded by the exclude list. Check the below files to see if it meets your needs.

```
cd /opt/xcat/share/xcat/netboot/rh/
view service.rhels7.ppc64le.pkglist
view service.rhels7.ppc64le.otherpkgs.pkglist
view service.rhels7.ppc64le.exlist
```

If you would like to change any of these files, copy them to a custom directory. This can be any directory you choose, but we recommend that you keep it `/install` somewhere. A good location is something like `/install/custom/netboot/<osimage>`.

```
export OSIMAGE_DIR=/install/custom/netboot/rhels7.3-ppc64le-netboot-service
mkdir -p $OSIMAGE_DIR

cp /opt/xcat/share/xcat/netboot/rh/service.rhels7.ppc64le.pkglist $OSIMAGE_DIR
cp /opt/xcat/share/xcat/netboot/rh/service.rhels7.ppc64le.otherpkgs.pkglist
  ↪ $OSIMAGE_DIR
cp /opt/xcat/share/xcat/netboot/rh/service.rhels7.ppc64le.exlist $OSIMAGE_DIR
```

And make sure that your `otherpkgs.pkglist` file has the following entries:

```
xcat/xcat-core/xCATsn
xcat/xcat-dep/<os>/<arch>/conserver-xcat
xcat/xcat-dep/<os>/<arch>/perl-Net-Telnet
xcat/xcat-dep/<os>/<arch>/perl-Expect
```

For example, for the `osimage rhels7.3-ppc64le-netboot-service`:

```
xcat/xcat-core/xCATsn
xcat/xcat-dep/rh7/ppc64le/conserver-xcat
xcat/xcat-dep/rh7/ppc64le/perl-Net-Telnet
xcat/xcat-dep/rh7/ppc64le/perl-Expect
```

**Note:** You will be installing the xCAT Service Node RPM `xCATsn-<version>` on the Service Node, not the xCAT Management Node RPM `xCAT-<version>`. Do not install xCAT Management Node RPM `xCAT-<version>` on the Service Node.

### 1. Create the `postinstall` script.

xCAT ships a default `postinstall` script for stateless Service Node. You may also choose to create an appropriate `/etc/fstab` file in your Service Node image. :

```
export OSIMAGE_DIR=/install/custom/netboot/rhels7.3-ppc64le-netboot-service
cp /opt/xcat/share/xcat/netboot/rh/service.rhels7.ppc64le.postinstall $OSIMAGE_
  ↪ DIR

vi $OSIMAGE_DIR/service.rhels7.ppc64le.postinstall
# uncomment the sample fstab lines and change as needed:
proc /proc proc rw 0 0
```

(continues on next page)

(continued from previous page)

```
sysfs /sys sysfs rw 0 0
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
service_ppc64le / tmpfs rw 0 1
none /tmp tmpfs defaults,size=10m 0 2
none /var/tmp tmpfs defaults,size=10m 0 2
```

1. Modify the Service Node osimage definition with given attributes.

```
export OSIMAGE_DIR=/install/custom/netboot/rhels7.3-ppc64le-netboot-service
chdef -t osimage -o rhels7.3-ppc64le-netboot-service \
    exlist=$OSIMAGE_DIR/service.rhels7.ppc64le.exlist \
    otherpkglist=$OSIMAGE_DIR/service.rhels7.ppc64le.otherpkgs.pkglist \
    pkglist=$OSIMAGE_DIR/service.rhels7.ppc64le.pkglist \
    postinstall=$OSIMAGE_DIR/service.rhels7.ppc64le.postinstall \
    rootimgdir=$OSIMAGE_DIR/service

lsdef -t osimage -l rhels7.3-ppc64le-netboot-service
Object name: rhels7.3-ppc64le-netboot-service
    exlist=/install/custom/netboot/rhels7.3-ppc64le-netboot-service/service.
↪rhels7.ppc64le.exlist
    imagetype=linux
    osarch=ppc64le
    osdistrname=rhels7.3-ppc64le
    osname=Linux
    osvers=rhels7.3
    otherpkgdir=/install/post/otherpkgs/rhels7.3/ppc64le
    otherpkglist=/install/custom/netboot/rhels7.3-ppc64le-netboot-service/
↪service.rhels7.ppc64le.otherpkgs.pkglist
    pkgdir=/install/rhels7.3/ppc64le
    pkglist=/install/custom/netboot/rhels7.3-ppc64le-netboot-service/service.
↪rhels7.ppc64le.pkglist
    postinstall=/install/custom/netboot/rhels7.3-ppc64le-netboot-service/
↪service.rhels7.ppc64le.postinstall
    postscripts=servicenode
    profile=service
    provmethod=netboot
    rootimgdir=/install/custom/netboot/rhels7.3-ppc64le-netboot-service/service
```

While you are here, if you'd like, you can do the same for your Service Node images, creating custom files and new custom osimage definitions as you need to.

- Make your xCAT software available for otherpkgs processing

Option 1:

If you downloaded xCAT to your management node for installation, place a copy of your xcat-core and xcat-dep in your otherpkgdir directory

```
lsdef -t osimage -o rhels7.3-ppc64le-netboot-service -i otherpkgdir
Object name: rhels7.3-ppc64le-netboot-service
    otherpkgdir=/install/post/otherpkgs/rhels7.3/ppc64le
cd /install/post/otherpkgs/rhels7.3/ppc64le
mkdir xcat
cd xcat
```

(continues on next page)

(continued from previous page)

```
cp -Rp <current location of xcat-core>/xcat-core
cp -Rp <current location of xcat-dep>/xcat-dep
```

Option 2:

If you installed your management node directly from the online repository, you will need to download the xcat-core and xcat-dep tarballs

- From <http://xcat.org/download.html>, download the xcat-core and xcat-dep tarball files. Copy these into a subdirectory in the otherpkgdir directory.

```
lsdef -t osimage -o rhels7.3-ppc64le-netboot-service -i otherpkgdir
Object name: rhels7.3-ppc64le-netboot-service
      otherpkgdir=/install/post/otherpkgs/rhels7.3/ppc64le

cd /install/post/otherpkgs/rhels7.3/ppc64le
mkdir xcat
cd xcat

# copy the <xcat-core> and <xcat-dep> tarballs here

# extract the tarballs
tar -jxvf <xcat-core>.tar.bz2
tar -jxvf <xcat-dep>.tar.bz2
```

- Run image generation for your osimage definition:

```
genimage rhels7.3-ppc64le-netboot-service
```

- Prevent DHCP from starting up until xcatd has had a chance to configure it:

```
export OSIMAGE_ROOT=/install/custom/netboot/rhels7.3-ppc64le-netboot-service/service
chroot $OSIMAGE_ROOT/rootimg chkconfig dhcpd off
chroot $OSIMAGE_ROOT/rootimg chkconfig dhcrelay off
```

- IF using NFS hybrid mode, export /install read-only in Service Node image:

```
export OSIMAGE_ROOT=/install/custom/netboot/rhels7.3-ppc64le-netboot-service/service
cd $OSIMAGE_ROOT/rootimg/etc
echo '/install *(ro,no_root_squash,fsid=13)' >exports
```

- Pack the image for your osimage definition:

```
packimage rhels7.3-ppc64le-netboot-service
```

## Install Service Nodes

```
rinstall service osimage=rhels7.3-ppc64le-netboot-service
```

Watch the installation progress using either `wcons` or `rcons` and monitor log messages:

```
wcons service      # make sure DISPLAY is set to your X server/VNC or
rcons <node_name>
tail -f /var/log/messages
```

## Enable localdisk for stateless Service Node (Optional)

If you want, you can leverage local disk to contain some directories during the stateless nodes running. And you can customize the `osimage` definition to achieve it. For Service Node, it is recommended to put below directories on local disk.

```
#/install           (Not required when using shared /install directory)
#/tftpboot         (Not required when using shared /tftpboot directory)
/var/log
/tmp
```

The following section explains how to accomplish this.

- Change the Service Node `osimage` definition to enable `localdisk`

```
#create a partition file to partition and mount the disk
export OSIMAGE=rhels7.3-ppc64le-netboot-service
cat<<EOF > /install/custom/netboot/$OSIMAGE/partitionfile
enable=yes
enablepart=yes

[disk]
dev=/dev/sda
clear=yes
parts=10,50

[localspace]
dev=/dev/sda2
fstype=ext4

[swapspace]
dev=/dev/sda1
EOF

#add the partition file to Service Node osimage definition and configure ``policy`` table
chdef -t osimage -o $OSIMAGE partitionfile=/install/custom/netboot/$OSIMAGE/partitionfile
chtab priority=7.1 policy.commands=getpartition policy.rule=allow

#define files or directories which are required to be put on local disk
#ctab litefile.image=$OSIMAGE litefile.file=/install/ litefile.options=localdisk
#ctab litefile.image=$OSIMAGE litefile.file=/tftpboot/ litefile.options=localdisk
ctab litefile.image=$OSIMAGE litefile.file=/var/log/ litefile.options=localdisk
ctab litefile.image=$OSIMAGE litefile.file=/tmp/ litefile.options=localdisk
```

- Run image generation and repacking for your osimage definition:

```
genimage rhels7.3-ppc64le-netboot-service
packimage rhels7.3-ppc64le-netboot-service
```

**Note:** `enablepart=yes` in partition file will partition the local disk at every boot. If you want to preserve the contents on local disk at next boot, change to `enablepart=no` after the initial provision.

For more information on `localdisk` option, refer to *Enabling the localdisk option*

## Update Service Node Stateless Image

To update the xCAT software in the image at a later time:

- Download the updated `xcat-core` and `xcat-dep` tarballs from <http://xcat.org/download.html> and place them in your osimage's `otherpkgdir` xcat directory as you did above.
- Generate and repack the image.
- Reinstall your Service Node.

```
genimage "<osimagename>"
packimage "<osimagename>"
rinstall service osimage="<osimagename>"
```

**Note:** The Service Nodes are set up as NFS-root servers for the compute nodes. Any time changes are made to any compute image on the mgmt node it will be necessary to sync all changes to all Service Nodes. In our case the `/install` directory is mounted on the Service Nodes, so the update to the compute node image is automatically available.

## Verify Service Node Installation

- ssh to the service nodes. You should not be prompted for a password.
- Check to see that the xcat daemon `xcatd` is running.
- Run some database command on the service node, e.g `tabdump site`, or `nodels`, and see that the database can be accessed from the service node.
- Check that `/install` and `/tftpboot` are mounted on the service node from the Management Node, if appropriate.
- Make sure that the Service Node has name resolution for all nodes it will service.
- Run `updatenode <compute node> -V -s` on management node and verify output contains `Running command on <service node>` that indicates the command from management node is sent to service node to run against compute node target.

See *Appendix B* for possible solutions.

## Appendix

### Appendix A: Setup backup Service Nodes

For reliability, availability, and serviceability purposes you may wish to designate backup Service Nodes in your hierarchical cluster. The backup Service Node will be another active Service Node that is set up to easily take over from the original Service Node if a problem occurs. This is not an automatic failover feature. You will have to initiate the switch from the primary Service Node to the backup manually. The xCAT support will handle most of the setup and transfer of the nodes to the new Service Node. This procedure can also be used to simply switch some compute nodes to a new Service Node, for example, for planned maintenance.

#### Initial deployment

Integrate the following steps into the hierarchical deployment process described above.

1. Make sure both the primary and backup service nodes are installed, configured, and can access the MN database.
2. When defining the CNs add the necessary service node values to the “servicenode” and “xcatmaster” attributes of the *node* definitions.
3. (Optional) Create an xCAT group for the nodes that are assigned to each SN. This will be useful when setting node attributes as well as providing an easy way to switch a set of nodes back to their original server.

To specify a backup service node you must specify a comma-separated list of two **service nodes** for the servicenode value of the compute node. The first one is the primary and the second is the backup (or new SN) for that node. Use the hostnames of the SNs as known by the MN.

For the **xcatmaster** value you should only include the primary SN, as known by the compute node.

In most hierarchical clusters, the networking is such that the name of the SN as known by the MN is different than the name as known by the CN. (If they are on different networks.)

The following example assume the SN interface to the MN is on the “a” network and the interface to the CN is on the “b” network. To set the attributes you would run a command similar to the following.

```
chdef <noderange> servicenode="xcatsn1a,xcatsn2a" xcatmaster="xcatsn1b"
```

The process can be simplified by creating xCAT node groups to use as the <noderange> in the *chdef* command to create a xCAT node group containing all the nodes that belong to service node “SN27”. For example:

```
mkdef -t group sn1group members=node[01-20]
```

**Note:** Normally backup service nodes are the primary SNs for other compute nodes. So, for example, if you have 2 SNs, configure half of the CNs to use the 1st SN as their primary SN, and the other half of CNs to use the 2nd SN as their primary SN. Then each SN would be configured to be the backup SN for the other half of CNs.

When you run *makedhcp* command, it will configure dhcp and tftp on both the primary and backup SNs, assuming they both have network access to the CNs. This will make it possible to do a quick SN takeover without having to wait for replication when you need to switch.



## xdcp Behaviour with backup servicenodes

The xdcp command in a hierarchical environment must first copy (scp) the files to the service nodes for them to be available to scp to the node from the service node that is its master. The files are placed in /var/xcatsyncfiles directory by default, or what is set in site table SNsyncfiledir attribute. If the node has multiple service nodes assigned, then xdcp will copy the file to each of the service nodes assigned to the node. For example, here the files will be copied (scp) to both service1 and rhsn. lsdef cn4 | grep servicenode.

```
servicenode=service1,rhsn
```

If a service node is offline (e.g. service1), then you will see errors on your xdcp command, and yet if rhsn is online then the xdcp will actually work. This may be a little confusing. For example, here service1 is offline, but we are able to use rhsn to complete the xdcp.

```
xdcp cn4 /tmp/lissa/file1 /tmp/file1

service1: Permission denied (publickey,password,keyboard-interactive).
service1: Permission denied (publickey,password,keyboard-interactive).
service1: lost connection
The following servicenodes: service1, have errors and cannot be updated
Until the error is fixed, xdcp will not work to nodes serviced by these service nodes.

xdsh cn4 ls /tmp/file1
cn4: /tmp/file1
```

## Switch to the backup SN

When an SN fails, or you want to bring it down for maintenance, use this procedure to move its CNs over to the backup SN.

### Move the nodes to the new service nodes

Use the *snmove* command to make the database changes necessary to move a set of compute nodes from one Service Node to another.

To switch all the compute nodes from Service Node sn1 to the backup Service Node sn2, run:

```
snmove -s sn1
```

## Modified database attributes

The snmove command will check and set several node attribute values.

- **servicenode:** This will be set to either the second server name in the servicenode attribute list or the value provided on the command line.
- **xcatmaster:** Set with either the value provided on the command line or it will be automatically determined from the servicenode attribute.
- **nfsserver:** If the value is set with the source service node then it will be set to the destination service node.
- **tftpserver:** If the value is set with the source service node then it will be reset to the destination service node.

- **monserver:** If set to the source service node then reset it to the destination servicenode and xcatmaster values.
- **conserver:** If set to the source service node then reset it to the destination servicenode and run `makeconservercf`

### Run postscripts on the nodes

If the CNs are up at the time the `snmove` command is run then `snmove` will run postscripts on the CNs to reconfigure them for the new SN. The “syslog” postscript is always run. The `mkresolvconf` and `setupntp` scripts will be run if they were included in the nodes postscript list.

You can also specify an additional list of postscripts to run.

### Modify system configuration on the nodes

If the CNs are up the `snmove` command will also perform some configuration on the nodes such as setting the default gateway and modifying some configuration files used by xCAT.

### Switching back

The process for switching nodes back will depend on what must be done to recover the original service node. If the SN needed to be reinstalled, you need to set it up as an SN again and make sure the CN images are replicated to it. Once you’ve done this, or if the SN’s configuration was not lost, then follow these steps to move the CNs back to their original SN:

- Use `snmove`:

```
snmove snlgroup -d sn1
```

## Appendix B: Diagnostics

- **root ssh keys not setup** – If you are prompted for a password when ssh to the service node, then check to see if `/root/.ssh` directory on MN has `authorized_keys` file. If the directory does not exist or no keys, run `xdsh service -K`, to exchange the ssh keys for root. You will be prompted for the root password, which should be the password you set for the `key=system` in the `passwd` table.
- **XCAT rpms not on SN** – On the SN, run `rpm -qa | grep xCAT` and make sure the appropriate xCAT rpms are installed on the servicenode. See the list of xCAT rpms in *Diskful (Stateful) Installation*. If rpms are missing, check your install setup as outlined in *Diskless (Stateless) Installation* for diskless or *Diskful (Stateful) Installation* for diskful installs.
- **otherpkgs(including xCAT rpms) installation failed on the SN** – The OS repository is not created on the SN. When the “yum” command is processing the dependency, the rpm packages (including `expect`, `nmap`, and `httpd`, etc) required by `XCATsn` can’t be found. In this case, check whether the `/install/postscripts/repos/<osver>/<arch>/` directory exists on the MN. If it is not on the MN, you need to re-run the `copycds` command, and there will be files created under the `/install/postscripts/repos/<osver>/<arch>` directory on the MN. Then, you need to re-install the SN.
- **Error finding the database/starting xcatd** – If on the Service node when you run `tabdump site`, you get “Connection failure: IO::Socket::SSL: connect: Connection refused at `/opt/xcat/lib/perl/xCAT/Client.pm`”. Then restart the `xcatd` daemon and see if it passes by running the command `service xcatd restart`. If it fails with the same error, then check to see if `/etc/xcat/cfgloc` file exists. It should exist and be the same as `/etc/xcat/cfgloc` on the MN. If it is not there, copy it from the MN to the SN. Then run `service xcatd`

restart. This indicates the servicenode postscripts did not complete successfully. Run `lsdef <service node> -i postscripts -c` and verify `servicenode` postscript appears on the list.

- **Error accessing database/starting xcatd credential failure**– If you run `tabdump site` on the service node and get “Connection failure: IO::Socket::SSL: SSL connect attempt failed because of handshake problem-error:14094418:SSL routines:SSL3\_READ\_BYTES:tlsv1 alert unknown at /opt/xcat/lib/perl/xCAT/Client.pm”, check `/etc/xcat/cert`. The directory should contain the files `ca.pem` and `server-cred.pem`. These were suppose to transfer from the MN `/etc/xcat/cert` directory during the install. Also check the `/etc/xcat/ca` directory. This directory should contain most files from the `/etc/xcat/ca` directory on the MN. You can manually copy them from the MN to the SN, recursively. This indicates the the servicenode postscripts did not complete successfully. Run `lsdef <service node> -i postscripts -c` and verify `servicenode` postscript appears on the list. Run `service xcatd restart` again and try the `tabdump site` again.
- **Missing ssh hostkeys** – Check to see if `/etc/xcat/hostkeys` on the SN, has the same files as `/etc/xcat/hostkeys` on the MN. These are the ssh keys that will be installed on the compute nodes, so root can ssh between compute nodes without password prompting. If they are not there copy them from the MN to the SN. Again, these should have been setup by the servicenode postscripts.
- **Errors running hierarchical commands such as xdash** – xCAT has a number of commands that run hierarchically. That is, the commands are sent from `xcatd` on the management node to the correct service node `xcatd`, which in turn processes the command and sends the results back to `xcatd` on the management node. If a hierarchical command such as `xcatd` fails with something like “Error: Permission denied for request”, check `/var/log/messages` on the management node for errors. One error might be “Request matched no policy rule”. This may mean you will need to add policy table entries for your xCAT management node and service node.
- **/install is not mounted on service node from management mode** – If service node does not have `/install` directory mounted from management node, run `lsdef -t site clustersite -i installloc` and verify `installloc="/install"`

## Appendix C: Migrating a Management Node to a Service Node

Directly converting an existing Management Node to a Service Node may have some issues and is not recommended. Do the following steps to convert the xCAT Management Node into a Service node:

1. backup your xCAT database on the Management Node
2. Install a new xCAT Management node
3. Restore your xCAT database into the new Management Node
4. Re-provision the old xCAT Management Node as a new Service Node

## Appendix D: Set up Hierarchical Conserver

To allow you to open the rcons from the Management Node using the conserver daemon on the Service Nodes, do the following:

- Set `nodehm.conserver` to be the service node (using the ip that faces the management node)

```
chdef -t <noderange> conserver=<servicenodeasknownbytheMN>
makeconservercf
service conserver stop
service conserver start
```

### 1.5.11 Software Kits

xCAT supports a unique software bundling concept called **software kits**. Software kit combines all of the required product components (packages, license, configuration, scripts, etc) to assist the administrator in the installation of software onto machines managed by xCAT. Software kits are made up of a collection of “kit components”, each of which is tailored to one specific environment for that particular version of the software product.

Prebuilt software kits are available as a tar file which can be downloaded and then added to the xCAT installation. After the kits are added to xCAT, kit components are then added to specific xCAT osimages to automatically install the software bundled with the kit during OS deployment. In some instances, software kits may be provided as partial kits. Partial kits need additional effort to complete the kit before it can be used by xCAT.

Software kits are supported for both diskful and diskless image provisioning.

#### Prebuilt Kits - IBM High Performance Computing

The IBM High Performance Computing (HPC) software can be installed by xCAT using Software Kits. Most products ship the complete Software Kit on the distribution media and can be consumed “as is”.

#### Quick Start Guide

This quick start is provided to guide users through the steps required to install the IBM High Performance Computing (HPC) software stack on a cluster managed by xCAT. (*NOTE: xCAT provides XLC and XLF partial kits, but all other HPC kits are provided by the HPC products teams, xCAT may not have any knowledges of their dependencies and requirements*)

The following software kits will be used to install the IBM HPC software stack on to a RedHat Enterprise Linux 7.2 operating system running on ppc64le architecture.

- xlc-13.1.3-0-ppc64le.tar.bz2<sup>1</sup>
- xlf-15.1.3-0-ppc64le.tar.bz2<sup>1</sup>
- pperte-2.3.0.0-1547a-ppc64le.tar.bz2
- pperte-2.3.0.2-s002a-ppc64le.tar.bz2
- essl-5.4.0-0-ppc64le.tar.bz2
- pessl-5.2.0-0-ppc64le.tar.bz2
- ppedev-2.2.0-0.tar.bz2

1. Using the `addkit` command, add each software kit package into xCAT:

```
addkit xlc-13.1.3-0-ppc64le.tar.bz2,xlf-15.1.3-0-ppc64le.tar.bz2
addkit pperte-2.3.0.0-1547a-ppc64le.tar.bz2,pperte-2.3.0.2-s002a-ppc64le.tar.bz2
addkit pessl-5.2.0-0-ppc64le.tar.bz2,essl-5.4.0-0-ppc64le.tar.bz2
addkit ppedev-2.2.0-0.tar.bz2
```

The `lskit` command can be used to view the kits after adding to xCAT.

2. Using the `addkitcomp` command, add the kitcomponent to the target osimage.

The order that the kit components are added to the osimage is important due to dependencies that kits may have with one another, a feature to help catch potential issues ahead of time. There are a few different types of dependencies:

<sup>1</sup> This guide assumes that the **complete** software kit is available for all the products listed below. For the IBM XL compilers, follow the *IBM XL Compiler* documentation to obtain the software and create the **complete** kit before proceeding.

- **internal kit dependencies** - kit components within the software kit have dependencies. For example, the software has a dependency on its license component. The `-a` option will automatically resolve internal kit dependencies.
- **external kit dependencies** - a software kit depends on another software provided in a separate kit. The dependency kit must be added first. `addkitcomp` will complain if it cannot resolve the dependency.
- **runtime dependencies** - the software provided in the kit has rpm requirements for external 3rd party RPMs not shipped with the kit. The administrator needs to configure these before deploying the osimage and `addkitcomp` cannot detect this dependencies.

In the following examples, the `rhels7.2-ppc64le-install-compute` osimage is used and the `-a` option is specified to resolve internal dependencies.

1. Add the **XLC** kitcomponents to the osimage:

```
addkitcomp -a -i rhels7.2-ppc64le-install-compute \
xlc.compiler-compute-13.1.3-0-rhels-7.2-ppc64le
```

2. Add the **XLf** kitcomponents to the osimage:

```
addkitcomp -a -i rhels7.2-ppc64le-install-compute \
xlf.compiler-compute-15.1.3-0-rhels-7.2-ppc64le
```

3. Add the PE RTE GA, **pperte-1547a**, kitcomponents to the osimage:

```
addkitcomp -a -i rhels7.2-ppc64le-install-compute \
pperte-login-2.3.0.0-1547a-rhels-7.2-ppc64le

addkitcomp -a -i rhels7.2-ppc64le-install-compute \
pperte-compute-2.3.0.0-1547a-rhels-7.2-ppc64le

addkitcomp -a -i rhels7.2-ppc64le-install-compute \
min-pperte-compute-2.3.0.0-1547a-rhels-7.2-ppc64le
```

4. Add the PE RTE PTF2, **pperte-s002a**, kitcomponents to the osimage.

The PTF2 update requires the `pperte-license` component, which is provided by the GA software kit. The `addkitcomp -n` option allows for multiple versions of the same kit component to be installed into the osimage. If only the PTF2 version is intended to be installed, you can skip the previous step for adding the GA ppetre kit component, but the GA software kit must have been added to xCAT with the `addkit` command in order to resolve the license dependency.

```
addkitcomp -a -n -i rhels7.2-ppc64le-install-compute \
pperte-login-2.3.0.2-s002a-rhels-7.2-ppc64le

addkitcomp -a -n -i rhels7.2-ppc64le-install-compute \
pperte-compute-2.3.0.2-s002a-rhels-7.2-ppc64le

addkitcomp -a -n -i rhels7.2-ppc64le-install-compute \
min-pperte-compute-2.3.0.2-s002a-rhels-7.2-ppc64le
```

5. Add the **ESSL** kitcomponents to the osimage.

The ESSL software kit has an *external dependency* to the `libxlf` which is provided in the XLf software kit. Since it's already added in the above step, there is no action needed here.

If CUDA toolkit is being used, ESSL has a runtime dependency on the CUDA rpms. The administrator needs to create the repository for the CUDA 7.5 toolkit or a runtime error will occur when provisioning the node. See the [Create CUDA software repository](#) section for more details about setting up the CUDA repository on the xCAT management node.

```
#
# Assuming that the cuda repo has been configured at:
# /install/cuda-7.5/ppc64le/cuda-core
#
chdef -t osimage rhels7.2-ppc64le-install-compute \
    pkgdir=/install/rhels7.2/ppc64le,/install/cuda-7.5/ppc64le

addkitcomp -a -i rhels7.2-ppc64le-install-compute \
    essl-computenode-6464rte-5.4.0-0-rhels-7.2-ppc64le

addkitcomp -a -i rhels7.2-ppc64le-install-compute \
    essl-computenode-3264rte-5.4.0-0-rhels-7.2-ppc64le

addkitcomp -a -i rhels7.2-ppc64le-install-compute \
    essl-computenode-5.4.0-0-rhels-7.2-ppc64le

addkitcomp -a -i rhels7.2-ppc64le-install-compute \
    essl-loginnode-5.4.0-0-rhels-7.2-ppc64le

addkitcomp -a -i rhels7.2-ppc64le-install-compute \
    essl-computenode-3264rtecuda-5.4.0-0-rhels-7.2-ppc64le
```

If the system doesn't have GPU and the CUDA toolkit is not needed, the administrator should not add the following kit components that requires the CUDA packages: `essl-loginnode-5.4.0-0-rhels-7.2-ppc64le`, `essl-computenode-3264rte-5.4.0-0-rhels-7.2-ppc64le` and `essl-computenode-3264rtecuda-5.4.0-0-rhels-7.2-ppc64le`. Check the ESSL installation guide: [http://www.ibm.com/support/knowledgecenter/SSFHY8\\_5.4.0/com.ibm.cluster.essl.v5r4.essl300.doc/am5il\\_xcatinstall.htm](http://www.ibm.com/support/knowledgecenter/SSFHY8_5.4.0/com.ibm.cluster.essl.v5r4.essl300.doc/am5il_xcatinstall.htm)

1. Add the **Parallel ESSL** kitcomponents to osimage.

**Note:** ESSL kitcomponents are required for the PESSL.

```
addkitcomp -a -i rhels7.2-ppc64le-install-compute \
    pessel-loginnode-5.2.0-0-rhels-7.2-ppc64le

addkitcomp -a -i rhels7.2-ppc64le-install-compute \
    pessel-computenode-5.2.0-0-rhels-7.2-ppc64le

addkitcomp -a -i rhels7.2-ppc64le-install-compute \
    pessel-computenode-3264rtempich-5.2.0-0-rhels-7.2-ppc64le
```

2. Add the **PE DE** kitcomponents to osimage:

```
addkitcomp -a -i rhels7.2-ppc64le-install-compute \
    ppedev.login-2.2.0-0-rhels-7.2-ppc64le

addkitcomp -a -i rhels7.2-ppc64le-install-compute \
    ppedev.compute-2.2.0-0-rhels-7.2-ppc64le
```

3. The updated osimage now contains the configuration to install using xCAT software kits:

```
lsdef -t osimage rhels7.2-ppc64le-install-compute
  Object name: rhels7.2-ppc64le-install-compute
  exlist=/install/osimages/rhels7.2-ppc64le-install-compute-kits/kits/KIT_
  COMPONENTS.exlist
  imagetype=linux
  kitcomponents=xlc.license-compute-13.1.3-0-rhels-7.2-ppc64le,xlc.rte-compute-13.
  1.3-0-rhels-7.2-ppc64le,xlc.compiler-compute-13.1.3-0-rhels-7.2-ppc64le,xlf.
  license-compute-15.1.3-0-rhels-7.2-ppc64le,xlf.rte-compute-15.1.3-0-rhels-7.2-
  ppc64le,xlf.compiler-compute-15.1.3-0-rhels-7.2-ppc64le,pperte-license-2.3.0.0-
  1547a-rhels-7.2-ppc64le,pperte-login-2.3.0.0-1547a-rhels-7.2-ppc64le,pperte-
  compute-2.3.0.0-1547a-rhels-7.2-ppc64le,min-pperte-compute-2.3.0.0-1547a-rhels-7.
  2-ppc64le,pperte-login-2.3.0.2-s002a-rhels-7.2-ppc64le,pperte-compute-2.3.0.2-
  s002a-rhels-7.2-ppc64le,min-pperte-compute-2.3.0.2-s002a-rhels-7.2-ppc64le,essl-
  license-5.4.0-0-rhels-7.2-ppc64le,essl-computenode-3264rte-5.4.0-0-rhels-7.2-
  ppc64le,essl-computenode-6464rte-5.4.0-0-rhels-7.2-ppc64le,essl-computenode-5.4.0-
  0-rhels-7.2-ppc64le,essl-loginnode-5.4.0-0-rhels-7.2-ppc64le,essl-computenode-
  3264rtecuda-5.4.0-0-rhels-7.2-ppc64le,ppedev.license-2.2.0-0-rhels-7.2-ppc64le,
  ppedev.login-2.2.0-0-rhels-7.2-ppc64le,ppedev.compute-2.2.0-0-rhels-7.2-ppc64le,
  pessel-license-5.2.0-0-rhels-7.2-ppc64le,pessel-loginnode-5.2.0-0-rhels-7.2-ppc64le,
  pessel-computenode-5.2.0-0-rhels-7.2-ppc64le,pessel-computenode-3264rtempich-5.2.0-
  0-rhels-7.2-ppc64le
  osarch=ppc64le
  osdistrname=rhels7.2-ppc64le
  osname=Linux
  osvers=rhels7.2
  otherpkgdir=/install/post/otherpkgs/rhels7.2/ppc64le
  otherpkglist=/install/osimages/rhels7.2-ppc64le-install-compute-kits/kits/KIT_
  DEPLOY_PARAMS.otherpkgs.pkglist,/install/osimages/rhels7.2-ppc64le-install-
  compute-kits/kits/KIT_COMPONENTS.otherpkgs.pkglist
  pkgdir=/install/rhels7.2/ppc64le,/install/cuda-7.5/ppc64le
  pkglist=/opt/xcat/share/xcat/install/rh/compute.rhels7.pkglist
  postbootscripts=KIT_pperte-login-2.3.0.0-1547a-rhels-7.2-ppc64le_pperte_postboot,
  KIT_pperte-compute-2.3.0.0-1547a-rhels-7.2-ppc64le_pperte_postboot,KIT_min-pperte-
  compute-2.3.0.0-1547a-rhels-7.2-ppc64le_pperte_postboot,KIT_pperte-login-2.3.0.2-
  s002a-rhels-7.2-ppc64le_pperte_postboot,KIT_pperte-compute-2.3.0.2-s002a-rhels-7.
  2-ppc64le_pperte_postboot,KIT_min-pperte-compute-2.3.0.2-s002a-rhels-7.2-ppc64le_
  pperte_postboot
  profile=compute
  provmethod=install
  template=/opt/xcat/share/xcat/install/rh/compute.rhels7.tpl
```

4. The osimage is now ready to deploy to the compute nodes.



## Product Specific Details

Refer to the following pages for product specific details and known issues.

## IBM XL Compilers

IBM provides XL compilers with advanced optimizing on IBM Power Systems running Linux. For more information, <http://www-03.ibm.com/software/products/en/xlcpp-linux>

## Partial Kits

The IBM XL compilers are dependencies for some of the HPC software products and is **not** available in xCAT Software Kit format.

To assist customers in creating a software kit for the IBM XL compilers, xCAT provides partial kits at: <https://xcat.org/files/kits/hpckits/>

## Creating Compiler Complete Kit

To use software kits that require compiler kit components, a compiler software kit must be available. The following example will outline the steps required to create the `xlc-12.1.0.8-151013-ppc64.tar.bz2` compiler software kit. Repeat the steps for each compiler kit required.

1. xCAT buildkit command requires `createrepo` is available on the machine:

```
which createrepo
```

If not available, use the default OS package manager to install. (`yum`, `zypper`, `apt-get`)

2. Obtain the IBM XL compilers rpms from the IBM Download site.

`xlc-12.1.0.8` is downloaded to `/tmp/kits/xlc-12.1.0.8`

```
# ls -l /tmp/kits/xlc-12.1.0.8/
vac.cmp-12.1.0.8-151013.ppc64.rpm
vac.lib-12.1.0.8-151013.ppc64.rpm
vac.lic-12.1.0.8-120323.ppc64.rpm
vacpp.cmp-12.1.0.8-151013.ppc64.rpm
vacpp.help.pdf-12.1.0.8-151013.ppc64.rpm
vacpp.lib-12.1.0.8-151013.ppc64.rpm
vacpp.man-12.1.0.8-151013.ppc64.rpm
vacpp.rte-12.1.0.8-151013.ppc64.rpm
vacpp.rte.lnk-12.1.0.8-151013.ppc64.rpm
vacpp.samples-12.1.0.8-151013.ppc64.rpm
xlc_compiler-12.1.0.8-151013.noarch.rpm
xlc_compute-12.1.0.8-151013.noarch.rpm
xlc_license-12.1.0.8-151013.noarch.rpm
xlc_rte-12.1.0.8-151013.noarch.rpm
xlmass.lib-7.1.0.8-151013.ppc64.rpm
xlsmp.lib-3.1.0.8-151013.ppc64.rpm
xlsmp.msg.rte-3.1.0.8-151013.ppc64.rpm
xlsmp.rte-3.1.0.8-151013.ppc64.rpm
```



3. Obtain the corresponding compiler partial kit from <https://xcat.org/files/kits/hpckits/>.<sup>1</sup>

**xlc-12.1.0.8-151013-ppc64.NEED\_PRODUCT\_PKGS.tar.bz2** is downloaded to /tmp/kits:

```
xlc-12.1.0.8-151013-ppc64.NEED_PRODUCT_PKGS.tar.bz2
```

4. Complete the partial kit by running the `buildkit addpkgs` command:

```
buildkit addpkgs xlc-12.1.0.8-151013-ppc64.NEED_PRODUCT_PKGS.tar.bz2 \
--pkgdir /tmp/kits/xlc-12.1.0.8
```

Sample output:

```
Extracting tar file /tmp/kits/xlc-12.1.0.8-151013-ppc64.NEED_PRODUCT_PKGS.tar.bz2.
↳Please wait.
Spawning worker 0 with 5 pkgs
Spawning worker 1 with 5 pkgs
Spawning worker 2 with 4 pkgs
Spawning worker 3 with 4 pkgs
Workers Finished
Saving Primary metadata
Saving file lists metadata
Saving other metadata
Generating sqlite DBs
Sqlite DBs complete
Creating tar file /tmp/kits/xlc-12.1.0.8-151013-ppc64.tar.bz2.
Kit tar file /tmp/kits/xlc-12.1.0.8-151013-ppc64.tar.bz2 successfully built.
```

5. The complete kit, /tmp/kits/xlc-12.1.0.8-151013-ppc64.tar.bz2 is ready to be used.

## Parallel Environment Runtime Edition (PE RTE)

xCAT software kits for PE RTE for Linux is available on:<sup>1</sup>

- PE RTE 1.3 1 and newer

### PE RTE and `mlnxofed_ib_install` Conflict

PPE requires the 32-bit version of `libibverbs`. The default behavior of the `mlnxofed_ib_install` postscript used to install the Mellanox OFED Infiniband (IB) driver is to remove any of the old IB related packages when installing. To bypass this behavior, set the variable `mlnxofed_options=--force` when running the `mlnxofed_ib_install` script.

<sup>1</sup> If the partial kit for the version needed does not exist on the download site, open an [issue](#) to the xcat development team.

<sup>1</sup> If using older releases of PE RTE, refer to [IBM HPC Stack in an xCAT Cluster](#)

## Install Multiple Versions

Beginning with **PE RTE 1.2.0.10**, the packages are designed to allow for multiple versions of PE RTE to coexist on the same machine.

The default behavior of xCAT software kits is to only allow one version of a `kitcomponent` to be associated with an xCAT osimage. When using `addkitcomp` to add a newer version of a kit component, xCAT will first remove the old version of the kit component before adding the new one.

To add multiple versions of PE RTE kit components to the same osimage, use the `-n` | `--noupgrade` option. For example, to add PE RTE 1.3.0.1 and PE RTE 1.3.0.2 to the `compute` osimage:

```
addkitcomp -i compute pperite_compute-1.3.0.1-0-rhels-6-x86_64
addkitcomp -i compute -n pperite_compute-1.3.0.2-0-rhels-6-x86_64
```

## POE hostlist

When running parallel jobs, POE requires the user pass it a host list file. xCAT can help to create this hostlist file by running the `nodeids` command against the desired node range and redirecting to a file.

```
nodeids compute > /tmp/hostlist
```

## Known Issues

- **[PE RTE 1.3.0.7]** - For developers creating the complete software kit. The `src rpm` is no longer required. It is recommended to create the new software kit for PE RTE 1.3.0.7 from scratch and not to use the older kits as a starting point.
- **[PE RTE 1.3.0.7]** - When upgrading `ppe_rte_man` in a diskless image, there may be errors reported during the `genimage` process. The new packages are actually upgraded, so the errors can be ignored with low risk.
- **[PE RTE 1.3.0.1 to 1.3.0.6]** - When uninstalling or upgrading `ppe_rte_man` in an diskless image, `genimage <osimage>` may fail and stop an an error. To workaround, simply rerun `genimage <osimage>` to finish the creation of the diskless image

## Parallel Environment Developer Edition (PE DE)

xCAT software kits for PE DE for Linux is available on:<sup>1</sup>

- PE DE 1.2.0.1 and newer (SystemX)
- PE DE 1.2.0.3 and newer (SystemP)

---

<sup>1</sup> If using an older release, refer to [IBM HPC Stack in an xCAT Cluster](#)

## Engineering and Scientific Subroutine Library (ESSL)

xCAT software kits for ESSL for Linux is available on:<sup>1</sup>

- ESSL 5.2.0.1 and newer

### Dependencies

- ESSL has a dependency on the XLC/XLF compilers

When adding the ESSL kit component to the osimage, ensure that the compiler kit component is already added to the osimage, or, use the `-a | --adddeps` option on `addkitcomp` to automatically assign the kit dependencies to the osimage.

To add the `essl-computenode` kit component to osimage `rhels7.2-ppc64le-install-compute`:

```
addkitcomp -a -i rhels7.2-ppc64le-install-compute \
  essl-computenode-3264rte-5.4.0-0-rhels-7.2-ppc64le
```

### Reference

Refer to ESSL installation guide for more information: [http://www.ibm.com/support/knowledgecenter/SSFHY8\\_5.4.0/com.ibm.cluster.essl.v5r4.essl300.doc/am5il\\_xcatinstall.htm](http://www.ibm.com/support/knowledgecenter/SSFHY8_5.4.0/com.ibm.cluster.essl.v5r4.essl300.doc/am5il_xcatinstall.htm)

## Parallel Engineering and Scientific Subroutine Library (PESSL)

xCAT software kits for PESSL for Linux is available on:<sup>1</sup>

- PESSL 4.2.0.0 and newer

### Dependencies

- PESSL has a dependency on the ESSL kit component

When adding PESSL kit component and want ESSL installed, ensure that the ESSL kit component is already added to the osimage, or, use the `-a | --adddeps` option on `addkitcomp` to automatically assign the kit dependencies to the osimage.

To add the `pessl-computenode` kit component to osimage `rhels7.2-ppc64le-install-compute`:

```
addkitcomp -a -i rhels7.2-ppc64le-install-compute \
  pessl-computenode-5.2.0-0-rhels-7.2-ppc64le
```

<sup>1</sup> If using an older release, refer to [IBM HPC Stack in an xCAT Cluster](#)

<sup>1</sup> If using an older release, refer to [IBM HPC Stack in an xCAT Cluster](#)

## Custom Kits - Creating Software Kits

### Introduction

#### Contents

A Software Kit is a tar file that contains the following:

**Kit Configuration File** — A file describing the contents of this kit and contains following information

- Kit name, version, description, supported OS distributions, license information, and deployment parameters
- Kit repository information including name, supported OS distributions, and supported architectures
- Kit component information including name, version, description, server roles, scripts, and other data

**Kit Repositories** — A directory for each operating system version this kit is supported in. Each directory contains all of the product software packages required for that environment along with repository metadata.

**Kit Components** — A product “meta package” built to require all of the product software dependencies and to automatically run installation and configuration scripts.

**Kit and Kit Component Files** — Scripts, deployment parameters, exclusion lists, and other files used to install and configure the kit components and product packages.

**Docs** [PCM only]<sup>1</sup> — Product documentation shipped as HTML files that can be displayed through the PCM GUI

**Plugins** [PCM only] — xCAT plugins that can be used for additional product configuration and customization during PCM image management and node management

#### Kit Components

Software Kits are deployed to xCAT managed nodes through the xCAT osimage deployment mechanism. The kit components are inserted into the attributes of the Linux osimage definition. The attributes that are modified are the following:

- `kitcomponents` - A list of the kitcomponents assigned to the OS image
- `serverrole` - The role of this OS image that must match one of the supported serverroles of a kitcomponent
- `otherpkglist` - Includes kitcomponent meta package names
- `postinstall` - Includes kitcomponent scripts to run during `genimage`
- `postbootscripts` - Includes kitcomponent scripts
- `exlist` - Exclude lists for diskless images
- `otherpkgdir` - Kit repositories are linked as subdirectories to this directory

Once the kit components are added to xCAT osimage definitions, administrators can use:

1. `standard` node deployment for installing the kit components during diskful OS provisioning
2. `genimage` command to create a diskless OS image installing the kit components for diskless OS provisioning
3. `updatenode` command to install the kit components on existing deployed nodes

The `kitcomponent` metadata defines the kit packages as dependency packages and the OS package manager (`yum`, `zypper`, `apt-get`) automatically installs the required packages during the xCAT `otherpkgs` install process.

---

<sup>1</sup> PCM is IBM Platform Cluster Manager

## Kit Framework

With time, the implementation of the xCAT Software Kit support may change.

In order to process a kit successfully, the kit must be compatible with the level of xCAT code that was used to build the kit. The xCAT kit commands and software kits contain the framework version and compatible supported versions.

To view the framework version, use the `-v` | `--version` option on *addkit*

```
# addkit -v
addkit - xCAT Version 2.11 (git commit 9ea36ca6163392bf9ab684830217f017193815be, built_
↪ Mon Nov 30 05:43:11 EST 2015)
    kitframework = 2
    compatible_frameworks = 0,1,2
```

If the commands in the xCAT installation is not compatible with the Software Kit obtained, update xCAT to a more recent release.

## Building Kits

### Requirements

The xCAT-buildkit rpm is required to create xCAT Software Kits. This rpm should be installed along with the rest of xCAT.

If the xCAT management node is not intended to be used to build the Software Kit, refer to the *Install Guide* to configure the xCAT repository on the target node and install xCAT-buildkit using one of the following commands:

- [RHEL]

```
yum clean metadata
yum install xCAT-buildkit
```

- [SLES]

```
zypper clean
zypper install xCAT-buildkit
```

- [UBUNTU]

```
apt-get clean
apt-get install xcat-buildkit
```

## Creating a New Kit

Use the *buildkit* command to create a kit template directory structure

```
buildkit create <kitbasename> [-l|--kitloc <kit location>]
```

## Kit Directory

The Kit directory location will be automatically populated with additional subdirectories and samples:

**buildkit.conf** - The sample Kit build configuration file.

**source\_packages** - This directory stores the source packages for Kit Packages and Non-Native Packages. The **buildkit** command will search these directories for source packages when building packages. This directory stores:

- RPM spec and tarballs. (A sample spec file is provided.)
- Source RPMs.
- Pre-built RPMs (contained in a subdirectory of source\_packages)
- Non-Native Packages

**scripts** - This directory stores the Kit Deployment Scripts. Samples are provided for each type of script.

**plugins** - This directory stores the Kit Plugins. Samples are provided for each type of plugin.

**docs** - This directory stores the Kit documentation files.

**other\_files**

- **kitdeployparams.lst**: Kit Deployment parameters file
- **exclude.lst**: File containing files/dirs to exclude in stateless image.

**build** - This directory stores files when the Kit is built.

- **kit\_repodir** - This directory stores the fully built Kit Package Repositories
- **<kitbasename>** - This directory stores the contents of the Kit tarfile before it is tar'ed up.

**<kitname>** - The kit tar file, partial kit name or complete kit tar file name (ex. kitname.tar.bz2)

## Kit Configuration File

The `buildkit.conf` file is a sample file that contains a description of all the supported attributes and indicates required or optional fields. The user needs to modify this file for the software kit to be built.<sup>1</sup>

**kit** — This stanza defines general information for the Kit. There must be exactly one kit stanza in a kit build file.

```
kit:
  basename=pperte
  description=Parallel Environment Runtime Edition
  version=1.3.0.6
  release=0
  ostype=Linux
  osarch=x86_64
  kitlicense=ILAN           <== the default kit license string is "EPL"
  kitdeployparams=pe.env    <== pe.env has to define in the other_files dir.
```

**kitrepo** — This stanza defines a Kit Package Repository. There must be at least one kitrepo stanza in a kit build file. If this kit need to support multiple OSes, user should create a separate repository for each OS. Also, no two repositories can be defined with the same OS name, major version, and arch.

---

<sup>1</sup> The latest version of the `buildkit.conf` file is located in the `/opt/xcat/share/xcat/kits/kit_template` directory.

```

kitrepo:
  kitrepoid=rhels6_x86_64
  osbasename=rhels
  osmajorversion=6
  osarch=x86_64

```

```

kitrepo:
  kitrepoid=sles11_x86_64
  osbasename=sles
  osmajorversion=11
  osarch=x86_64

```

minor version can be support following format:

```

osminorversion=2 <<-- minor version has to be exactly matched to 2
osminorversion=>=2 <<-- minor version can be 2 or greater than 2
osminorversion=<=2 <<-- minor version can be 2 or less than 2
osminorversion=>2 <<-- minor version has to be greater than 2
osminorversion=<2 <<-- minor version has to be less than 2

```

**kitcomponent** — This stanza defines one Kit Component. A kitcomponent definition is a way of specifying a subset of the product Kit that may be installed into an xCAT osimage. A kitcomponent may or may not be dependent on other kitcomponents. If user want to build a component which supports multiple OSes, need to create one kitcomponent stanza for each OS.

```

kitcomponent:
  basename=pperte_license
  description=PE RTE for compute nodes
  serverroles=compute
  # These packages must be shipped with the OS distro
  ospkgdeps=at,rsh,rsh-server,xinetd,sudo,libibverbs(x86-32),libibverbs(x86-64),
  ↪redhat-lsb
  kitrepoid=rhels6_x86_64
  kitpkgdeps=ppe_rte_license
kitcomponent:
  basename=pperte_compute
  description=PE RTE for compute nodes
  serverroles=compute
  kitrepoid=rhels6_x86_64
  kitcompdeps=pperte_license
  kitpkgdeps=pperte,pperteman,ppertesamples,src
  exlist=pe.exlist <=== the file needs to define in the other_files dir
  # All those post script need to define in the scripts dir
  postinstall=pperte_postinstall
  postupgrade=pperte_postinstall
  postbootscripts=pperte_postboot
kitcomponent:
  basename=pperte_license
  description=PE RTE for compute nodes
  serverroles=compute
  ospkgdeps=at,rsh-server,xinetd,sudo,libibverbs-32bit,libibverbs,insserv
  kitrepoid=sles11_x86_64
  kitpkgdeps=ppe_rte_license

```

**kitpackage** — This stanza defines Kit Package (ie. RPM). There can be zero or more kitpackage stanzas. For multiple package supports, need to

1. Define one kitpackage section per supported OS. or
2. Define one kitpacakge stanza which contains multiple kitrepoid lines. For the RPM packages, users need to responsible for creating an RPM spec file that can run on multiple OSes.

```
kitpackage:
    filename=pperte-*.x86_64.rpm
    kitrepoid=rhels6_x86_64,sles11_x86_64
kitpackage:
    filename=pperteman-*.x86_64.rpm
    kitrepoid=rhels6_x86_64,sles11_x86_64
kitpackage:
    filename=ppertesamples-*.x86_64.rpm
    kitrepoid=rhels6_x86_64,sles11_x86_64
kitpackage:
    filename=ppe_rte-*.x86_64.rpm
    kitrepoid=rhels6_x86_64,sles11_x86_64
kitpackage:
    filename=ppe_rte_man-*.x86_64.rpm
    kitrepoid=rhels6_x86_64,sles11_x86_64
kitpackage:
    filename=ppe_rte_samples-*.x86_64.rpm
    kitrepoid=rhels6_x86_64,sles11_x86_64
kitpackage:
    filename=src-*.i386.rpm
    kitrepoid=rhels6_x86_64,sles11_x86_64
#License rpm gets placed in all repos
kitpackage:
    filename=ppe_rte_license-*.x86_64.rpm
    kitrepoid=rhels6_x86_64,sles11_x86_64
```

## Partial vs. Complete Kits

A **complete** software kits includes all the product software and is ready to be consumed as is. A **partial** software kit is one that does not include all the product packages and requires the consumer to download the product software and complete the kit before it can be consumed.

To build partial kits, the `isexternalpkg=yes` needs to be set in the `kitpackage` stanza in the `buildkit.conf` file:

```
kitpackage:
    filename=foobar_runtime-*.x86_64.rpm
    kitrepoid=rhels6_x86_64
    isexternalpkg=yes
```



## Validating Kits

After modifying the `buildkit.conf` file and copying all the necessary files to the kit directories, use the `chkconfig` option on *buildkit* to validate the configuration file:

```
buildkit chkconfig
```

This command will verify all required fields defined in the `buildkit.conf`. If errors are found, fix the specified error and rerun the command until all fields are validated.

## Build Kit Repositories

After the `buildkit` configuration file is validated, run the `buildrepo` subcommand to build the Kit Package Repositories. The build server has to have same OS distributions, versions, or architectures with build kit repositories. User can copy the kit template directory to an appropriate server to build the repository then copy the results back the current system.

IBM HPC Products are using pre-built rpms. There are no OS/arch specific in the `kitcomponent` meta-package rpm and should be able to build all repositories on the same server.

To list the repos defined in the `buildkit.conf`:

```
buildkit listrepo
```

To build the repositories, specify a particular repository:

```
buildkit buildrepo <kit repo name>
```

or build all the repositories for this kit:

```
buildkit buildrep all
```

The repository would be built in `<Kit directory location>/build/kit_repodir/` subdirectory. If the Kit Package Repository is already fully built, then this command performs no operation. If the Kit Package Repository is not fully built, the command builds it as follows:

1. Create the Kit Package Repository directory `<Kit directory location>/build/kit_repodir/<Kit Pkg Repo>` .
2. Build the Component Meta-Packages associated with this Kit Package Repository. Create the packages under the Kit Package Repository directory
3. Build the Kit Packages associated with this Kit Package Repository. Create the packages under the Kit Package Repository directory
4. Build the repository meta-data for the Kit Package Repository. The repository meta-data is based on the OS native package format. For example, for RHEL, we build the YUM repository meta-data with the `createrepo` command.

## Build Tar File

After the Kit package repositories are built, run the `buildtar` subcommand in the Kit directory to build the final kit tarfile.

```
buildkit buildtar
```

The tar file will be built in the kit directory location. A complete kit will be named:

```
ex: kitname-1.0.0-x86_64.tar.bz2
```

A partial kit will have “NEED\_PRODUCT\_PKGS” string in its name:

```
ex: kitname-1.0.0-x86_64.NEED_PRODUCT_PKGS.tar.bz2
```

## Using Partial Kits with newer Software Versions

If the product packages are for a newer version or release than what specified in the partial kit tar file name, user may still be able to build a complete kit with the packages, assuming that the partial kit is compatible with those packages.

Note: Basically, the latest partial kit available online will work until there is a newer version available.

To build a complete kit with the new software, user can provide the new version and/or release of the software on the `buildkit` command line.

```
buildkit addpkgs <kitname.NEED_PRODUCT_PKGS.tar.bz2> --pkgdir <product package_
↳ directories> \
    --kitversion <new version> --kitrelease <new release>
```

For example, if the partial kit was created for a product version of 1.3.0.2 but wish to complete a new kit for product version 1.3.0.4 then can add “-k 1.3.0.4” to the `buildkit` command line.

## Completing a partial kit

Follow these steps to complete the kit build process for a partial kit.

1. copy the partial kit to a working directory
2. copy the product software packages to a convenient location or locations
3. cd to the working directory
4. Build the complete kit tarfile

```
buildkit addpkgs <kit.NEED_PRODUCT_PKGS.tar.bz2> --pkgdir <product package directories>
```

The complete kit tar file will be created in the working directory

## Using Kits

### Adding a Kit to xCAT

#### Adding a complete Kit to xCAT

A complete kit must be added to the xCAT management node and defined in the xCAT database before its kit components can be added to xCAT osimages or used to update diskful cluster nodes.

To add a kit run the following command:

```
addkit <complete kit tarfile>
```

The addkit command will expand the kit tarfile. The default location will be <site.installdir>/kits directory but an alternate location may be specified. (Where site.installdir is the value of the installdir attribute in the xCAT site definition.)

It will also add the kit to the xCAT database by creating xCAT object definitions for the kit as well as any kitrepo or kitcomponent definitions included in the kit.

Kits are added to the kit table in the xCAT database keyed by a combination of kit basename and version values. Therefore, user can add multiple kit definitions for the same product. For example, user could have one definition for release 1.2.0.0 and one for 1.3.0.0 of the product. This means that user will be able to add different versions of the kit components to different osimage definitions if desired.

#### Listing a kit

The xCAT kit object definition may be listed using the xCAT lsdef command.

```
lsdef -t kit -l <kit name>
```

The contents of the kit may be listed by using the lskit command.

```
lskit <kit name>
```

## Adding Kit Components

### Adding Kit Components to an OS Image Definition

In order to add a kitcomponent to an OS image definition, the kitcomponent must support the OS distro, version, architecture, serverrole for that OS image.

Some kitcomponents have dependencies on other kitcomponents. For example, a kit component may have a dependency on the product kit license component. Any kit components they may be required must also be defined in the xCAT database.

Note: A kit component in the latest product kit may have a dependency on a license kit component from an earlier kit version.

To check if a kitcomponent is valid for an existing OS image definition run the chkkitcomp command:

```
chkkitcomp -i <osimage> <kitcompname>
```

If the kit component is compatible then add the kitcomponent to the OS image definition using the addkitcomp command.

```
addkitcomp -a -i <osimage> <kitcompname>
```

When a kitcomponent is added to an OS image definition, the addkitcomp command will update several attributes in the xCAT database.

### Listing kit components

The xCAT kitcomponent object definition may be listed using the xCAT lsdef command.

```
lsdef -t kitcomponent -l <kit component name>
```

The contents of the kit component may be listed by using the lskitcomponent command.

```
lskitcomp <kit component name>
```

### Adding Multiple Versions of the Same Kit Component to an OS Image Definition

xCAT allows to have multiple versions/releases of a product software kit available in the cluster. Typically, different OS image definitions corresponding to the different versions/releases of a product software stack. However, in some instances, may need multiple versions/releases of the same product available within a single OS image. This is only feasible if the software product supports the install of multiple versions or releases of its product within an OS image.

Currently, it is not possible to install multiple versions of a product into an OS image using xCAT commands. xCAT uses yum on RedHat and zypper on SLES to install product rpms. These package managers do not provide an interface to install different versions of the same package, and will always force an upgrade of the package. We are investigating different ways to accomplish this function for future xCAT releases.

Some software products have designed their packaging to leave previous versions of the software installed in an OS image even when the product is upgraded. This is done by using different package names for each version/release, so that the package manager does not see the new version as an upgrade, but rather as a new package install. In this case, it is possible to use xCAT to install multiple versions of the product into the same image.

By default, when a newer version/release of a kitcomponent is added to an existing OS image definition, addkitcomp will automatically upgrade the kitcomponent by removing the old version first and then adding the new one. However, user can force both versions of the kitcomponent to be included in the OS image definition by specifying the full kitcomponent name and using the addkitcomp -n (-nougrade) flag with two separate command calls. For example, to include both myprod\_compute.1-0.1 and myprod\_compute.1-0.2 into an the compute osimage, you would run in this order:

```
addkitcomp -i compute myprod_compute.1-0.1
addkitcomp -i compute -n myprod_compute.1-0.2

lsdef -t osimage -o compute -i kitcomponents
  Object name:  compute
  kitcomponents=myprod_compute.1-0.1,myprod_compute.1-0.2
```

When building a diskless image for the first time, or when deploying a diskful node, xCAT will first install version 1-0.1 of myprod, and then in a separate yum or zypper call, xCAT will install version 1-0.2. The second install will either upgrade the product rpms or install the new versions of the rpms depending on how the product named the packages.

## Modifying Kit Deployment Parameters for an OS Image Definition

Some product software kits include kit deployment parameter files to set environment variables when the product packages are being installed in order to control some aspects of the install process. To determine if a kit includes such a file:

```
lsdef -t kit -o <kitname> -i kitdeployparams
```

If the kit does contain a deployment parameter file, the contents of the file will be included in the OS image definition when user add one of the kitcomponents to the image. User can view or change these values if need to change the install processing that they control for the software product:

```
addkitcomp -i <image> <kitcomponent name>
vi /install/osimages/<image>/kits/KIT_DEPLOY_PARAMS.otherpkgs.pkglist
```

NOTE: Be sure to know how changing any kit deployment parameters will impact the install of the product into the OS image. Many parameters include settings for automatic license acceptance and other controls to ensure proper unattended installs into a diskless image or remote installs into a diskful node. Changing these values will cause problems with genimage, updatenode, and other xCAT deployment commands.

## Completing the software update

### updating diskless images

For diskless OS images, run the genimage command to update the image with the new software. Example:

```
genimage <osimage>
```

Once the osimage has been updated you may follow the normal xCAT procedures for packing and deploying the image to your diskless nodes.

### installing diskful nodes

For new stateful deployments, the kitcomponent will be installed during the otherpkgs processing. Follow the xCAT procedures for your hardware type. Generally, it will be something like:

```
chdef <nodelist> provmethod=<osimage>
nodeset <nodelist> osimage
rpower <nodelist> reset
```

### updating diskful nodes

For existing active nodes, use the updatenode command to update the OS on those nodes. The updatenode command will use the osimage assigned to the node to determine the software to be updated. Once the osimage has been updated, make sure the correct image is assigned to the node and then run updatenode:

```
chdef <nodelist> provmethod=<osimage>
updatenode <nodelist>
```

## Removing Kit

### Removing Kit Components from an OS Image Definition

To remove a kit component from an OS image definition, first list the existing kitcomponents to get the name to remove:

```
lsdef -t osimage -o <image> -i kitcomponents
```

Then, use that name to remove it from the image definition:

```
rmkitcomp -i <image> <kitcomponent name>
```

Or, if know the basename of the kitcomponent, simply:

```
rmkitcomp -i <image> <kitcompent basename>
```

Note that this **ONLY** removes the kitcomponent from the image definition in the xCAT database, and it will **NOT** remove any product packages from the actual OS image. To set up for an uninstall of the kitcomponent from the diskless image or the stateful node, specify the uninstall option:

```
rmkitcomp -u -i <image> <kitcomponent>
```

The next time when run genimage for the diskless image, or updatenode to the fulldisk nodes, the software product will be un-installed.

### Removing a Kit from the xCAT Management Node

To remove a kit from xCAT, first make sure that no OS images are assigned any of the kitcomponents. To do this, run the following database queries:

```
lsdef -t kitcomponent -w 'kitname==<kitname>'
```

For each kitcomponent returned:

```
lsdef -t osimage -i kitcomponents -c | grep <kitcomponent>
```

If no osimages have been assigned any of the kitcomponents from this kit, can safely remove the kit by running:

```
rmkit <kitname>
```

## 1.5.12 Mixed Cluster

The concept of mixed cluster support in xCAT is managing cluster which contain different hardware architectures. Most commonly, this is usually a cluster consisting of both Power and x86 systems.

## Support Matrix

	RHEL ppc64 CN	SLES ppc64 CN	RHEL x86_64 CN	SLES x86_64 CN	Ubuntu x86_64 CN	RHEL ppc64le CN	SLES ppc64le CN	Ubuntu ppc64el CN
RHEL ppc64 MN/SN	yes	yes	yes <sup>1</sup>	yes <sup>1</sup>	yes <sup>1</sup>	yes	yes	yes
SLES ppc64 MN/SN	yes	yes	yes <sup>1</sup>	yes <sup>1</sup>	yes <sup>1</sup>	yes	yes	yes
RHEL x86_64 MN/SN	yes <sup>4</sup>	yes <sup>4</sup>	yes	yes	yes	yes	yes	yes
SLES x86_64 MN/SN	yes <sup>4</sup>	yes <sup>4</sup>	yes	yes	yes	yes	yes	yes
Ubuntu x86_64 MN/SN	yes <sup>5</sup>	yes <sup>5</sup>	yes	yes	yes	yes	yes	yes
RHEL ppc64le MN/SN	yes <sup>2</sup>	yes <sup>2</sup>	yes	yes	yes	yes	yes	yes
SLES ppc64le MN/SN	no	no	yes	yes	yes	yes	yes	yes
Ubuntu ppc64el MN/SN	yes <sup>3</sup>	yes <sup>3</sup>	yes	yes	yes	yes	yes	yes

### Notes:

- The support statements refers to hardware control, operating system (os) provisioning, and general purpose system management where we do not see any obvious problems with the indicated combination.
- For diskless mixed cluster support, the initial diskless image must be created on a node running the target operating system version and architecture. see [Building Stateless/Diskless Images](#) for more details.

<sup>1</sup> To manage x86\_64 servers from ppc64/ppc64le nodes, will need to install the packages **xnba-undi elilo-xcat** and **syslinux-xcat** manually on the management node. And manually run command “cp /opt/xcat/share/xcat/netboot/syslinux/pxelinux.0 /tftpboot/”

<sup>4</sup> If the compute nodes are DFM managed systems, will need the ppc64le DFM and ppc64le hardware server on the management node.

<sup>5</sup> Does not support DFM managed compute nodes, hardware control does not work.

<sup>2</sup> If the compute nodes are DFM managed systems, will need xCAT 2.9.1 or high versions and the ppc64le DFM and ppc64le hardware server on the management node.

<sup>3</sup> If the compute nodes are DFM managed systems, will need xCAT 2.10 or high versions and the ppc64le DFM and ppc64le hardware server on the management node.

## Power Management Node

### Provision x86 Diskful

In order to provision x86\_64 ipmi-based machines from Power-based xCAT management node, there are a few required xCAT dependency RPMs that must be installed:

- elilo-xcat
- xnba-undi
- syslinux-xcat

Install these RPMs using the following command:

```
yum install elilo-xcat xnba-undi syslinux-xcat
```

On the Power-based management node, obtain an x86\_64 operating system ISO and add it into the xCAT osimage table by using the copycds command:

```
copycds /tmp/RHEL-6.6-20140926.0-Server-x86_64-dvd1.iso
```

Create a node definition for the x86\_64 compute node, here is a sample:

```
lsdef -z c910f04x42
# <xCAT data object stanza file>

c910f04x42:
  objtype=node
  arch=x86_64
  bmc=10.4.42.254
  bmcpassword=PASSWORD
  bmcusername=USERID
  cons=ipmi
  groups=all
  initrd=xcat/osimage/rhels6.6-x86_64-install-compute/initrd.img
  installnic=mac
  kcmdline=quiet repo=http://!myipfn!:80/install/rhels6.6/x86_64 ks=http://!myipfn!:80/
  ↪install/autoinst/c910f04x42 ksdevice=34:40:b5:b9:c0:18 cmdline console=tty0
  ↪console=ttyS0,115200n8r
  kernel=xcat/osimage/rhels6.6-x86_64-install-compute/vmlinuz
  mac=34:40:b5:b9:c0:18
  mgt=ipmi
  netboot=xnba
  nodetype=osi
  os=rhels6.6
  profile=compute
  provmethod=rhels6.6-x86_64-install-compute
  serialflow=hard
  serialport=0
  serialspeed=115200
```

Verify the genesis packages:

- [RHEL/SLES]: `rpm -qa | grep -i genesis`
- [Ubuntu]: `dpkg -l | grep -i genesis`



If missing, install the packages `xCAT-genesis-base` and `xCAT-genesis-scripts` from `xcat-deps` repository and run `mknb <arch>` to create the genesis network boot root image.

Provision the node using the following commands:

```
# The following prepares the boot files in /install and /tftpboot
nodeset c910f04x42 osimage=rhels6.6-x86_64-install-compute

# Tells the BIOS to network boot on the next power on
rsetboot c910f04x42 net

# Reboots the node
rpower c910f04x42 boot
```

## Provision x86 Diskless

### Troubleshooting

**Error:** The following Error message comes out when running `nodeset`:

```
Error: Unable to find pxelinux.0 at /opt/xcat/share/xcat/netboot/syslinux/pxelinux.0
```

**Resolution:**

The `syslinux` network booting files are missing. Install the `syslinux-xcat` package provided in the `xcat-deps` repository:  
`yum -y install syslinux-xcat`

## x86 Management Node

### Building Stateless/Diskless Images

A **stateless**, or **diskless**, provisioned nodes is one where the operating system image is deployed and loaded into memory. The Operating System (OS) does not store its files directly onto persistent storage (i.e. hard disk drive, shared drive, usb, etc) and so subsequent rebooting of the machine results in loss of any state changes that happened while the machine was running.

To deploy stateless compute nodes, you must first create a stateless image. The “netboot” osimages created from `copycds` in the **osimage** table are sample osimage definitions that can be used for deploying stateless nodes.

In a homogeneous cluster, the management node is the same hardware architecture and running the same Operating System (OS) as the compute nodes, so `genimage` can directly be executed from the management node.

The issues arises in a heterogeneous cluster, where the management node is running a different level operating system *or* hardware architecture as the compute nodes in which to deploy the image. The `genimage` command that builds stateless images depends on various utilities provided by the base operating system and needs to be run on a node with the same hardware architecture and *major* Operating System release as the nodes that will be booted from the image.

## Same Operating System, Different Architecture

The following describes creating stateless images of the same Operating System, but different hardware architecture. The example will use the following nodes:

Management Node: xcatmn (ppc64)

Target node: n01 (x86\_64)

1. On xCAT management node, xcatmn, select the osimage you want to create from the list of osimage definitions. To list out the osimage definitions:

```
lsdef -t osimage
```

2. **optional:** Create a copy of the osimage definition that you want to modify.

To take the sample rhels6.3-x86\_64-netboot-compute osimage definition and create a copy called mycomputeimage, run the following command:

```
lsdef -t osimage -z rhels6.3-x86_64-netboot-compute | sed 's/^[^ ]\+:/\
↪mycomputeimage:/' | mkdef -z
```

3. To obtain the genimage command to execute on n01, execute the genimage command with the --dryrun option:

```
genimage --dryrun mycomputeimage
```

The result output will look similar to the following:

```
Generating image:
cd /opt/xcat/share/xcat/netboot/rh;
./genimage -a x86_64 -o rhels6.3 -p compute --permission 755 --srcdir /install/
↪rhels6.3/x86_64 --pkglist \
/opt/xcat/share/xcat/netboot/rh/compute.rhels6.x86_64.pkglist --otherpkgdir /
↪install/post/otherpkgs/rhels6.3/x86_64 --postinstall \
/opt/xcat/share/xcat/netboot/rh/compute.rhels6.x86_64.postinstall --rootimgdir /
↪install/netboot/rhels6.3/x86_64/compute mycomputeimage
```

4. Go to the target node, n01 and run the following:

1. mount the /install directory from the xCAT Management Node:

```
mkdir /install
mount -o soft xcatmn:/install /install
```

2. Copy the executable files from the /opt/xcat/share/xcat/netboot from the xCAT Management node to the target node:

```
mkdir -p /opt/xcat/share/xcat/
scp -r xcatmn:/opt/xcat/share/xcat/netboot /opt/xcat/share/xcat/
```

5. Execute the genimage command obtained from the --dryrun:

```
cd /opt/xcat/share/xcat/netboot/rh;
./genimage -a x86_64 -o rhels6.3 -p compute --permission 755 --srcdir /install/
↪rhels6.3/x86_64 --pkglist \
```

(continues on next page)

(continued from previous page)

```
/opt/xcat/share/xcat/netboot/rh/compute.rhels6.x86_64.pkglist --otherpkgdir /
→install/post/otherpkgs/rhels6.3/x86_64 --postinstall \
/opt/xcat/share/xcat/netboot/rh/compute.rhels6.x86_64.postinstall --rootimgdir /
→install/netboot/rhels6.3/x86_64/compute mycomputeimage
```

If problems creating the stateless image, provide a local directory for `--rootimgdir`:

```
mkdir -p /tmp/compute
```

Rerun `genimage`, replacing `--rootimgdir /tmp/compute`:

```
cd /opt/xcat/share/xcat/netboot/rh;
./genimage -a x86_64 -o rhels6.3 -p compute --permission 755 --srcdir /install/
→rhels6.3/x86_64 --pkglist \
/opt/xcat/share/xcat/netboot/rh/compute.rhels6.x86_64.pkglist --otherpkgdir /
→install/post/otherpkgs/rhels6.3/x86_64 --postinstall \
/opt/xcat/share/xcat/netboot/rh/compute.rhels6.x86_64.postinstall --rootimgdir /
→tmp/compute mycomputeimage
```

Then copy the contents from `/tmp/compute` to `/install/netboot/rhels6.3/compute`

6. Now return to the management node and execute `packimage` on the `osimage` and continue provisioning the node

```
packimage mycomputeimage
```

## 1.5.13 Networking

### Ethernet Switches

#### Configure Ethernet Switches

It is recommended that spanning tree be set in the switches to portfast or edge-port for faster boot performance. See the relevant switch documentation as to how to configure this item.

It is recommended that lldp protocol in the switches is enabled to collect the switch and port information for compute node during discovery process.

**Note:** this step is necessary if you want to use **xCAT**'s automatic switch-based discovery described in [switch-based discovery](#) for IPMI-controlled rack-mounted servers (Includes OpenPOWER server and x86\_64 server) and Flex chassis. If you have a small cluster and prefer to use the sequential discover method described in [Sequential-based discovery](#) or manually enter the MACs for the hardware, you can skip this section. Although you may want to still set up your switches for management so you can use **xCAT** tools to manage them, as described in [Switch Management](#).

**xCAT** will use the ethernet switches during node discovery to find out which switch port a particular MAC address is communicating over. This allows **xCAT** to match a random booting node with the proper node name in the database. To set up a switch, give it an IP address on its management port and enable basic **SNMP** functionality. (Typically, the **SNMP** agent in the switches is disabled by default.) The easiest method is to configure the switches to give the **SNMP** version 1 community string called "public" read access. This will allow **xCAT** to communicate to the switches without further customization. (**xCAT** will get the list of switches from the **switch** table.) If you want to use **SNMP** version 3 (e.g. for better security), see the example below. With **SNMP** V3 you also have to set the user/password and AuthProto (default is **md5**) in the switches table.

If for some reason you can't configure **SNMP** on your switches, you can use sequential discovery or the more manual method of entering the nodes' MACs into the database.

**SNMP V3 Configuration example:**

xCAT supports many switch types, such as **BNT** and **Cisco**. Here is an example of configuring **SNMP V3** on the **Cisco** switch 3750/3650:

1. First, user should switch to the configure mode by the following commands:

```
[root@x346n01 ~]# telnet xcat3750
Trying 192.168.0.234...
Connected to xcat3750.
Escape character is '^]'.
User Access Verification
Password:

xcat3750-1>enable
Password:

xcat3750-1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
xcat3750-1(config)#
```

2. Configure the **snmp-server** on the switch:

```
Switch(config)# access-list 10 permit 192.168.0.20 # 192.168.0.20 is the IP of MN
Switch(config)# snmp-server group xcatadmin v3 auth write v1default
Switch(config)# snmp-server community public RO 10
Switch(config)# snmp-server community private RW 10
Switch(config)# snmp-server enable traps
```

3. Configure the **snmp** user id (assuming a user/pw of xcat/passw0rd):

```
Switch(config)# snmp-server user xcat xcatadmin v3 auth SHA passw0rd access 10
```

4. Check the **snmp** communication to the switch :

```
On the MN: make sure the snmp rpms have been installed. If not, install them:

yum install net-snmp net-snmp-utils

Run the following command to check that the snmp communication has been setup.
↳ successfully (assuming the IP of the switch is 192.168.0.234):

snmpwalk -v 3 -u xcat -a SHA -A passw0rd -X cluster -l authnoPriv 192.168.0.234 .1.
↳ 3.6.1.2.1.2.2.1.2
```

Later on in this document, it will explain how to make sure the switch and switches tables are setup correctly.

## Switch Management

When managing Ethernet switches, the admin often logs into the switches one by one using SSH or Telnet and runs the switch commands. However, it becomes time consuming when there are a lot of switches in a cluster. In a very large cluster, the switches are often identical and the configurations are identical. It helps to configure and monitor them in parallel from a single command.

For managing Mellanox IB switches and Qlogic IB switches, see [Mellanox IB switches and Qlogic IB switches](#)

xCAT will not do a lot of switch management functions. Instead, it will configure the switch so that the admin can run remote command such as `xdsh` for it. Thus, the admin can use the `xdsh` to run proprietary switch commands remotely from the xCAT mn to enable **VLAN**, **bonding**, **SNMP** and others.

## Running Remote Commands in Parallel

You can use `xdsh` to run parallel commands on Ethernet switches. The following shows how to configure xCAT to run `xdsh` on the switches:

**Note:** For this to work, configure the switch to allow **ssh** or **telnet**. The procedure varies from switch to switch, consult the reference guides for your switch to find out how to do this.

Add the switch in xCAT DB. Refer to the “Discovering Switches” section if you want xCAT to discover and define the switches for you.

```
mkdef bntc125 groups=switch mgt=switch ip=10.4.25.1 nodetype=switch switchtype=BNT
```

Set the ssh or telnet username and password.

```
chdef bntc125 username=admin \
                password=password \
                protocol=ssh
or
chdef bntc125 username=admin \
                password=password \
                protocol=telnet
```

If there are a lot of switches **and** they have the same user name **and** password **for** ssh **or** telnet connection, you can put them **in** the passwd table keyed by **\*\*switch\*\***. You can use the comments attribute to describe it **is for** ssh to telnet. The blank means ssh. ::

```
#key,username,password,cryptmethod,authdomain,comments,disable
"system","root","cluster",,,,
"switch","admin","password",,,,
```

Run `xdsh` command

```
xdsh bntc125 --devicetype EthSwitch::BNT "enable;configure terminal;vlan 3;end;show vlan
↪"
```

**Note:** You can run multiple switch commands, each command is comma separated.

Also note that `--devicetype` is used here. xCAT supports the following switch types out of the box:

```
* BNT
* Cisco
* Juniper
* Mellanox (for IB and Ethernet switches)
```

If you have different type of switches, you can either use the general flag

“--devicetype EthSwitch” or add your own switch types. (See the following section).

Here is what result will look like:

```
bntc125: start SSH session...
bntc125: RS G8000>enable
bntc125: Enable privilege granted.
bntc125: configure terminal
bntc125: Enter configuration commands, one per line. End with Ctrl/Z.
bntc125: vlan 3
bntc125: end
bntc125: show vlan
bntc125: VLAN                               Name                               Status                               Ports
bntc125: ----                               -
bntc125: 1      Default VLAN                ena      45-XGE4
bntc125: 3      VLAN 3                      dis      empty
bntc125: 101    xcatpriv101                 ena      24-44
bntc125: 2047   9.114.34.0-pub              ena      1-23 44
```

You can run xdsh against more than one switches at a time, just like running xdsh against nodes.

Use xcoll to summarize the result. For example:

```
xdsh bntc1,bntc2 --devicetype EthSwitch::BNT "show access-control" |xcoll
```

The output looks like this:

```
=====
bntc1,bntc2
=====
start Telnet session...
terminal-length 0
show access-control
Current access control configuration:
  No ACLs configured.
  No IPv6 ACL configured.
  No ACL group configured.
  No VMAP configured.
```

## Add New Switch Types

For any new switch types that's not supported by xCAT yet, you can use the general **--device EthSwitch** flag with xds command.

```
xds <switch_names> --devicetype EthSwitch "cmd1;cmd2..."
```

The only problem is that the page break is not handled well when the command output is long. To remove the page break, you can add a switch command that sets the terminal length to 0 before all other commands.

```
xds <switch_names> --devicetype EthSwitch "command-to-set-term-length-to-0;cmd1;cmd2..."
```

where `command-to-set-term-length-to-0` is the command to set the terminal length to 0 so that the output does **not** have page breaks.

You can add this command to the configuration file to avoid specifying it for each xds by creating a new switch type. Here is what you do:

```
cp /opt/xcat/share/xcat/devicetype/EthSwitch/Cisco/config \
/var/opt/xcat/EthSwitch/XXX/config
```

where XXX is the name of the new switch type. You can give it any name. Then add the command for set terminal length to 0 to the "pre-command" line. The new configuration file will look like this:

```
# cat /var/opt/xcat/EthSwitch/XXX/config
[main]
ssh-setup-command=
[xds]
pre-command=command-to-set-term-length-to-0;
post-command=NULL
```

For BNT switches, the `command-to-set-term-length-to-0` is `terminal-length 0`.

Make sure to add a semi-colon at the end of the "pre-command" line.

Then you can run the xds like this:

```
xds <switch_names> --devicetype EthSwitch::XXX "cmd1;cmd2..."
```

## Open Network Install Environment Switches

The Open Network Install Environment, or "ONIE"<sup>1</sup>, is an open source project defining an **install environment** for bare metal switches. This environment allows choice for the end users when selecting a network operating system to install onto these bare metal switches.

<sup>1</sup> Open Network Install Environment: Created by Cumulus Networks, Inc. in 2012, the Open Network Install Environment (ONIE) Project is a small operating system, pre-installed as firmware on bare metal network switches, that provides an environment for automated operating system provisioning.

## Cumulus Linux OS

This documentation will focus on installing the Cumulus Network Operating System (<https://www.cumulusnetworks.com/>) onto a “white-box” Edgecore switch but the same concepts should apply to other ONIE enabled switches using other network operating systems.

### Preparation

Prepare the Cumulus Linux files on the xCAT Management Node.

1. Obtain a valid Cumulus Linux License and download the Cumulus Linux OS installer.
2. Copy the above files into a location under the xCAT /install directory.

```
# Create a directory to hold the cumulus linux files
mkdir -p /install/custom/sw_os/cumulus/

# copy the license file
cp licensefile.txt /install/custom/sw_os/cumulus/

# copy the installer
cp cumulus-linux-3.1.0-bcm-armel.bin /install/custom/sw_os/cumulus/
```

### Cumulus osimage

xCAT can able to create a cumulus osimage definition via copycds command. copycds will copy cumulus installer to a destination directory, and create several relevant osimage definitions. **cumulus<release>-<arch>** is the default osimage name.

```
#run copycds command
# copycds cumulus-linux-3.5.2-bcm-armel.bin
```

The pkgdir attribute will contain full path of cumulus installer as **/install/cumulus<release>/<arch>/<installer>**.

```
# lsdef -t osimage cumulus3.5.2-armel
Object name: cumulus3.5.2-armel
  description=Cumulus Linux
  imagetype=linux
  osarch=armel
  osname=cumulus
  osvers=cumulus3.5.2
  pkgdir=/install/cumulus3.5.2/armel/cumulus-linux-3.5.2-bcm-armel.bin
  provmethod=install
```



## Installation and Configuration

### Cumulus OS Installtion

**Important:** The following assumes that the physical switches have power and have obtained a DHCP IP address from the xCAT open range.

xCAT provides support for detecting and installing the Cumulus Linux OS into ONIE enabled switches by utilizing DHCP to detect “**onie\_vendor**” from the `vendor-class-identifier` string and then send it the Cumulus Linux OS installer.

1. Create a pre-defined switch definition for the ONIE switch using the `onieswitch` template.

The mac address of the switch management port is required for xCAT to configure the DHCP information and send over the OS to install on the switch.

#### Small Clusters

If you know the mac address of the management port on the switch, create the pre-defined switch definition providing the mac address.

```
mkdef frame01sw1 --template onieswitch arch=armv71 \
    ip=192.168.1.1 mac="aa:bb:cc:dd:ee:ff"
```

#### Large Clusters

xCAT’s `switchdiscover` command can be used to discover the mac address and fill in the predefined switch definitions based on the switch/switchport mapping.

1. Define all the switch objects providing the switch/switchport mapping:

```
mkdef frame01sw1 --template onieswitch arch=armv71 \
    ip=192.168.1.1 switch=coresw1 switchport=1
mkdef frame02sw1 --template onieswitch arch=armv71 \
    ip=192.168.2.1 switch=coresw1 switchport=2
mkdef frame03sw1 --template onieswitch arch=armv71 \
    ip=192.168.3.1 switch=coresw1 switchport=3
mkdef frame04sw1 --template onieswitch arch=armv71 \
    ip=192.168.4.1 switch=coresw1 switchport=4
...
```

2. Leverage `switchdiscover` over the DHCP range to automatically detect the MAC address and write them into the predefined switches above.

```
switchdiscover --range <IP range>
```

2. Run the `nodeset` command to set the `provmethod` attribute of the target switch(es) to the Cumulus Linux install image and prepare the DHCP/BOOTP lease information for the switch:

```
# nodeset frame01sw1 osimage=cumulus3.5.2-armel
# lsdef frame01sw1
Object name: frame01sw1
arch=armv71
groups=switch,edge_switch
ip=172.21.208.03
```

(continues on next page)

(continued from previous page)

```
mac=8C:EA:1B:E8:78:C0
mgt=switch
netboot=onie
nodetype=switch
postbootscripts=otherpkgs
postscripts=syslog,remoteshell
provmethod=cumulus3.5.2-armel
switch=mid08
switchport=3
switchtype=onie
usercomment=Edgecore Networks Switch
```

nodeset will execute the `makedhcp` command and kick off the network install of the ONIE enabled switch. If there is no OS pre-loaded on the switch, the switch continues to send a DHCPREQUEST out to the network. After `makedhcp` is run against the switch, an entry is added to the leases file that will respond to the request with the Cumulus Linux installer file.

```
host frame1sw1 {
    dynamic;
    hardware ethernet 8c:ea:1b:12:ca:40;
    fixed-address 192.168.3.200;
    supersede server.ddns-hostname = "frame1sw1";
    supersede host-name = "frame1sw1";
    if substring (option vendor-class-identifier, 0, 11) = "onie_vendor" {
        supersede www-server = "http://192.168.27.1/install/cumulus3.5.2/armel/
        ↪cumulus-linux-3.5.2-bcm-armel.bin";
    }
}
```

*Typical installation time is around 1 hour*

## Configure xCAT Remote Commands

After Cumulus Linux OS is installed, a default user `cumulus` will be created with default password: `CumulusLinux!`.

To ease in the management of the switch, xCAT provides a script to help configure password-less ssh as the `root` user. This script sends over the xCAT ssh keys so that the xCAT remote commands (`xdsh`, `xdcp`, etc) can be run against the ONIE switches.

Execute the following to sync the xCAT keys to the switch:

```
rspconfig frame01sw1 sshcfg
```

Validate the ssh keys are correctly configured by running a `xdsh` command:

```
xdsh frame01sw1 uptime
```

## Activate the License

After Cumulus Linux OS is installed onto the ONIE switch, only the serial port console and the management ethernet port is enabled. To activate the rest of the switch ports, the license file needs to be installed onto the switch.

1. Copy the license file to the switch:

```
xdcp frame01sw1 /install/custom/sw_os/cumulus/licensefile.txt /root/
```

2. Activate the license:

```
xdsh frame01sw1 "/usr/cumulus/bin/cl-license -i /root/licensefile.txt"
```

3. Verify that the license file is successfully installed:

```
xdsh frame01sw1 /usr/cumulus/bin/cl-license
```

Output should be similar to: frame01sw1 xxx@xx.com|xxxxxxxxxxxxxxxxxx

4. Reboot the switch to apply the license file:

```
xdsh frame01sw1 reboot
```

## Enable SNMP

In order to utilize `xcatprobe switch_macmap`, snmp needs to be enabled. To enable, run the `enablesnmp` postscript on the switch:

```
updatenode frame01sw1 -P enablesnmp
```

To configuring SNMPv3 after enable snmp, set user, authentication and/or encryption for the switches:

```
chdef frame01sw1 snmpauth=sha snmppassword=xcatpassw0rd snmpprivacy=DES_
↪ snmpusername=xcatadmin
```

then execute the `configonie` command to add the snmp user for the switch:

```
/opt/xcat/share/xcat/scripts/configonie --switches frame01sw1 --snmp
```

To verify the SNMPv3 configuration, run `xcatprobe switch_macmap` command, will show following results:

```
#xcatprobe switch_macmap frame01sw1 -V
<INFO>frame1sw1: Attempting to refresh switch information...
<INFO>frame1sw1: Generate SNMP session with parameter:
    'UseNumeric' => '1'
    'SecName' => 'xcatadmin'
    'AuthPass' => 'xcatpassw0rd'
    'Version' => '3'
    'PrivProto' => 'DES'
    'DestHost' => '172.21.253.102'
    'SecLevel' => 'authPriv'
    'AuthProto' => 'SHA'
    'PrivPass' => 'xcatpassw0rd'
<INFO>frame1sw1: SNMP Session query OID:".1.3.6.1.2.1.31.1.1.1.1"
```

(continues on next page)

(continued from previous page)

```
<INFO>frame1sw1: SNMP Session get data for OID:".1.3.6.1.2.1.31.1.1.1.1":
    '1' => 'lo'
    '2' => 'eth0'
    '3' => 'swp1'
    '4' => 'swp2'
    '5' => 'swp3'
.....more output.....
```

## Switch Management

### Sync File support

xCAT supports synchronize of configuration files for cumulus switches.

1. Use instructions in doc: *The synclist file* to set up syncfile.
2. Add syncfile to cumulus osimage.

```
# chdef -t osimage cumulus3.5.2-armel synclists=/tmp/synclists
1 object definitions have been created or modified.
```

3. run updatenode to sync the files to cumulus switches.

```
# updatenode mid08tor03 -F
File synchronization has completed for nodes: "mid08tor03"
```

## Switch Port and VLAN Configuration

xCAT places the front-panel port configuration in `/etc/network/interfaces.d/xCAT.intf`.

The `configinterface` postscript can be used to pull switch interface configuration from the xCAT Management Node (MN) to the switch. Place the switch specific configuration files in the following directory on the MN: `/install/custom/sw_os/cumulus/interface/`.

xCAT will look for files in the above directory in the following order:

1. file name that matches the switch hostname
2. file name that matches the switch group name
3. file name that has the word 'default'

---

**Note:** If the postscript cannot find a configuration file on the MN, it will set all ports on the switch to be part of VLAN 1.

---

Execute the script using the following command:

```
updatenode <switch> -P configinterface
```

## Re-install OS

There may be occasions where a re-install of the Cumulus Linux OS is required. The following commands can be used to invoke the install:

---

**Important:** This assumes that the Cumulus Linux files are on the xCAT MN in the correct place.

---

- **Using xCAT**, `xdsh` can invoke the reinstall of the OS:

```
# to clear out all the previous configuration, use the -k option (optional)
xdsh <switch> "/usr/cumulus/bin/onie-select -k

# to invoke the reinstall of the OS
xdsh <switch> "/usr/cumulus/bin/onie-select -i -f;reboot"
```

- **Manually**, log into the switch and run the following commands:

```
sudo onie-select -i
sudo reboot
```

## Cumulus OS Upgrade

The Cumulus OS on the ONIE switches can be upgraded using one of the following methods:

### Full Install

Perform a full install from the `.bin` file of the new Cumulus Linux OS version, using ONIE.

---

**Important:** Make sure you back up all your data and configuration files as the binary install will erase all previous configuration.

---

1. Place the binary image under `/install` on the xCAT MN node.

In this example, IP=172.21.253.37 is the IP on the Management Node.

```
mkdir -p /install/onie/
cp cumulus-linux-3.4.1.bin /install/onie/
```

2. Invoke the upgrade on the switches using `xdsh`:

```
xdsh switch1 "/usr/cumulus/bin/onie-install -a -f -i \
http://172.21.253.37/install/onie/cumulus-linux-3.4.1.bin && reboot"
```

**Attention:** The full upgrade process may run 30 minutes or longer.

3. After upgrading, the license should be installed, see *Activate the License* for details.
4. Restore your data and configuration files on the switch.

## Update Changed Packages

This is the preferred method for upgrading the switch OS for incremental OS updates.

## Create Local Mirror

If the switches do not have access to the public Internet, you can create a local mirror of the Cumulus Linux repo.

1. Create a local mirror on the Management Node:

```
mkdir -p /install/mirror/cumulus
cd /install/mirror/cumulus
wget -m --no-parent http://repo3.cumulusnetworks.com/repo/
```

2. Create a `sources.list` file to point to the local repo on the Management node. In this example, IP=172.21.253.37 is the IP on the Management Node.

```
# cat /tmp/sources.list
deb      http://172.21.253.37/install/mirror/cumulus/repo3.cumulusnetworks.com/repo_
↳CumulusLinux-3 cumulus upstream
deb-src  http://172.21.253.37/install/mirror/cumulus/repo3.cumulusnetworks.com/repo_
↳CumulusLinux-3 cumulus upstream

deb      http://172.21.253.37/install/mirror/cumulus/repo3.cumulusnetworks.com/repo_
↳CumulusLinux-3-security-updates cumulus upstream
deb-src  http://172.21.253.37/install/mirror/cumulus/repo3.cumulusnetworks.com/repo_
↳CumulusLinux-3-security-updates cumulus upstream

deb      http://172.21.253.37/install/mirror/cumulus/repo3.cumulusnetworks.com/repo_
↳CumulusLinux-3-updates cumulus upstream
deb-src  http://172.21.253.37/install/mirror/cumulus/repo3.cumulusnetworks.com/repo_
↳CumulusLinux-3-updates cumulus upstream
```

3. Distribute the `sources.list` file to your switches using `xdcp`.

```
xdcp switch1 /tmp/sources.list /etc/apt/sources.list
```

## Invoke the Update

1. Use xCAT `xdsh` to invoke the update:

```
#
# A reboot may be needed after the upgrade
#
xdsh switch1 'apt-get update && apt-get upgrade && reboot'
```

2. Check in `/etc/os-release` file to verify that the OS has been upgraded.

## Setup ONIE switches with ZTP in large cluster

Zero Touch Provisioning (ZTP) is a feature shipped in many network devices to enable them to be quickly deployed in large-scale environments. In Cumulus OS on ONIE switches with ZTP enabled, the URL of an user provided script can be specified in the DHCP response for the DHCP request triggered by one of the following events:

- Booting the switch
- Plugging a cable into or unplugging it from the eth0 port
- Disconnecting then reconnecting the switch's power cord.

the script will be then downloaded and executed on the network device.

Leveraging the ZTP mechanism, xCAT provides the capability to setup ONIE switches from white-box without touching anything, including Cumulus OS installation, discovery and configuration. Please follow the steps below to setup ONIE switches in the cluster:

1. Ensure that xCAT is configured with an DHCP open range to detect when new switches request DHCP IPs

- (1). Make sure the network in which the management interface of the ONIE switches are connected has been defined in `networks` table

```
# tabdump networks
#netname,net,mask,mgtifname,gateway,dhcpserver,tftpserver,nameservers,ntpserver,
↪logserver,dynamicrange,staticrange,staticrangeincrement,nodehostname,ddnsdomain,
↪vlanid,domain,mtu,comments,disable
"172.21.0.0-255.255.0.0","172.21.0.0","255.255.0.0","enP3p3s0d1","<xcatmaster>",
↪"172.21.253.27","172.21.253.27",,,,"172.21.253.100-172.21.253.200",,,,,,,,,,
```

- (2). Prepare the DHCP configuration for ONIE switches setup

Add the management node's NIC facing the ONIE switches' management interface to the `site`.  
`dhcpiinterfaces`

```
chdef -t site -p dhcpiinterfaces=enP3p3s0d1
```

Add dynamic range for the temporary IP addresses used in the OS provision and discovery of ONIE switches

```
chdef -t network 172.21.0.0-255.255.0.0 dynamicrange="172.21.253.100-172.21.
↪253.200"
```

Update DHCP configuration file

```
makedhcp -n
```

2. Predefine ONIE switches according to the network plan

```
mkdef mid05tor10 --template onieswitch ip=172.21.205.10 switch=mgmtsw01
↪switchport=10
```

`ip` is the IP address of the management interface of the ONIE switch

`switch` is the core switch to which the management interface of ONIE switch is connected.

`switchport` is the port on the core switch to which the management interface of ONIE switch is connected.

3. Add the predefined switches into `/etc/hosts`

```
makehosts mid05tor10
```

4. [If the Cumulus OS have been installed on the ONIE switches, please skip this step] Prepare the Cumulus installation image, `/install/onie/onie-installer` is the hard-coded path of the Cumulus installation image, or the link to the Cumulus installation image on the management node

```
mkdir -p /install/onie/  
cp /install/custom/sw_os/cumulus/cumulus-linux-3.1.0-bcm-armel.bin /install/onie/  
ln -s /install/onie/cumulus-linux-3.1.0-bcm-armel.bin /install/onie/onie-installer
```

5. Plug the ONIE switches into the cluster according to the network plan and power on them.

For the white-box ONIE switches, the Cumulus OS will be installed, then the switches will be discovered and configured automatically, the whole process will take about 1 hour.

For the ONIE switches already with Cumulus OS installed, please make sure the ZTP have been enabled and none of the following manual configuration have been made:

- Password changes
- Users and groups changes
- Packages changes
- Interfaces changes
- The presence of an installed license

Otherwise, please run `ztp -R` on the switches to reset the ZTP state before switch boot up for setup. The whole setup process will take about 1-2 minutes.

6. The switch definition in xCAT will be updated when the switch is configured

```
# lsdef mid05tor10  
Object name: mid05tor10  
arch=armv7l  
groups=switch  
ip=172.21.205.10  
mac=xx:xx:xx:xx:xx:xx  
mgt=switch  
netboot=onie  
nodetype=switch  
postbootscripts=otherpkgs  
postscripts=syslog,remoteshell,syncfiles  
serial=11S01FT690YA50YD73EACH  
status=configured  
statustime=06-22-2017 23:14:14  
supportedarchs=armv7l  
switch=mgmtsw01  
switchport=10  
switchtype=Edgecore Networks Switch
```

`status=configured` indicates that the switch has been discovered and configured.



## Switch Discover

### Discovering Switches

Use switchdiscover command to discover the switches that are attached to the neighboring subnets on xCAT management node.

```
switchdiscover [noderange|--range ip_ranges] [-s scan_methods] [-r|-x|-z] [-w]
```

where the scan\_methods can be **nmap**, **snmp**, or **\*\*lldp**. The default is **nmap**. (**nmap** comes from most os distribution.)

To discover switches over the IP range 10.4.25.0/24 and 192.168.0.0/24, use the following command:

```
# switchdiscover --range 10.4.25.0/24,192.168.0.0/24
Discovering switches using nmap...
ip                name                vendor                mac
-----
192.168.0.131     switch_192_168_0_131  Mellanox Technologie  00:02:C9:AA:00:53
10.4.25.1         switch_10_4_25_1      Juniper networks      2C:6B:F5:00:11:22
```

If -w flag is specified, the command will write the discovered switches into xCAT databases. If the command above was executed with -w the following switch objects would be created:

```
# lsdef switch_name
Object name: switch_name
groups=switch
ip=switch_ip
mgt=switch
nodetype=switch
switchtype=switch_vendor
```

The **Ip** address is stored in the hosts table. You can run the following command to add the IP addresses in the **/etc/hosts**

```
makehosts
```

The discovery process works with the following four kind of switches:

```
Mellanox (IB and Ethernet switches)
Cisco
BNT
Juniper
```

The switchdiscover command can display the output in xml format, stanza format and normal list format. See the man pages for this command for details.

## Switch-based Switch Discovery

Currently, xCAT supports switch based hardware discovery, the servers are identified through the switches and switch ports they are directly connected to. A similar method can be used to discover switches using switch-based discovery within the user defined dynamic IP range.

### Pre-requirement

In order to do switch-based switch discovery, the admin

1. Needs to manually setup and configure core-switch, SNMP v3 needs to be enabled in order for xCAT access to it. **username** and **userpassword** attributes are for the remote login. It can be for **ssh** or **telnet**. If it is for **telnet**, set protocol to “telnet”. If the **username** is blank, the **username** and **password** will be retrieved from the passwd table with “switch” as the key. SNMP attributes will be used for SNMPv3 communication. **nodetype** has to be set to “switch” to differentiate between switch-based node discovery or switch-based switch discovery. Refer to switches table attributes. Example of core-switch definition:

```
lsdef switch-10-5-23-1
  Object name: switch-10-5-23-1
  groups=switch
  ip=10.5.23.1
  mac=ab:cd:ef:gh:dc
  mgt=switch
  nodetype=switch
  password=admin
  postbootscripts=otherpkgs
  postscripts=syslog,remoteshell,syncfiles
  protocol=telnet
  snmpauth=sha
  snmppassword=userpassword
  snmpusername=snmpadmin
  snmpversion=3
  switchtype=BNT
  usercomment=IBM
  username=root
```

2. Predefine all top-rack switches which connect to core-switch. The attribute **ip** is static IP address for the switch. When **switchdiscover --setup** command is issued, this IP address will replace dhcp IP address on the switch. **nodetype=switch** needs to be set to differentiate between switch-based node discovery or switch-based switch discovery during discovery process. the attribute **switch** is hostname of core-switch and **switchport** is the port number in the core-switch that top-rack switch is connected to.

```
lsdef switch-192-168-5-22
  objtype=node
  groups=switch
  ip=192.168.5.22
  mgt=switch
  nodetype=switch
  switch=switch-10-5-23-1
  switchport=45
  switchtype=BNT
```

3. Add switches to /etc/hosts for hostname lookup and xdsh command.

```
makehosts switch-192-168-5-23
makehosts switch-192-168-5-22
```

4. Setup Dynamic IP range in network table for discovered switches to use.

```
# tabdump networks
#netname,net,mask,mgtifname,gateway,dhcpserver,tftpserver,nameservers,ntpservers,
↪logservers,dynamicrange,staticrange,staticrangeincrement,nodehostname,ddnsdomain,
↪vlanid,domain,mtu,comments,disable
"192_168_0_0-255_255_0_0","192.168.0.0","255.255.0.0","enP4p1s0f2","<xcatmaster>",,
↪"192.168.3.29",,,,"192.168.5.150-192.168.5.170",,,,,,,,,,
```

dhcp should be restarted after setting up dynamic IP range.

## Discover Switches

**Note:** Only BNT and Mellanox switches are supported for switch-based switch discovery

xCAT can automatically discover switches that are connected to the defined subnets from the management node using the `switchdiscover` command.

For switch-based switch discovery, use the `--setup` option:

```
switchdiscover [noderange|--range ip_ranges][-s scan_methods] [--setup]
```

The `--setup` option will perform the following steps:

1. Scan the IP range using `snmp` or `nmap` to find all switches that respond. The available switches will be stored into the switch hash table with `hostname`, `switchtype`, `mac`, and vendor information.
2. Based on MAC address for each switch defined in the hash table, call `find_mac` subroutine. The `find_mac` subroutine will go through the switch and switch ports and find matched mac address.
  - If discovered switch didn't find any predefined switch matches, it will log the message `NO predefined switch matched`.
  - If discovered switch matched with one of pre-defined switch, it will update the predefined switch with

```
otherinterface=x.x.x.x (discovered ip)
state=matched
switchtype=type of switch
usercomment=vendor information
```

3. If the switches are matched, the `switchdiscover` command will execute the following scripts to configure static IP address, hostname, and enable the `snmpv3`.
  - `/opt/xcat/share/xcat/scripts/configBNT`
  - `/opt/xcat/share/xcat/scripts/configMellanox`
4. After discovery process the predefined node attribute in the xCAT database will be updated.

```
lsdef switch-192-168-5-22
groups=switch
ip=192.168.5.22
```

(continues on next page)

(continued from previous page)

```
mac=a8:97:dc:02:92:00
mgt=switch
nodetype=switch
password=admin
postbootscripts=otherpkgs
postscripts=syslog,remoteshell,syncfiles
protocol=telnet
snmpauth=sha
snmppassword=xcatadminpasswd@snmp
snmpusername=xcatadmin
snmpversion=3
status=hostname_configured
statustime=08-31-2016 15:35:49
supportedarchs=ppc64
switch=switch-10-5-23-1
switchport=45
switchtype=BNT
usercomment=IBM Networking Operating System RackSwitch G8052
username=root
```

**Tip:** Refer to *Discovering Switches* for more info.

## Configure switches

The **switchdiscover** command with `--setup` flag will set up switches with static IP address, change the hostname from predefined switches and enable snmpv3 configuration. For other switches configuration, refer to *Configure Ethernet Switches* and *IB Switch Configuration*.

These two config files are located in the `/opt/xcat/share/xcat/scripts` directory. The **switchdiscover** process will call the config files with `--all` option. User can call these scripts to setup one of options manually.

1. **configBNT** is for configure BNT switches.

```
./configBNT --help
Usage:
configBNT [-?|-h|--help]
configBNT [--switches switchnames] [--all]
configBNT [--switches switchnames] [--ip]
configBNT [--switches switchnames] [--name ]
configBNT [--switches switchnames] [--snmp] [--user snmp_user] [--password snmp_
password] [--group snmp_group]
configBNT [--switches switchnames] [--port port] [--vlan vlan]
```

2. **configMellanox** is for configuring Mellanox switch. The script will configure ntp service on the switch with xCAT MN and will use `rspconfig` command to
  - enable ssh
  - enable snmp function on the switch
  - enable the snmp trap
  - set logging destination to xCAT MN

```
./configMellanoX --help
Usage:
  configMellanoX [-?|-h|--help]
  configMellanoX [--switches switchnames] [--all]
  configMellanoX [--switches switchnames] [--ip]
  configMellanoX [--switches switchnames] [--name]
  configMellanoX [--switches switchnames] [--config]
```

## Switch Status

During the switch-based switch discovery process, there are four states displayed. User may only see **switch\_configured** status on node definition if discovery process successfully finished.

**Matched** — Discovered switch is matched to predefined switch, **otherinterfaces** attribute is updated to dhcp IP address, and mac address, **switch type** and **usercomment** also updated with vendor information for the predefined switch.

**ip\_configured** — switch is set up to static IP address based on predefined switch IP address. If failure to set up IP address, the status will stay as **Matched**.

**hostname\_configured** – switch host name is changed based on predefined switch hostname. If failure to change hostname on the switch, the status will stay as **ip\_configured**.

**switch\_configured** – snmpv3 is setup for the switches. This should be final status after running **switchdiscover --setup** command. If failure to setup snmpv3, the status will stay as **hostname\_configured**.

## VLANs

### VLAN Configuration

#### Overview

The main intent for this feature is the following scenario: you have a cluster with 100 nodes all on the Ethernet switch. A user requests 10 nodes from the cluster, but wants to run some sensitive things on the 10 nodes, so wants them isolated from the other 90 nodes. So as part of scheduling these 10 nodes for this user, you want to create a VLAN for these 10 nodes that is separate from the LAN the other 90 nodes are on.

xCAT has provided the following commands to do the VLAN creation and manipulation in the **xCAT-vlan** package.

- mkvlan
- chvlan
- lsvlan
- rmvlan

These VLAN functions are supported for stand-alone nodes as well as virtual machines such as KVMs.

## Install the package

Install the xCAT-vlan package using the package manager on the OS.

[RHEL]

```
yum install xCAT-vlan
```

[SLES]

```
zypper install xCAT-vlan
```

[Ubuntu]

```
apt-get install xCAT-vlan
```

## Prepare the Cluster

Assume the management node is installed and the other nodes are defined in the cluster. The following is what you need to do in order to use the VLAN Configuration feature.

### 1. Populate the switches table

Make sure the correct SNMP information is entered into the **switches** table for the switches involved. If there are more than one switches involved in a vlan, the ports that connect to the switches need to be entered in switches.linkports with the following format:

```
<port number>:switch,<port number>:switch....
```

For example:

```
# switch,snmpversion,username,password,privacy,auth,linkports,comments,disable
"switch1","3","user1","passw0rd",,"sha","42:switch2",,
"switch2","2c",,,,,,"50:switch1",,
"switch3","3","admin","passw0rd","des","sha",,,
"switch4","2c",,,"mycommunity",,,,,,
```

This means port 42 of switch1 is connected to port 50 of switch2. And switch1 can be accessed using SNMP version 3 and switch 2 can be accessed using SNMP version 2.

---

**Note:** The **username** and the **password** on the switches table are NOT the same as SSH user name and password. You have to configure SNMP on the switch for these parameters and then fill up this table. Use **tabdump switches -d** command to find out the meaning of each column.

---

### 2. Populate the switch table

Make sure each node has the correct switch name and port number in the **switch** table.

For example:

```
# node,switch,port,vlan,interface,comments,disable
"node1","switch1","33",,,,
"node2","switch1","34",,,,
"node3","switch2","10",,,,

```

For xCAT-vlan 2.7.5 and later versions, it supports creating vlans for other networks as well. To specify other networks in the switch table, the switch.interface must be specified. For example:

```
# node,switch,port,vlan,interface,comments,disable
"node1","switch1","33",,,,
"node2","switch1","34",,"primary",,
"node3","switch2","10",,"primary:eth0",,
"node1","switch2","11",,"eth1",,
"node2","switch2","12",,"eth1",,
"node3","switch2","13",,"eth1",,
```

The interface eth1 is for the application network on node1, node2 and node3. Note that there are two rows for each node. One is for the management network and the other is for the application network. The value for **switch.interface** for management network can be empty, the word “primary” or “primary:ethx”.

### 3. Configure the switch for SNMP access

Make sure that the MN can access the switch using SNMP and the switch is configured such that it has SNMP read and write permissions.

You can use **snmpwalk/snmpget** and **snmpset** commands on the mn to check. These commands are from **net-snmp-utils** rpm.

### 4. Define the VLAN subnet and mask pattern

The **site** table keys called “vlannets” and “vlanmasks” will be used to define a range of networks that can be used to define vlans. The format is a regular expression.

For example:

```
vlannets: |(\d+)|10.($1+0).0.0|
vlanmasks: 255.255.255.0
```

This means that the network for the vlan id 5 will be 10.5.0.0 and the mask is 255.255.255.0.

However, user can also customize a vlan network and netmask using -t and -m flags on **mkvlan** command.

### 5. Customize host names and ip addresses for nodes

Within the vlan, by default the hostnames for the nodes are having the following format:

```
v<vlanid>nY
```

where Y is the node number.

For example, the hostname for node 5 on vlan 10 is v10n5.

User can customize the host name and ip addresses using the **hosts** table. If the host name and ip addresses are found on the **hosts.otherinterfaces**, then it will be used. For example:

```
#node,ip,hostnames,otherinterfaces,comments,disable
"node1","192.168.1.1",,"test1:10.0.0.1",,
"node2","192.168.1.2",,"test2:10.0.0.2",,
```

### 6. For KVM clients

If you are going to include KVM clients in the VLANs, set the site table key “usexhrm” to be 1.

```
chdef -t site usexhrm=1
```

## Create a VLAN

For standalone nodes, VLAN can be created while the nodes are running or down.

To make a private vlan for stand-alone nodes for the management network:

```
mkvlan -n node1,node2,node3
```

You can specify vlan id, subnet and netmask etc.

```
mkvlan 3 -n node1,node2,node3 -t 10.3.2.0 -m 255.255.255.0
```

For virtual machines, the vm guests must be down. To make a private vlan for KVM guests.

```
chdef -t site -o clustersite usexhrm=1
mkdef node1 arch=x86_64 groups=kvm,all installnic=mac primarynic=mac mgt=kvm netboot=pxe
↪nfsserver=10.1.0.204 os=rhels6 profile=compute provmethod=install serialport=0
↪serialspeed=115200 vmcpus=1 vmhost=x3650n01 vmmemory=512 vmnics=br0 vmstorage=nfs://10.
↪1.0.203/vms
mkdef node2 arch=x86_64 groups=kvm,all installnic=mac primarynic=mac mgt=kvm netboot=pxe
↪nfsserver=10.1.0.204 os=rhels6 profile=compute provmethod=install serialport=0
↪serialspeed=115200 vmcpus=1 vmhost=x3650n01 vmmemory=512 vmnics=br0 vmstorage=nfs://10.
↪1.0.203/vms
mkvlan -n node1,node2
mkvm node1,node2 -s 20G
rpower node1,node2 on
rinstall node1,node2
```

For xCAT-vlan 2.7.5 and later versions, you can create vlans for other networks. This can be done by using -i flag to specify the interface of the network. For example:

```
mkvlan -n node1,node2,node3 -i eth1
```

A tagged vlan will be created for the network that is on eth1 for node1, node2 and node3. For KVM clients, -i specifies the interface name on the KVM host that the vlan will be tagged on. If -i is omitted, the management networks will be assumed.

---

**Note:** After the vlan is created, the nodes can still be accessed by the mn using the management network. You can use **lsvlan** command to list all the vlans.

---

For example:

```
# lsvlan
vlan 3:
  subnet 10.3.0.0
  netmask 255.255.0.0
vlan 99:
  subnet 10.99.0.0
  netmask 255.255.0.0

# lsvlan 3
vlan 3
  subnet 10.3.0.0
```

(continues on next page)



(continued from previous page)

```
netmask 255.255.0.0
hostname      ip address      node          vm host
v3n1          10.3.0.1      node1
v3n2          10.3.0.2      node2
v3n3          10.3.0.3      node3          host1
```

## Modify a VLAN

You can use the **chvlan** command to add or remove nodes to/from an existing vlan.

For standalone nodes, just run the command while the node are running or not. For example:

To add

```
chvlan 3 -n node4,node5
```

To remove

```
chvlan 3 -n node4,node5 -d
```

For virtual machines, adding them to the vlan requires that they are defined and they are not up and running.

For example:

```
mkdef node4 arch=x86_64 groups=kvm,all installnic=mac primarynic=mac mgt=kvm netboot=pxe
↪nfsserver=10.1.0.204 os=rhels6 profile=compute provmethod=install serialport=0
↪serialspeed=115200 vmcpus=1 vmhost=x3650n01 vmmemory=512 vmnics=br0 vmstorage=nfs://10.
↪1.0.203/vms
mkdef node5 arch=x86_64 groups=kvm,all installnic=mac primarynic=mac mgt=kvm netboot=pxe
↪nfsserver=10.1.0.204 os=rhels6 profile=compute provmethod=install serialport=0
↪serialspeed=115200 vmcpus=1 vmhost=x3650n01 vmmemory=512 vmnics=br0 vmstorage=nfs://10.
↪1.0.203/vms
chvlan 3 -n node4,node5
mkvm node4,node5 -s 20G
rpower node4,node5 on
rinstall node4,node5
```

For xCAT-vlan 2.7.5 and later versions, you can modify vlans for other networks. This can be done by using **-i** flag to specify the interface of the network. For KVM clients, **-i** specifies the interface name on the KVM host that the vlan will be tagged on. If **-i** is omitted, the management networks will be assumed.

For example:

```
chvlan 3 -n node4,node5 -i eth1
```

There is no need to specify **-i** flag for removing nodes from a vlan.

## Remove a VLAN

The **rmvlan** command removes the given vlan ID from the cluster. It removes the vlan id from all the switches involved, deconfigures the nodes so that vlan adaptor (tag) will be removed, cleans up /etc/hosts, DNS and database tables for the given vlan.

For example:

```
rmvlan 3
```

## VLAN Security

To make the vlan more secure, the root guard and the bpdu guard are enabled for each ports within the vlan by **mkvlan** and **chvlan** commands. This way it guards the topology changes on the switch by the hackers who hack the STP. However, when the vlan is removed by the **rmvlan** and the **chvlan (-d)** commands, the root guard and the bpdu guard are not disabled because the code cannot tell if the guards were enabled by the admin or not. If you want to remove the guards after the vlan is removed, you need to use the switch command line interface to do so. Refer to the documents for the switch command line interfaces for details.

## Limitation

Current xCAT-vlan package does not work on the following os distributions. More work will be done in the future releases.

- ubuntu
- rhel7 and later
- sles12 and later

## InfiniBand (Mellanox)

xCAT has the ability to help with Mellanox InfiniBand (IB) adapter installation and network configuration as part of the node provisioning process.

## Mellanox OFED Installation Script

Mellanox provides a tested and packaged version of the OpenFabrics Enterprise Distribution (OFED) driver, named Mellanox OFED (MLNX\_OFED). To assist with the installation of the MLNX\_OFED driver, xCAT provides a sample postscript: `mlnxofed_ib_install`.

## Preparation

### Download MLNX\_OFED ISO

**xCAT only supports installation using the ISO format.**

Download the Mellanox OFED ISO file [here \(MLNX\\_OFED\)](#).

## Prepare Installation Script

The `mlnxofed_ib_install` is a sample script intended to assist with the installation of the Mellanox OFED drivers. The following support matrix documents the limited number of scenarios that have been verified: [support matrix](#).

1. Copy the `mlnxofed_ib_install` to `/install/postscripts`, renaming to `mlnxofed_ib_install`.

```
cp /opt/xcat/share/xcat/ib/scripts/Mellanox/mlnxofed_ib_install \
/install/postscripts/mlnxofed_ib_install

# ensure the script has execute permission
chmod +x /install/postscripts/mlnxofed_ib_install
```

2. Familiarize the options available for the xCAT `mlnxofed_ib_install` script.

Op- tion	Re- quire	Description
-p	Yes	The full path to the MLNX_OFED ISO image
-m	No	Use this option to pass arguments to the Mellanox OFED installation script <code>mlnxofedinstall</code> . The special keyword <code>-end-</code> must be added to the end of the string to mark the completion of the option list option list. If nothing is specified, xCAT passes the following <code>--without-32bit --without-fw-update --force</code>
-i	For disk- less	The image root path of the diskless image
-n	For disk- less	nodeset status, value is <code>genimage</code>

A very basic usage of the install script:

```
/install/postscripts/mlnxofed_ib_install -p /install/<path-to>/<MLNX_OFED_LINUX.iso>
```

To pass the `--add-kernel-support` option to `mlnxofedinstall`, use the following command:

```
/install/postscripts/mlnxofed_ib_install -p /install/<path-to>/<MLNX_OFED_LINUX.iso>
↔ \
-m --without-32bit --without-fw-update --add-kernel-support --force -end-
```

## Configuration

The process to configure the `osimage` to install the Mellanox OFED Drivers for Diskful and Diskless scenarios are outlined below.

## Diskful Installation

### 1. Prepare dependency packages in the pkglist

In order for the Mellanox installation script to execute successfully, certain dependency packages are required to be installed on the compute node. xCAT provides sample package list files to help resolve these dependencies. The samples are located at `/opt/xcat/share/xcat/ib/netboot/<os>/`.

To use the `/opt/xcat/share/xcat/ib/netboot/rh/ib.rhels7.ppc64le.pkglist`, edit your existing `pkglist` file for the target osimage and add the following at the bottom:

```
#INCLUDE:/opt/xcat/share/xcat/ib/netboot/rh/ib.rhels7.ppc64le.pkglist#
```

### 2. Configure the mlnxofed\_ib\_install script to install the MLNX\_OFED drivers

xCAT has a concept of postscripts that can be used to customize the node after the operating system is installed.

Mellanox recommends that the operating system is rebooted after the drivers are installed, so xCAT recommends using the `postscripts` attribute to avoid the need for a second reboot. To invoke the `mlnxofed_ib_install` as a postscript

```
chdef -t node -o <node_name> \
  -p postscripts="mlnxofed_ib_install -p /install/<path-to>/<MLNX_OFED_LINUX.iso>"
```

**[kernel mismatch issue]** The Mellanox OFED ISO is built against a series of specific kernel version. If the version of the linux kernel does not match any of the Mellanox offered pre-built kernel modules, you can pass the `--add-kernel-support --force` argument to the Mellanox installation script to build the kernel modules based on the version you are using.

```
chdef -t node -o <node_name> \
  -p postscripts="mlnxofed_ib_install -p /install/<path-to>/<MLNX_OFED_LINUX.iso> \
  -m --add-kernel-support --force -end-"
```

### 3. Provision the node

```
rinstall <node> osimage=
```

### 4. Verification

- Check the status of `openibd` service

sysVinit:

```
service openibd status
```

systemd:

```
systemctl status openibd.service
```

- Verify that the Mellanox IB drivers are located at: `/lib/modules/<kernel_version>/extra/`
- Use the `ibv_devinfo` command to obtain information about the InfiniBand adapter.

## Diskless Installation

### 1. Prepare dependency packages in the pkglist

In order for the Mellanox installation script to execute successfully, certain dependency packages are required to be installed on the compute node. xCAT provides sample package list files to help resolve these dependencies. The samples are located at `/opt/xcat/share/xcat/ib/netboot/<os>/`.

To use the `/opt/xcat/share/xcat/ib/netboot/rh/ib.rhels7.ppc64le.pkglist`, edit your existing `pkglist` file for the target osimage and add the following at the bottom:

```
#INCLUDE:/opt/xcat/share/xcat/ib/netboot/rh/ib.rhels7.ppc64le.pkglist#
```

### 2. Configure the mlnxofed\_ib\_install script to install the MLNX\_OFED drivers

Edit the `postinstall` script on the osimage to invoke the `mlnxofed_ib_install` install script.

For example, take `rhels7.2-ppc64le-netboot-compute`:

#### 1. Find the path to the postinstall script:

```
# lsdef -t osimage -o rhels7.2-ppc64le-netboot-compute -i postinstall
Object name: rhels7.2-ppc64le-netboot-compute
      postinstall=/opt/xcat/share/xcat/netboot/rh/compute.rhels7.ppc64le.
      ↪postinstall
```

#### 2. Edit the `/opt/xcat/share/xcat/netboot/rh/compute.rhels7.ppc64le.postinstall` and add the following:

```
/install/postscripts/mlnxofed_ib_install \
-p /install/<path-to>/<MLNX_OFED_LINUX.iso> -i $1 -n genimage
```

**Note:** The `$1` is a argument that is passed to the `postinstall` script at runtime.

### Tip: Kernel Mismatch

The Mellanox OFED ISO is built against a series of specific kernel version. If the version of the linux kernel being used does not match any of the pre-built kernels, pass `--add-kernel-support --without-32bit --without-fw-update --force` to the Mellanox installation script to build the kernel modules based on the kernel you are using. Example:

```
/install/postscripts/mlnxofed_ib_install \
-p /install/<path-to>/<MLNX_OFED_LINUX.iso> -m --add-kernel-support --without-32bit ↪
      ↪--without-fw-update --force -end- \
-i $1 -n genimage
```

### 3. Generate the diskless image

Use the `genimage` command to generate the diskless image from the osimage definition

```
genimage <osimage>
```

Use the `packimage` command to pack the diskless image for deployment

```
packimage <osimage>
```

#### 4. Provision the node

```
rinstall <node> osimage=rhels7.2-ppc64le-netboot-compute
```

#### 5. Verification

- Check the status of openibd service

sysVinit:

```
service openibd status
```

systemd:

```
systemctl status openibd.service
```

- Verify that the Mellanox IB drivers are located at: `/lib/modules/<kernel_version>/extra/`
- Use the `ibv_devinfo` command to obtain information about the InfiniBand adapter.

## MLNX\_OFED Support Matrix

The following ISO images and attributes have been verified by the xCAT Team.

### RedHat Enterprise Linux

RHEL 7.2 (ppc64le)	
<b>OFED ISO</b>	MLNX_OFED_LINUX-3.2-2.0.0.0-rhel7.2-ppc64le.iso
<b>Attribute Supported</b>	–without-32bit –without-fw-update –add-kernel-support –force
<b>IB.pkglist</b>	ib.rhels7.ppc64le.pkglist

RHEL 7.1 (ppc64)	
<b>OFED ISO</b>	MLNX_OFED_LINUX-3.2-2.0.0.0-rhel7.1-ppc64.iso
<b>Attribute Supported</b>	–without-32bit –without-fw-update –add-kernel-support –force
<b>IB.pkglist</b>	ib.rhels7.ppc64.pkglist

### Suse Linux Enterprise Server

SLES 12 (ppc64le)	
<b>OFED ISO</b>	MLNX_OFED_LINUX-3.2-2.0.0.0-sles12sp1-ppc64le.iso
<b>Attribute Supported</b>	–without-32bit –without-fw-update –force
<b>IB.pkglist</b>	ib.sles12.ppc64le.pkglist

### Ubuntu

Ubuntu14.04.3 (ppc64le)	
<b>OFED ISO</b>	MLNX_OFED_LINUX-3.2-2.0.0.0-ubuntu14.04-ppc64le.iso
<b>Attribute Supported</b>	–without-32bit –without-fw-update –add-kernel-support –force
<b>IB.pkglist</b>	ib.ubuntu14.ppc64le.pkglist

## Known Issues

### Preventing upgrade of the Mellanox Drivers

On RedHat operating systems, after the Mellanox drivers are installed, you may have a requirement to update your operating system to a later version. Some operating systems may ship InfiniBand drivers that are higher version than the Mellanox drivers you have installed and therefor may update the existing drivers.

To prevent this from happening, add the following in the `/etc/yum.conf`

```
exclude=dapl* libib* ibacm infiniband* libmlx* librdma* opensm* ibutils*
```

### Development packages in SLES

If using the `--add-kernel-support` attribute on SLES operating systems, you may find problems with installing some dependency packages which are not shipped by the SLES server DVDs. The development rpms are provided by the SDK DVDs. Refer to [Add Additional Software Packages](#) to configure the SDK repositories.

## IB Network Configuration

xCAT provides a script `configib` to help configure the Infiniband adapters on the compute nodes.

The Infiniband adapter is considered an additional interface for xCAT. The process for configuring Infiniband adapters complies with the process of [Configure Additional Network Interfaces](#).

Below are an simple example to configure Mellanox IB in Ubuntu 14.04.1 on Power8 LE

If your target Mellanox IB adapter has 2 ports, and you plan to give port `ib0` 4 different IPs, 2 are IPV4 (20.0.0.3 and 30.0.0.3) and another 2 are IPV6 (1:2::3 and 2:2::3).

1. Define your networks in networks table

```
chdef -t network -o ib0ipv41 net=20.0.0.0 mask=255.255.255.0 mgtifname=ib0
chdef -t network -o ib0ipv42 net=30.0.0.0 mask=255.255.255.0 mgtifname=ib0
chdef -t network -o ib0ipv61 net=1:2::/64 mask=/64 mgtifname=ib0 gateway=1:2::2
chdef -t network -o ib0ipv62 net=2:2::/64 mask=/64 mgtifname=ib0 gateway=
```

2. Define IPs for `ib0`

```
chdef <node> nicips.ib0="20.0.0.3|30.0.0.3|1:2::3|2:2::3" \
nicnetworks.ib0="ib0ipv41|ib0ipv42|ib0ipv61|ib0ipv62" nictypes.ib0="Infiniband"
```

3. Configure `ib0`

- To configure during node installation

```
chdef <node> -p postscripts="confignetwork --ibaports=2"
nodeset <node> osimage=<osimagename>
rsetboot <node> net
rpower <node> reset
```

- To configure on a node which has had operating system

```
updatenode <node> -P "confignetwork --ibaports=2"
```

## IB Switch Configuration

### Setup the xCAT Database

The Mellanox Switch is only supported in xCAT Release 2.7 or later.

Add the switch ip address in the /etc/hosts file

Define IB switch as a node:

```
chdef -t node -o mswitch groups=all nodetype=switch mgt=switch
```

Add the login user name and password to the switches table:

```
tabch switch=mswitch switches.sshusername=admin switches.sshpassword=admin switches.
↪switchtype=MellanoxIB
```

The switches table will look like this:

```
#switch,...,sshusername,sshpassword,switchtype,...
"mswitch",,,,,,"admin","admin","MellanoxIB",,
```

If there is only one admin and one password for all the switches then put the entry in the xCAT passwd table for the admin id and password to use to login.

```
tabch key=switch passwd.username=admin passwd.password=admin
```

The passwd table will look like this:

```
#key,username,password,cryptmethod,comments,disable
"switch","admin","admin",,,,
```

### Setup ssh connection to the Mellanox Switch

To run commands like xdsh and script to the Mellanox Switch, we need to setup ssh to run without prompting for a password to the Mellanox Switch. To do this, first you must add a configuration file. This configuration file is NOT needed for xCAT 2.8 and later.

```
mkdir -p /var/opt/xcat/IBSwitch/Mellanox
cd /var/opt/xcat/IBSwitch/Mellanox
cp /opt/xcat/share/xcat/devicetype/IBSwitch/Mellanox/config .
```

The file contains the following:



```
[main]
[xdsh]
pre-command=cli
post-command=NULL
```

Then run the following:

```
rspconfig mswitch sshcfg=enable
```

**Warning:** For Mellanox switch in manufacturing defaults, the user may need to log in once and answer **no** to the configuration wizard as `rspconfig` will fail when prompted against the wizard.

## Setup syslog on the Switch

Use the following command to consolidate the syslog to the Management Node or Service Nodes, where `ip` is the address of the MN or SN as known by the switch.

```
rspconfig mswitch logdest=<ip>
```

## Configure xdsh for Mellanox Switch

To run `xdsh` commands to the Mellanox Switch, you must use the `--devicetype` input flag to `xdsh`. In addition, for xCAT versions less than 2.8, you must add a configuration file, see [Setup ssh connection to the Mellanox Switch](#) section.

For the Mellanox Switch the `--devicetype` is `IBSwitch::Mellanox`. See [xdsh man page](#) for details.

Now you can run the switch commands from the mn using `xdsh`. For example:

```
xdsh mswitch -l admin --devicetype IBSwitch::Mellanox \
'enable;configure terminal;show ssh server host-keys'
```

## Commands Supported for the Mellanox Switch

Setup the snmp alert destination:

```
rspconfig <switch> snmpdest=<ip> [remove]
```

where “remove” means to remove this ip from the snmp destination list.

Enable/disable setting the snmp traps.

```
rspconfig <switch> alert=enable/disable
```

Define the read only community for snmp version 1 and 2.

```
rspconfig <switch> community=<string>
```

Enable/disable snmp function on the switch.

```
rspconfig <switch> snmpcfg=enable/disable
```

Enable/disable ssh-ing to the switch without password.

```
rspconfig <switch> sshcfg=enable/disable
```

Setup the syslog remove receiver for this switch, and also define the minimum level of severity of the logs that are sent. The valid levels are: emerg, alert, crit, err, warning, notice, info, debug, none, remove. “remove” means to remove the given ip from the receiver list.

```
rspconfig <switch> logdest=<ip> [<level>]
```

For doing other tasks on the switch, use xdsh. For example:

```
xdsh mswitch -l admin --devicetype IBSwitch::Mellanox 'show logging'
```

Interactive commands are not supported by xdsh. For interactive commands, use ssh.

## Send SNMP traps to xCAT Management Node

First, get [http://www.mellanox.com/related-docs/prod\\_ib\\_switch\\_systems/MELLANOX-MIB.zip](http://www.mellanox.com/related-docs/prod_ib_switch_systems/MELLANOX-MIB.zip) , unzip it. Copy the mib file MELLANOX-MIB.txt to /usr/share/snmp/mibs directory on the mn and sn (if the sn is the snmp trap destination.)

Then,

To configure, run:

```
monadd snmpmon  
moncfg snmpmon <mswitch>
```

To start monitoring, run:

```
monstart snmpmon <mswitch>
```

To stop monitoring, run:

```
monstop snmpmon <mswitch>
```

To deconfigure, run:

```
mondecfg snmpmon <mswitch>
```

For more details on monitoring the cluster: TODO [Monitoring\\_an\\_xCAT\\_Cluster/#snmp-monitoring](#)

## UFM Configuration

UFM server are just regular Linux boxes with UFM installed. xCAT can help install and configure the UFM servers. The XCAT mn can send remote command to UFM through xdsh. It can also collect SNMP traps and syslogs from the UFM servers.

### Setup xdsh to UFM and backup

Assume we have two hosts with UFM installed, called host1 and host2. First define the two hosts in the xCAT cluster. Usually the network that the UFM hosts are in a different than the compute nodes, make sure to assign correct servicenode and xcatmaster in the noderes table. And also make sure to assign correct os and arch values in the nodetype table for the UFM hosts. For example:

```
mkdef -t node -o host1,host2 groups=ufm,all os=sles11.1 arch=x86_64 servicenode=10.0.0.1
↪ xcatmaster=10.0.0.1
```

Then exchange the SSH key so that it can run xdsh.

```
xdsh host1,host2 -K
```

Now we can run xdsh on the UFM hosts.

```
xdsh ufm date
```

### Consolidate syslogs

Run the following command to make the UFM hosts to send the syslogs to the xCAT mn:

```
updatenode ufm -P syslog
```

To test, run the following commands on the UFM hosts and see if the xCAT MN receives the new messages in /var/log/messages

```
logger xCAT "This is a test"
```

### Send SNMP traps to xCAT Management Node

You need to have the Advanced License for UFM in order to send SNMP traps.

1. Copy the mib file to /usr/share/snmp/mibs directory on the mn.

```
scp ufmhost:/opt/ufm/files/conf/vol_ufm3_0.mib /usr/share/snmp/mibs
```

Where ufmhost is the host where UFM is installed.

2. On the UFM host, open the /opt/ufm/conf/gv.cfg configuration file. Under the [Notifications] line, set

```
snmp_listeners = <IP Address 1>[:<port 1>][,<IP Address 2>[:<port 2>].]
```

The default port is 162. For example:

```
ssh ufmhost
vi /opt/ufm/conf/gv.cfg

....
[Notifications]
snmp_listeners = 10.0.0.1
```

Where 10.0.0.1 is the ip address of the management node.

3. On the UFM host, restart the ufmd

```
service ufmd restart
```

4. From UFM GUI, click on the “Config” tab; bring up the “Event Management” Policy Table. Then select the SNMP check boxes for the events you are interested in to enable the system to send an SNMP traps for these events. Click “OK”.
5. Make sure snmptrapd is up and running on mn and all monitoring servers.

It should have the ‘-m ALL’ flag.

```
ps -ef |grep snmptrapd
root 31866 1 0 08:44 ? 00:00:00 /usr/sbin/snmptrapd -m ALL
```

If it is not running, then run the following commands:

```
monadd snmpmon
monstart snmpmon
```

## Firmware Updates

### Adapter Firmware Update

Download the OFED IB adapter firmware from the Mellanox site [http://www.mellanox.com/page/firmware\\_table\\_IBM](http://www.mellanox.com/page/firmware_table_IBM).

Obtain device id:

```
lspci | grep -i mel
```

Check current installed fw level:

```
mstflint -d 0002:01:00.0 q | grep FW
```

Copy or mount firmware to host:

Burn new firmware on each ibaX:

```
mstflint -d 0002:01:00.0 -i <image location> b
```

---

**Note:** If this is a PureFlex MezzanineP adapter, you must select the correct image for each ibaX device.

**The difference in the firmware image at the end of the file name:**

- \_0.bin (iba0/iba2)

- \_1.bin (iba1/iba3)

Verify download successful:

```
mstflint -d 0002:01:00.0 q
```

Activate the new firmware:

```
reboot the image
```

**Note:** The above 0002:01:00.0 device location is used as an example only. Validate your device location using the `lspci` command.

## Mellanox Switch Firmware Upgrade

This section provides manual procedure to help update the firmware for Mellanox Infiniband (IB) Switches. You can download IB switch firmware like IB6131 (image-PPC\_M460EX-SX\_3.2.xxx.img) from the Mellanox website [http://www.mellanox.com/page/firmware\\_table\\_IBM](http://www.mellanox.com/page/firmware_table_IBM) and place into your xCAT Management Node or server that can communicate to Flex IB6131 switch module. There are two ways to update the MLNX-OS switch package. This process works regardless if updating an internal PureFlex chassis Infiniband switch (IB6131) or for an external Mellanox switch.

### Update via Browser

This method is straight forward if your switches are on the public network or your browser is already capable to tunnel to the private address. If neither is the case then you may prefer to use option two.

After logging into the switch (id=admin, pwd=admin)

Select the “System” tab and then the “MLNX-OS Upgrade” option

Under the “Install New Image”, select the “Install via scp” URL: scp://userid@fwhost/directoryofimage/imagename

Select “Install Image”

The image will then be downloaded to the switch and the installation process will begin.

Once completed, the switch must be rebooted for the new package to be activate

### Update via CLI

Login to the IB switch:

```
ssh admin@<switchipaddr>
enable (get into correct CLI mode. You can use en)
configure terminal (get into correct CLI mode. You can use co t)
```

List current images and Remove older images to free up space:

```
show image
image delete <ibimage>
(you can paste in ibimage name from show image for image delete)
```

Get the new IB image using fetch with scp to a server that contains new IB image. An example of IB3161 image would be “image-PPC\_M460EX-SX\_3.2.0291.img” Admin can use different protocol . This image fetch scp command is about 4 minutes.

```
image fetch ?
image fetch scp://userid:password@serveripddr/<full path ibimage location>
```

Verify that new IB image is loaded, then install the new showIB image on IB switch. The install image process goes through 4 stages Verify image, Uncompress image, Create Filesystems, and Extract Image. This install process takes about 9 minutes.

```
show image
image install <newibimage>
(you can paste in new IB image from "show image" to execute image install)
```

Toggle boot partition to new IB image, verify image install is loaded , and that next boot setting is pointing to new IB image.

```
image boot next
show image
```

Save the changes made for new IB image:

```
configuration write
```

Activate the new IB image (reboot switch):

```
reload
```

For more information about Mellanox products, refer to <http://www.mellanox.com>.

## Predict network adapter name during discovery

Traditionally, network interfaces in Linux are enumerated as eth[0123...], but these names do not correspond to actual labels on the chassis. Now, most of the linux distribution support naming the adapter with slot information which makes adapter name predictable. xCAT add `getadapter` script which can be run during discovery stage to detect the adapter names and pci slot information to help customer configure the network.

## How to use getadapter

Set the chain table to run `getadapter` script

```
chdef <noderange> chain="runcmd=getadapter"
```

After the discovery completed, the column `nicsadapter` of `nics` table is updated.

View result with `lsdef` command

```
# lsdef <node>
.....
nicsadapter.enP3p3s0f0=mac=98:be:94:59:fa:cc linkstate=DOWN pci=/pci0003:00/0003:00:00.0/
↳0003:01:00.0/0003:02:01.0/0003:03:00.0 candidatename=enP3p3s0f0/enx98be9459facc
nicsadapter.enP3p3s0f1=mac=98:be:94:59:fa:cd linkstate=DOWN pci=/pci0003:00/0003:00:00.0/
```

(continues on next page)

(continued from previous page)

```

↪0003:01:00.0/0003:02:01.0/0003:03:00.1 candidatename=enP3p3s0f1/enx98be9459facd
nicsadapter.enP3p3s0f2=mac=98:be:94:59:fa:ce linkstate=DOWN pci=/pci0003:00/0003:00:00.0/
↪0003:01:00.0/0003:02:01.0/0003:03:00.2 candidatename=enP3p3s0f2/enx98be9459face
nicsadapter.enP3p3s0f3=mac=98:be:94:59:fa:cf linkstate=UP pci=/pci0003:00/0003:00:00.0/
↪0003:01:00.0/0003:02:01.0/0003:03:00.3 candidatename=enP3p3s0f3/enx98be9459facf
.....

```

Below are the information `getadapter` trying to inspect:

- **name:** the real adapter name used by genesis operation system
- **pci:** the pci slot location
- **mac:** the MAC address
- **candidatename:** All the names which satisfy predictable network device naming scheme, if customer needs to customize their network adapter name, they can choose one of them. (`confignetwork` needs to do more work to support this. if customer want to use their own name, xcat should offer a interface to get customer's input and change this column)
- **linkstate:** The link state of network device

## 1.5.14 PDUs

Power Distribution Units (PDUs) are devices that distribute power to servers in a frame. They have the capability of monitoring the amount of power that is being used by devices plugged into it and cycle power to individual receptacles. xCAT can support two kinds of PDUs, infrastructure PDU (`irpdu`) and collaborative PDU (`crpdu`).

The Infrastructure rack PDUs are switched and monitored 1U PDU products which can connect up to nine C19 devices or up to 12 C13 devices and an additional three C13 peripheral devices to a single dedicated power source. The Collaborative PDU is on the compute rack and has the 6x IEC 320-C13 receptacles that feed the rack switches. These two types of PDU have different design and implementation. xCAT has different code path to maintains PDU commands via **pdutype**.

## Discovering PDUs

xCAT provides `pdudiscover` command to discover the PDUs that are attached to the neighboring subnets on xCAT management node.

```
pdudiscover [<noderange>|--range ipranges] [-r|-x|-z] [-w] [-V|--verbose] [--setup]
```

xCAT uses `snmp` scan method to discover PDU. Make sure `net-snmp-utils` package is installed on xCAT MN in order to use `snmpwalk` command.

Options:

```

--range Specify one or more IP ranges. Each can be an ip address (10.1.2.3) or an ip_
↪range
        (10.1.2.0/24). If the range is huge, for example, 192.168.1.1/8, the pdu
        discover may take a very long time to scan. So the range should be exactly
        specified. It accepts multiple formats. For example:
        192.168.1.1/24, 40-41.1-2.3-4.1-100.

        If the range is not specified, the command scans all the subnets that the_
↪active

```

(continues on next page)

(continued from previous page)

```

network interfaces (eth0, eth1) are on where this command is issued.
-r      Display Raw responses.
-x      XML formatted output.
-z      Stanza formatted output.
-w      Writes output to xCAT database.
--setup Process switch-based pdu discovery and configure the PDUs. For crpdu, --
↳setup options will configure passwordless , change ip address from dhcp to static,
↳hostname changes and snmp v3 configuration. For irpdu, it will configure ip address,
↳and hostname. It required predefined PDU node definition with switch name and switch,
↳port attributes for mapping.

```

## Define PDU Objects

1. Define pdu object

```
mkdef f5pdu3 groups=pdu ip=50.0.0.8 mgt=pdu nodetype=pdu pdutype=irpdu
```

2. Define switch attribute for pdu object which will be used for pdudiscover **--setup** options.

```
chdef f5pdu3 switch=mid08 switchport=3
```

3. Add hostname to /etc/hosts:

```
makehosts f5pdu3
```

4. Verify the SNMP command responds against the PDU:

```
snmpwalk -v1 -cpublic -mALL f5pdu3 system
```

## Infrastructure PDU

Users can access Infrastructure PDU via telnet and use the **IBM PDU Configuration Utility** to set up and configure the PDU. xCAT supports PDU commands for power management and monitoring through SNMP.

## PDU Commands

Administrators will need to know the exact mapping of the outlets to each server in the frame. xCAT cannot validate the physical cable is connected to the correct server.

Add a pdu attribute to the compute node definition in the form "PDU\_Name:outlet":

```

#
# Compute server cn01 has two power supplies
# connected to outlet 6 and 7 on pdu=f5pdu3
#
chdef cn01 pdu=f5pdu3:6,f5pdu3:7

```

The following commands are supported against a compute node:

- Check the pdu status for a compute node:



```
# rpower cn01 pdustat
cn01: f5pdu3 outlet 6 is on
cn01: f5pdu3 outlet 7 is on
```

- Power off the PDU outlets for a compute node:

```
# rpower cn01 pduoff
cn01: f5pdu3 outlet 6 is off
cn01: f5pdu3 outlet 7 is off
```

- Power on the PDU outlets for a compute node:

```
# rpower cn01 pduon
cn01: f5pdu3 outlet 6 is on
cn01: f5pdu3 outlet 7 is on
```

- Power cycling the PDU outlets for a compute node:

```
# rpower cn01 pdureset
cn01: f5pdu3 outlet 6 is reset
cn01: f5pdu3 outlet 7 is reset
```

The following commands are supported against a PDU:

- To change hostname of IR PDU:

```
# rspconfig f5pdu3 hosname=f5pdu3
```

- To change ip address of IR PDU:

```
# rsconfig f5pdu3 ip=x.x.x.x netmaks=255.x.x.x
```

- Check the status of the full PDU:

```
# rpower f5pdu3 stat
f5pdu3: outlet 1 is on
f5pdu3: outlet 2 is on
f5pdu3: outlet 3 is on
f5pdu3: outlet 4 is on
f5pdu3: outlet 5 is on
f5pdu3: outlet 6 is off
f5pdu3: outlet 7 is off
f5pdu3: outlet 8 is on
f5pdu3: outlet 9 is on
f5pdu3: outlet 10 is on
f5pdu3: outlet 11 is on
f5pdu3: outlet 12 is on
```

- Power off the full PDU:

```
# rpower f5pdu3 off
f5pdu3: outlet 1 is off
f5pdu3: outlet 2 is off
f5pdu3: outlet 3 is off
f5pdu3: outlet 4 is off
```

(continues on next page)

(continued from previous page)

```
f5pdu3: outlet 5 is off
f5pdu3: outlet 6 is off
f5pdu3: outlet 7 is off
f5pdu3: outlet 8 is off
f5pdu3: outlet 9 is off
f5pdu3: outlet 10 is off
f5pdu3: outlet 11 is off
f5pdu3: outlet 12 is off
```

- Power on the full PDU:

```
# rpower f5pdu3 on
f5pdu3: outlet 1 is on
f5pdu3: outlet 2 is on
f5pdu3: outlet 3 is on
f5pdu3: outlet 4 is on
f5pdu3: outlet 5 is on
f5pdu3: outlet 6 is on
f5pdu3: outlet 7 is on
f5pdu3: outlet 8 is on
f5pdu3: outlet 9 is on
f5pdu3: outlet 10 is on
f5pdu3: outlet 11 is on
f5pdu3: outlet 12 is on
```

- Power reset the full PDU:

```
# rpower f5pdu3 reset
f5pdu3: outlet 1 is reset
f5pdu3: outlet 2 is reset
f5pdu3: outlet 3 is reset
f5pdu3: outlet 4 is reset
f5pdu3: outlet 5 is reset
f5pdu3: outlet 6 is reset
f5pdu3: outlet 7 is reset
f5pdu3: outlet 8 is reset
f5pdu3: outlet 9 is reset
f5pdu3: outlet 10 is reset
f5pdu3: outlet 11 is reset
f5pdu3: outlet 12 is reset
```

- PDU inventory information:

```
# rinv f6pdu16
f6pdu16: PDU Software Version: "OPDP_sIBM_v01.3_2"
f6pdu16: PDU Machine Type: "1U"
f6pdu16: PDU Model Number: "dPDU4230"
f6pdu16: PDU Part Number: "46W1608"
f6pdu16: PDU Name: "IBM PDU"
f6pdu16: PDU Serial Number: "4571S9"
f6pdu16: PDU Description: "description"
```

- PDU and outlet power information:

```
# rvitals f6pdu15
f6pdu15: Voltage Warning: 0
f6pdu15: outlet 1 Current: 0 mA
f6pdu15: outlet 1 Max Capacity of the current: 16000 mA
f6pdu15: outlet 1 Current Threshold Warning: 9600 mA
f6pdu15: outlet 1 Current Threshold Critical: 12800 mA
f6pdu15: outlet 1 Last Power Reading: 0 Watts
f6pdu15: outlet 2 Current: 0 mA
f6pdu15: outlet 2 Max Capacity of the current: 16000 mA
f6pdu15: outlet 2 Current Threshold Warning: 9600 mA
f6pdu15: outlet 2 Current Threshold Critical: 12800 mA
f6pdu15: outlet 2 Last Power Reading: 0 Watts
f6pdu15: outlet 3 Current: 1130 mA
f6pdu15: outlet 3 Max Capacity of the current: 16000 mA
f6pdu15: outlet 3 Current Threshold Warning: 9600 mA
f6pdu15: outlet 3 Current Threshold Critical: 12800 mA
f6pdu15: outlet 3 Last Power Reading: 217 Watts
```

**Note:** For BMC based compute nodes, turning the PDU outlet power on does not automatically power on the compute side. Users will need to issue `rpower <node>` on to power on the compute side after the BMC boots.

## Collaborative PDU

Collaborative PDU is also referred as Coral PDU, it controls power for compute Rack. User can access PDU via SSH and can use the **PduManager** command to configure and manage the PDU product.

## Pre-Defined PDU Objects

A pre-defined PDU node object is required before running `pdudiscover` command.

```
mkdef coralpdu groups=pdu mgt=pdu nodetype=pdu (required)
```

all other attributes can be set by `chdef` command or `pdudiscover` command.

```
--switch      required for pdudiscover command to do mapping
--switchport   required for pdudiscover command to do mapping
--ip           ip address of the pdu.
--mac          can be filled in by pdudiscover command
--pdutype      crpdu(for coral pdu) or irpdu(for infrastructure PDUs)
```

The following attributes need to be set in order to configure snmp with non-default values.

```
--community    community string for coral pdu
--snmpversion   snmp version number, required if configure snmpv3 for coral pdu
--snmpuser      snmpv3 user name, required if configure snmpv3 for coral pdu
--authkey       auth passphrase for snmpv3 configuration
--authtype      auth protocol (MD5|SHA) for snmpv3 configuration
--privkey       priv passphrase for snmpv3 configuration
--privtype      priv protocol (AES|DES) for snmpv3 configuration
--seclvl        security level (noAuthNoPriv|authNoPriv|authPriv) for snmpv3 configuration
```

Make sure to run `makehosts` after pre-defined PDU.

```
makehosts coralpdu
```

## Configure PDUs

After pre-defining PDUs, user can use **pdudiscover --range ip\_range --setup** to configure the PDUs, or following commands can be used:

- To configure passwordless of Coral PDU:

```
# rspconfig coralpdu sshcfg
```

- To change hostname of Coral PDU:

```
# rspconfig coralpdu hostname=f5pdu3
```

- To change ip address of PDU:

```
# rsconfig coralpdu ip=x.x.x.x netmaks=255.x.x.x
```

- To configure SNMP community string or snmpv3 of PDU (the attribute needs to pre-defined):

```
# rspconfig coralpdu snmpcfg
```

## Remote Power Control of PDU

Use the **rpower** command to remotely power on and off PDU.

- To check power stat of PDU:

```
# rpower coralpdu stat
```

- To power off the PDU:

```
# rpower coralpdu off
```

- To power on the PDU:

```
# rpower coralpdu on
```

Coral PDUs have three relays, the following commands are for individual relay support of PDU:

- To check power stat of relay:

```
# rpower coralpdu relay=1 stat
```

- To power off the relay:

```
# rpower coralpdu relay=2 off
```

- To power on the relay:

```
# rpower coralpdu relay=3 on
```

## Show Monitor Data

Use the `rvitals` command to show realtime monitor data(input voltage, current, power) of PDU.

```
# rvitals coralpdu
```

## Show manufacture information

Use the `rinv` command to show MFR information of PDU

```
# rinv coralpdu
```

## 1.5.15 Port Usage

The following table lists the ports that must be open between the xCAT management node and the nodes it manages, unless otherwise noted. The xCAT service nodes use the same ports as the management node. A service (or protocol) applies to both AIX and Linux, unless stated otherwise. Service names are typical strings that appear in the `/etc/services` file, or in firewall/IP filtering logs. Local customization of the `/etc/services` files, daemon configuration options, like overriding the default port number, and differences in software source implementations, may yield other service information results.

The category of required or optional is difficult to fill in because depending on what function you are running what might be listed here as optional, may actually be required. The Trusted side is behind the firewall, the Non-trusted side is in front of the firewall.

### xCAT Port Usage Table

Service Name	Port number	Protocol	Range	Required or optional
xcatdport	3001	tcp		required
xcatdport	3001	udp		required
xcatiport	3002	tcp		required
xcatiport	3002	udp		required
xcatlport	3003(default)	tcp		optional
echo-udp	7	udp		required
ssh-tcp	22	tcp		required
ssh-udp	22	udp		required
rsync	873	tcp		required
rsync	873	udp		required
domain-tcp	53	tcp		optional
domain-udp	53	udp		optional
bootps	67	udp		required on aix and p-linux
dhcp	67	tcp		required on linux, optional on AIX
dhcpc	68	tcp		required on linux, optional on AIX
bootpc	68	udp		required on AIX
tftp-tcp	69	tcp		required
tftp-udp	69	udp		required
www-tcp	80	tcp		required
www-udp	80	udp		required
kerberos	88	tcp		not supported/used by xCAT anymore

continues on next page

Table 6 – continued from previous page

Service Name	Port number	Protocol	Range	Required or optional
kerberos	88	udp		not supported/used by xCAT anymore
sunrpc-udp	111	udp		required on linux statelite and AIX
shell	514	tcp	1-1023	optional
rsyslogd	514	tcp		required on linux
rsyslogd	514	udp		required on linux
kshell	544	tcp	1-1023	required on AIX
rmc-tcp	657	tcp	1-1023	required for RMC monitoring
rmc-udp	657	udp	1-1023	required for RMC monitoring
conserver	782	tcp		required on the mgmt and service nodes
nim	1058	tcp	1-1023	required on AIX
nfsd-tcp	2049	tcp	1-1023	required on linux statelite and AIX
nfsd-udp	2049	udp	1-1023	required on linux statelite and AIX
pxe	4011	tcp		required for linux
rpc-mount	100005	see Note2		required on linux statelite and AIX
mount-tcp	see Note1	tcp		required on linux statelite and AIX
mount-udp	see Note1	udp		required on linux statelite and AIX
awk	300	tcp		optional
ipmi	623	tcp		required on x86_64 and p8
ipmi	623	udp		required on x86_64 and p8
snmp	161	tcp		required on Flex
snmp	161	udp		required on Flex
snmptrap	162	tcp		required for snmp monitoring
snmptrap	162	udp		required for snmp monitoring

- xcatdport

The port used by the xcatd daemon for client/server communication.

- xcatiport

The port used by xcatd to receive install status updates from nodes.

- xcatlport

The port used by xcatd to record command log, you can customize it by edit site table, if you don't configure it, 3003 will be used by default.

- echo-udp

Needed by RSCT Topology Services.

- ssh-udp

Needed to use ssh. This service defines the protocol for upd. This is required when installing or running updatenode, xdsh,xdcp,psh,pcp through the firewall.

- rsync

Need to use updatenode or xdcp to rsync files to the nodes or service nodes.

- domain-tcp

Used when Domain Name Services (DNS) traffic from the Non-trusted nodes and the firewall node to a DNS server is explicitly handled by the firewall. Some firewall applications can be configured to explicitly handle all DNS traffic. This for tcp DNS traffic.

- domain-udp

Used when Domain Name Services (DNS) traffic from the Non-trusted nodes and the firewall node to a DNS server is explicitly handled by the firewall. Some firewall applications can be configured to explicitly handle all DNS traffic. This for udp DNS traffic.

- bootps

Bootp server port needed when installing an Non-trusted AIX or System p node through the firewall. This service is issued by the client to the Management Node , for an install request. It is not required to install the Non-trusted nodes through the firewall or to apply maintenance. This is the reason why the service is considered optional.

- dhcp

Needed to install Linux nodes through the firewall. This is the port for the dhcp server. This service defines the protocol for tcp.

- dhcpc

Needed to install Linux through the firewall. This is the port for the dhcp client. This service defines the protocol for tcp.

- bootpc

Bootp client port needed when installing an Non-trusted AIX or System p node through the firewall. This service is issued by the Management Node back to the client, in response to an install request from the client. It is not required to install the Non-trusted nodes through the firewall or to apply maintenance. This is the reason why the service is considered optional.

- tftp-tcp

Needed to install Linux nodes. This service defines the protocol for tcp.

- tftp-udp

Needed to install Linux nodes. This service defines the protocol for udp.

- www-tcp

Needed to use World Wide Web http. This service defines the protocol for tcp.

- www-udp

Needed to use World Wide Web http. This service defines the protocol for udp.

- kerberos

Kerberos Version 5 KDC. Needed if running Kerberos Version 5 remote command authentication. This service defines the protocol for tcp.

- kerberos

Kerberos Version 5 KDC. Needed if running Kerberos Version 5 remote command authentication. This service defines the protocol for udp.

- sunrpc-udp

The portmapper service. Needed when installing a Non-trusted node through the firewall. Specifically required mount request that takes place during node install.

- shell

Used when rsh/rcp is enabled for Standard (std) authentication protocol. Needed for xdsh operations when using rsh for remote commands.

- rsyslogd

Used for system log monitoring. This is for tcp protocol.

- rsyslogd  
Used for system log monitoring. This is for udp protocol.
- kshell  
Used rsh/rcp is enabled for Kerberos authentication. Not currently supported in xCAT. Network Installation Management client traffic generated by an Non-trusted node during node boot/shutdown. Required if using NIM. AIX only.
- rmc-tcp  
Resource Monitoring and Control (RMC) used for hardware monitoring, key exchange. This is for tcp protocol.
- rmc-udp  
Resource Monitoring and Control (RMC) used for hardware monitoring, key exchange. This is for udp protocol.
- conserver  
Required on the xCAT management node and service nodes. This service defines the protocol for tcp.
- nfsd-tcp  
Needed to use the AIX mount command. This service defines the protocol for tcp. Required when installing an Non-trusted node through the firewall. Needed when an installp is issued on an Non-trusted node and the resource exists on the Trusted side.
- nfsd-udp  
Needed to use the AIX mount command. This service defines the protocol for udp. Required when installing an Non-trusted node through the firewall.
- pxe  
Needed to install System x nodes through the firewall. This is the port for the PXE boot server. This service defines the protocol for tcp.
- rpc-mount  
Remote Procedure Call (RPM) used in conjunction with NFS mount request. See note 2. ssh-tcp Needed to use ssh. This service defines the protocol for tcp. This is required when installing or running updatenode through the firewall.
- mount-tcp  
Needed to use the AIX mount command. This service defines the protocol for tcp. Required when installing an Non-trusted node through the firewall. Needed when installp is issued on an Non-trusted node and the resource exists on the Trusted side. Needed to run updatenode command. See note 1.
- mount-udp  
Needed to use the AIX mount command. This service defines the protocol for udp. Needed when installp is issued on an Non-trusted node and the resource exists on the Trusted side. Needed to run updatenode command. See note 1.
- awk  
For awk communication during node discovery.
- impi  
For ipmi traffic.
- snmp  
For SNMP communication to blade chassis.



- snmptrap

For SNMP communication to blade chassis.

### Note 1 - AIX mount

On AIX, the mountd port range is usually determined at the time of the mount request. Part of the communication flow within a mount command is to query the remote mountd server and find out what ports it is using. The mountd ports are selected dynamically each time the mountd server is initialized. Therefore, the port numbers will vary from one boot to another, or when mountd is stopped and restarted.

Unfortunately, this causes a problem when used through a firewall, as no rule can be defined to handle traffic with a variable primary port. To create a service for mountd (server) traffic that has a fixed port, and one that can be trapped by a rule, you will need to update the /etc/services file on the host that is the target of the mount with new mountd entries for TCP and UDP, where the port numbers are known to be unused (free). The mountd TCP and UDP ports must be different. Any free port number is valid. The mountd must be stopped and started to pick up the new port values.

For example, issuing a mount request on Non-trusted node X, whose target is the Management Server, that is,

```
mount ms2112:/images /images
```

would require that the /etc/services file on ms2112 be updated with something similar to the following:

```
mountd 33333/tcp mountd 33334/udp
```

For mountd to detect its new port values you must stop and start rpc.mountd. The stopping and starting of mountd takes place on the same host where the /etc/services file mountd updates were made. In the above example, ms2112's mountd is stopped and started. You can verify that mountd is using the new port definitions by issuing the rpcinfo command.

This procedure shows how to change ports used by mountd:

```
lssrc -s rpc.mountd
```

Produces output similar to:

```
Subsystem Group PID Status rpc.mountd nfs 12404 active
```

Then

```
rpcinfo -p ms2112 | grep mount
```

Produces output similar to:

```
100005 1 udp 37395 mountd 100005 2 udp 37395 mountd 100005 3 udp 37395 mountd 100005 1.
↪tcp 34095 mountd 100005 2 tcp 34095 mountd 100005 3 tcp 34095 mountd
```

Then

```
stopsrc -s rpc.mount
```

Produces output similar to:

```
0513-044 The rpc.mountd Subsystem was requested to stop.
```

Update /etc/services with new mountd entries.

Note: Make a backup copy of /etc/services before making changes.

```
grep mountd /etc/services
```

Produces output similar to:

```
mountd 33333/tcp mountd 33334/udp
```

Then

```
startsrc -s rpc.mountd
```

Produces output similar to:

```
0513-059 The rpc.mountd Subsystem has been started. Subsystem PID is 19536.
```

Then

```
rpcinfo -p ms2112 | grep mount
```

Produces output similar to:

```
100005 1 udp 33334 mountd 100005 2 udp 33334 mountd 100005 3 udp 33334 mountd 100005 1_
↪tcp 33333 mountd 100005 2 tcp 33333 mountd 100005 3 tcp 33333 mountd
```

## Note 2

The rpc-mount service differs from the other service definitions in the following way. There is no associated protocol, because by definition it is UDP based. There is no source port.

## 1.5.16 xCAT probe

To help identify some of the common issues with xCAT, a new tool suite is now available **xCAT probe**.

You can use `xcatprobe -l` to list all valid subcommands, output will be as below

```
# xcatprobe -l
osdeploy          Probe operating system provision process. Supports two modes -
↪ 'Realtime monitor' and 'Replay history'.
xcatmn            After xcat installation, use this command to check if xcat has_
↪ been installed correctly and is
                  ready for use. Before using this command, install 'tftp',
↪ 'nslookup' and 'wget' commands.
switch-macmap     To retrieve MAC address mapping for the specified switch, or_
↪ all the switches defined in
                  'switches' table in xCAT db.
.....
```

## xcatmn

**xcatmn** can be used to check if xcat has been installed correctly and is ready for use.

**Note:** For several check items(eg. tftp service, dns service, http service), 'tftp', 'nslookup' and 'wget' are need. If not installed, a warning message will be displayed..

Command is as below

```
xcatprobe xcatmn -i <install_nic> [-V]
```

- **-i:** [Required] Specify the network interface name of provision network on management node.
- **-V:** Output more information for debug.

For example, run command on Management Node

```
xcatprobe xcatmn -i eth0
```

Output will be similar to:

```
# xcatprobe xcatmn -i eth0
[MN]: Sub process 'xcatd: SSL listener' is running
→ [ OK ]
[MN]: Sub process 'xcatd: DB Access' is running
→ [ OK ]
[MN]: Sub process 'xcatd: UDP listener' is running
→ [ OK ]
[MN]: Sub process 'xcatd: install monitor' is running
→ [ OK ]
[MN]: Sub process 'xcatd: Discovery worker' is running
→ [ OK ]
[MN]: Sub process 'xcatd: Command log writer' is running
→ [ OK ]
[MN]: xcatd is listening on port 3001
→ [ OK ]
[MN]: xcatd is listening on port 3002
→ [ OK ]
[MN]: 'lsxcatd -a' works
→ [ OK ]
[MN]: The value of 'master' in 'site' table is an IP address
→ [ OK ]
[MN]: NIC enp0s1 exists on current server
→ [ OK ]
[MN]: Get IP address of NIC eth0
→ [ OK ]
[MN]: The IP *.*.*.* of eth0 equals the value of 'master' in 'site' table
→ [ OK ]
[MN]: IP *.*.*.* of NIC eth0 is a static IP on current server
→ [ OK ]
[MN]: *.*.*.* belongs to one of networks defined in 'networks' table
→ [ OK ]
[MN]: There is domain definition in 'site' table
→ [ OK ]
[MN]: There is a configuration in 'passwd' table for 'system' for node provisioning
→ [ OK ]
```

(continues on next page)

(continued from previous page)

```
[MN]: There is /install directory on current server
↳ [ OK ]
[MN]: There is /tftpboot directory on current server
↳ [ OK ]
[MN]: The free space of '/' is less than 12 G
↳ [ OK ]
[MN]: SELinux is disabled on current server
↳ [ OK ]
[MN]: Firewall is closed on current server
↳ [ OK ]
[MN]: HTTP service is ready on *.*.*.*
↳ [ OK ]
[MN]: TFTP service is ready on *.*.*.*
↳ [ OK ]
[MN]: DNS server is ready on *.*.*.*
↳ [ OK ]
[MN]: The size of /var/lib/dhcpd/dhcpd.leases is less than 100M
↳ [ OK ]
[MN]: DHCP service is ready on *.*.*.*
↳ [ OK ]
=====do summary=====
[MN]: Check on MN PASS.
↳ [ OK ]
```

[MN] means that the verification is performed on the Management Node. Overall status of PASS or FAILED will be displayed after all items are verified..

Service Nodes are checked automatically for hierarchical clusters.

For Service Nodes, the output will contain [SN:nodename] to distinguish different Service Nodes.

## detect\_dhcpd

**detect\_dhcpd** can be used to detect the dhcp server in a network for a specific mac address.

## image

## osdeploy

**osdeploy** operating system provision process. Supports two modes - 'Realtime monitor' and 'Replay history'.

Realtime monitor: This is a default. This tool with monitor provision state of the node. Trigger 'Realtime monitor' before rebooting target node to do provisioning.

Replay history: Used after provisioning is finished to probe the previously completed provisioning.

**Note:** Currently, hierarchical structure is not supported.

## Usage

```
xcatprobe osdeploy -h
xcatprobe osdeploy -n <node_range> [-t <max_waiting_time>] [-V]
xcatprobe osdeploy -n <node_range> -r <xxhxxm> [-V]
```

Options:

- **-n**: The range of nodes to be monitored or replayed.
- **-r**: Trigger ‘Replay history’ mode. Follow the duration of rolling back. Units are ‘h’ (hour) or ‘m’ (minute). If unit is not specified, hour will be used by default.
- **-t**: The maximum time to wait when doing monitor, unit is minutes. default is 60.
- **-V**: Output more information.

-r means replay history of OS provision, if no -r means to do realtime monitor.

## Realtime monitor

To monitor OS provisioning in real time, open at least 2 terminal windows. One to run osdeploy probe:

```
xcatprobe osdeploy -n cn1 [-V]
```

After some pre-checks, the probe will wait for provisioning information, similar to output below:

```
# xcatprobe osdeploy -n c910f03c17k20
The install NIC in current server is enp0s1
[INFO]
All nodes which will be deployed are valid
[ OK ]
-----
Start capturing every message during OS provision process.....
-----
```

Open second terminal window to run provisioning:

```
nodeset cn1 osimage=<osimage>
rpower cn1 boot
```

When all the nodes complete provisioning, the probe will exit and display output similar to:

```
# xcatprobe osdeploy -n c910f03c17k20
The install NIC in current server is enp0s1
[INFO]
All nodes which will be deployed are valid
[ OK ]
-----
Start capturing every message during OS provision process.....
-----

[c910f03c17k20] Use command reinstall to reboot node c910f03c17k20
[c910f03c17k20] Node status is changed to powering-on
[c910f03c17k20] Receive DHCPDISCOVER via enp0s1
```

(continues on next page)

(continued from previous page)

```
[c910f03c17k20] Send DHCP OFFER on 10.3.17.20 back to 42:d0:0a:03:11:14 via enp0s1
[c910f03c17k20] DHCPREQUEST for 10.3.17.20 (10.3.5.4) from 42:d0:0a:03:11:14 via enp0s1
[c910f03c17k20] Send DHCPACK on 10.3.17.20 back to 42:d0:0a:03:11:14 via enp0s1
[c910f03c17k20] Via TFTP download /boot/grub2/grub2-c910f03c17k20
[c910f03c17k20] Via TFTP download /boot/grub2/powerpc-ieee1275/normal.mod
.....
[c910f03c17k20] Postscript: otherpkgs exited with code 0
[c910f03c17k20] Node status is changed to booted
[c910f03c17k20] done
[c910f03c17k20] provision completed.(c910f03c17k20)
[c910f03c17k20] provision completed
↪ [ OK ]
All nodes specified to monitor, have finished OS provision process
↪ [ OK ]
=====osdeploy_probe_report=====
All nodes provisioned successfully
↪ [ OK ]
```

If there is something wrong when provisioning, this probe will exit when timeout is reached or Ctrl+C is pressed by user. The maximum time can be set by using `-t` as below(default 30 minutes)

```
xcatprobe osdeploy -n cn1 -t 30
```

## Replay history

To replay history of OS provision from 1 hour 20 minutes ago, use command as

```
xcatprobe osdeploy -n cn1 -r 1h20m
```

Output will be similar to:

```
# xcatprobe osdeploy -n c910f03c17k20
The install NIC in current server is enp0s1
↪ [INFO]
All nodes which will be deployed are valid
↪ [ OK ]
Start to scan logs which are later than *****, waiting for a while.....
=====osdeploy_probe_report=====
All nodes provisioned successfully
↪ [ OK ]
```

## discovery

**discovery** can be used to probe the discovery process, including pre-check for required configuration and realtime monitor of discovery process.

switch-macmap

nodecheck

osimagecheck

## 1.5.17 RAID

### Hardware RAID

#### Overview

In many new compute machines, disks have been formatted into RAID oriented format in manufacturer, so admin must create raid arrays using these disks manually before provisioning OS. How to configure raid arrays in unattended way for hundreds of machines turns to be a problem.

IBM has offered a tool `iprconfig` to configure raid for IBM power machine. To leverage this tool, xCAT enabled it in xCAT genesis.ppc64 so that admin can use `genesis.ppc64` to configure RAID arrays.

There are two commands (`diskdiscover` and `configraid`) shipped in `xCAT-genesis-scripts` package to support RAID arrays configuration using `iprconfig`, you can use the `runcmd` facility to configure raid in the hardware discovery procedure using them, `runcmd` is a facility which will be run in xcat genesis system. You can also use separated manual steps to use `configraid` in xcat genesis system shell.

- **diskdiscover** : Scan disk devices in xcat genesis system, give out disks and RAID arrays information.
- **configraid** : Delete RAID arrays, create RAID arrays in xcat genesis system.

Following sections show how to use `diskdiscover` and `configraid`, we assume `cn1` is compute node in all examples.

#### Discovering disk devices

Command `diskdiscover` scans disk devices, it can get the overview of disks and RAID arrays information from compute node; The outputs contain useful information for `configraid` to configure RAID arrays, user can get `pci_id`, `pci_slot_name`, `disk names`, `RAID arrays` and other information from the outputs. It should be ran in xcat genesis system. It can be executed without input parameter or with `pci_id`, `pci_id` includes PCI vendor and device ID. For example, power8 SAS adapter `pci_id` is `1014:034a`, `1014` is vendor info, `034a` is PCI-E IPR SAS Adapter, more info about `pci_id` refer to <http://pci-ids.ucw.cz/read/PC/1014/>.

Here are steps to use `diskdiscover`:

1. Start xCAT genesis system in compute node, let compute node `cn1` enter xCAT genesis system shell:

```
nodeset cn1 shell
rpower cn1 reset
```

Note: If user modify `diskdiscover` or `configraid` scripts, he needs to run the `mknb <arch>` command before `nodeset` command to update network boot root image.

2. On xcat management node, executing `xdsh` to use `diskdiscover`:

```
xdsh cn1 diskdiscover
```

Or:

```
xdsh cn1 'diskdiscover <pci_id>'
```

The outputs format is as following:

```
# xdsh cn1 diskdiscover
cn1: -----
cn1: PCI_ID      PCI_SLOT_NAME  Resource_Path  Device  Description  Status
cn1: -----
cn1: 1014:034a    0001:08:00.0   0:0:0:0       sg0     Function Disk Active
cn1: 1014:034a    0001:08:00.0   0:0:1:0       sg1     0 Array Member Active
cn1: -----
cn1: Get ipr RAID arrays by PCI_SLOT_NAME: 0001:08:00.0
cn1: -----
cn1: Name      PCI/SCSI Location      Description      Status
cn1: -----
cn1: sda      0001:08:00.0/0:2:0:0   RAID 0 Disk Array      Optimized
```

## Configuring hardware RAID

### Command configraid introduction

We can use configraid to delete RAID arrays or create RAID arrays:

```
configraid delete_raid=[all|"<raid_array_list>"|null]
                stripe_size=[16|64|256]
                create_raid="r1#<raidlevel>|[pci_id#<num>|pci_slot_name#<pci_slot_name>|disk_
↪names#<sg0>#.#.<sgn>]|disk_num#<number>" ...
```

Here are the input parameters introduction:

1. **delete\_raid** : List raid arrays which should be removed.
  - If its value is all, all raid arrays detected should be deleted.
  - If its value is a list of raid array names, these raid arrays will be deleted. Raid array names should be separated by #.
  - If its value is null or there is no delete\_raid, no raid array will be deleted.
  - If there is no delete\_raid, the default value is null.
2. **stripe\_size** : It is optional used when creating RAID arrays. If stripe size is not specified, it will default to the recommended stripe size for the selected RAID level.
3. **create\_raid** : To create a raid array, add a line beginning with create\_raid, all attributes keys and values are separated by #. The formats are as followings:
  - r1 means RAID level, RAID level can be any supported RAID level for the given adapter, such as 0, 10, 5, 6. r1 is a mandatory attribute for every create\_raid. Supported RAID level is depend on physical server's RAID adapter.
  - User can select disks based on following attributes value. User can find these value based on diskdiscover outputs as above section described.
    - a. pci\_id is PCI vendor and device ID.
    - b. pci\_slot\_name is the specified PCI location. If using pci\_slot\_name, this RAID array will be created using disks from it.
    - c. disk\_names is a list of advanced format disk names. If using disk\_names, this RAID array will be created using these disks.



- `disk_num` is the number of disks this RAID array will contain, default value is all unused disks in its pci slot.

More examples of input parameters:

1. Delete all original RAID arrays, create one RAID 10 array from pci\_id 1014:034a, it uses the first two available disks:

```
delete_raid=all create_raid="rl#0|pci_id#1014:034a|disk_num#2"
```

2. Delete original RAID arrays sda and sdb on compute node, create one RAID 0 array from pci slot 0001:08:00.0, its RAID level is 0, it uses first two disks:

```
delete_raid="sda#sdb" create_raid="rl#0|pci_slot_name#0001:08:00.0|disk_num#2"
```

3. Create one RAID array from pci\_id 1014:034a, RAID level is 0, stripe\_size is 256kb, using first two available disks:

```
stripe_size=256 create_raid="rl#0|pci_id#1014:034a|disk_num#2"
```

4. Create two RAID arrays, RAID level is 0, one array uses one disks from pci\_id 1014:034a, the other array uses two disks from pci\_slot\_name 0001:08:00.0:

```
create_raid="rl#0|pci_id#1014:034a|disk_num#1" create_raid="rl#0|pci_slot_name  
↪#0001:08:00.0|disk_num#2"
```

5. Create two RAID arrays, RAID level is 0, one array uses disks sg0 and sg1, the other array uses disks sg2 and sg3:

```
create_raid="rl#0|disk_names#sg0#sg1" create_raid="rl#0|disk_names#sg2#sg3"
```

## Configuring RAID arrays process

Command `configraid` is running in xcat genesis system, its log is saved under `/tmp` on compute node genesis system.

## Configuring RAID in hardware discovery procedure

1. Using `runcmd` facility to configure raid in the hardware discovery procedure, after configuring RAID, compute node enter xcat genesis system shell. In the following example, `configraid` deletes all original RAID arrays, it creates one RAID 0 array with first two disks from pci\_id 1014:034a:

```
nodeset cn1 runcmd="configraid delete_raid=all create_raid=rl#0|pci_id  
↪#1014:034a|disk_num#2",shell  
rpower cn1 reset
```

2. Using `rcons` to monitor the process:

```
rcons cn1
```

## Configuring RAID manually in xcat genesis system shell

1. Starting xCAT genesis system in compute node, let compute node cn1 enter xCAT genesis system shell:

```
nodeset cn1 shell
rpower cn1 reset
```

2. On xcat management node, executing xdsh to use configraid to configure RAID:

```
xdsh cn1 'configraid delete_raid=all create_raid="rl#0|pci_id#1014:034a|disk_num#2"'
```

## Monitoring and debugging RAID configuration process

1. Creating some RAID level arrays take very long time, for example, If user creates RAID 10, it will cost tens of minutes or hours. During this period, you can use xCAT xdsh command to monitor the progress of raid configuration.

```
xdsh cn1 iprconfig -c show-config
```

2. Logs for configraid is saved under tmp in compute node genesis system. User can login compute node and check configraid logs to debug.
3. When configuring RAID in hardware discovery procedure, user can use rcons command to monitor or debug the process:

```
rcons cn1
```

## 1.5.18 Rest API

xCAT provides REST API (also called a web services API) that is currently implemented as a front end (translation layer) to xcatd. This provides programmatic access to xCAT from any language. This document describes how to set it up and use it.

**NOTE:** This doc is working with xCAT 2.8.4 and later.

### Set Up Web Service for Rest API

The following steps describe how to setup the Web Service for xCAT management node to offer the Rest API service.

### Enable the HTTPS protocol for REST API

To improve the security between the REST API clients and server, enabling the secure transfer protocol (https) is the default configuration.

- [RHEL6/7/8 (x86\_64/ppc64/ppc64le) and RHEL5 (x86\_64)]

```
yum install mod_ssl
service httpd restart
yum install perl-JSON
```

- [RHEL5 (ppc64)]

Uninstall httpd.ppc64 and install httpd.ppc:

```
rpm -e --nodeps httpd.ppc64
rpm -i httpd.ppc mod_ssl.ppc
```

- [SLES10/11/12 (x86\_64/ppc64)]

```
a2enmod ssl
a2enflag SSL
/usr/bin/gensslcert
cp /etc/apache2/vhosts.d/vhost-ssl.template /etc/apache2/vhosts.d/vhost-ssl.conf
Insert line 'NameVirtualHost *:443' before the line '## SSL Virtual Host Context'
/etc/init.d/apache2 restart
zypper install perl-JSON
```

- [Ubuntu]

```
sudo a2enmod ssl
ln -s ../sites-available/default-ssl.conf /etc/apache2/sites-enabled/ssl.conf
sudo service apache2 restart

# verify it is loaded:

sudo apache2ctl -t -D DUMP_MODULES | grep ssl
apt-get install libjson-perl
```

**Note:** If use of non-secure HTTP protocol is required, edit /etc/httpd/conf.d/xcat-ws.conf for RHEL or /etc/apache2/conf.d/xcat-ws.conf for others and change RewriteEngine On to RewriteEngine Off, then restart httpd or apache.

## Enable the Certificate of HTTPs Server (Optional)

Enabling the certificate functionality of https server is useful for the Rest API client to authenticate the server.

The certificate for xcatd has already been generated when installing xCAT, it can be reused by the https server. To enable the server certificate authentication, the hostname of xCAT MN must be a fully qualified domain name (FQDN). The REST API client also must use this FQDN when accessing the https server. If the hostname of the xCAT MN is not a FQDN, you need to change the hostname first.

Typically the hostname of the xCAT MN is initially set to the NIC which faces to the cluster (usually an internal/private NIC). If you want to enable the REST API for public client, set the hostname of xCAT MN to one of the public NIC.

To change the hostname, edit /etc/sysconfig/network (RHEL) or /etc/HOSTNAME (SLES) and run:

```
hostname <newFQDN>
```

After changing the hostname, run the xcat command xcatconfig to generate a new server certificate based on the correct hostname:

```
xcatconfig -c
```

**Note:** If you had previously generated a certificate for non-root usersids to use xCAT, you must regenerate them using `/opt/xcat/share/xcat/scripts/setup-local-client.sh <username>`

---

The steps to configure the certificate for https server:

```
export sslcfgfile=/etc/httpd/conf.d/ssl.conf          # rhel
export sslcfgfile=/etc/apache2/vhosts.d/vhost-ssl.conf # sles
export sslcfgfile=/etc/apache2/sites-enabled/ssl.conf # ubuntu

sed -i 's/^\(s*\)SSLCertificateFile.*$/\1SSLCertificateFile \/etc\/xcat\/cert\/server-
↪cred.pem/' $sslcfgfile
sed -i 's/^\(s*\)SSLCertificateKeyFile.*$/\1/' $sslcfgfile

service httpd restart      # rhel
service apache2 restart    # sles/ubuntu
```

The REST API client needs to download the xCAT certificate CA from the xCAT http server to authenticate the certificate of the server.

```
cd /root
wget http://<xcat MN>/install/postscripts/ca/ca-cert.pem
```

When accessing the REST API, the certificate CA must be specified and the FQDN of the https server must be used. For example:

```
curl -X GET --cacert /root/ca-cert.pem 'https://<FQDN of xCAT MN>/xcatws/nodes?
↪userName=root&userPW=<root-pw>'
```

**Attention:** Some operations like ‘create osimage’ (i.e. copycds) may require a longer time to complete and may result in a “504 Gateway Timeout” error. To avoid this, modify the `httpd.conf` file and extend the timeout to a larger value: `Timeout: 600`

## Set Up an Account for Web Service Access

User needs a username and password to access the REST API. When the REST API request is passed to `xcatsd`, the username and password will be verified based on the *xCAT passwd Table*, and then `xcatsd` will look in the *xCAT policy Table* to see if the user is allowed to perform the requested operation.

The account with key of **xcats** will be used for the REST API authentication. The username and password should be passed in as the attributes of URL:

**userName**

Pass the username of the account

**userPW**

Pass the password of the account (xCAT 2.10)

**password**

Pass the password of the account (xCAT earlier than 2.10)

You can use the root userid for your API calls, but we recommend you create a new userid (for example `wsuser`) for the API calls and give it the specific privileges you want it to have.

## Use root Account

The certificate and ssh keys for **root** account has been created during the install of xCAT. The public ssh key also has been uploaded to compute node so that xCAT MN can ssh to CN without password. Then the only thing left to do is to add the password for the **root** in the passwd table.

```
tabch key=xcat,username=root passwd.password=<root-pw>
```

## Use non-root Account

Create new user and setup the password and policy rules.

```
# create a user
useradd -u <wsuid> <wsuser>
# set the password
passwd <wsuser>
# add password to passwd table
tabch key=xcat,username=<wsuser> passwd.password=<wspw>
# add user to policy table
mkdef -t policy 6 name=<wsuser> rule=allow
```

**Note:** Using the tabch command, you can use the salted password from /etc/shadow into the xCAT password table instead of a clear text password.

Identical user with the same name and uid need to be created on each compute node.

```
# create a user
useradd -u <wsuid> <wsuser>
# set the password
passwd <wsuser>
```

Create the SSL certificate under that user's home directory so that user can be authenticated to xCAT. This is done by running the following command on the Management node as root:

```
/opt/xcat/share/xcat/scripts/setup-local-client.sh <wsuser>
```

When running this command you'll see SSL certificates created. Enter "y" where prompted and take the defaults.

To enable the POST method of resources like nodeshell, nodecopy, updating and filesyncing for the non-root user, you need to enable the ssh communication between xCAT MN and CN without password. Log in as <username> and run following command:

```
xdsh <noderange> -K
```

Run a test request to see if everything is working:

```
curl -X GET --cacert /root/ca-cert.pem 'https://<xcat-mn-host>/xcatws/nodes?userName=
-><wsuser>&userPW=<wspw>'
```

or if you did not set up the certificate:

```
curl -X GET -k 'https://<xcat-mn-host>/xcatws/nodes?userName=<wsuser>&userPW=<wspw>'
```

You should see some output that includes your list of nodes.

If errors returned, check `/var/log/httpd/ssl_error_log` on xCAT MN.

---

**Note:** When passwords are changed, make sure to update the xCAT passwd table. The REST API service uses passwords stored there to authenticate users.

---

## How to Use xCAT Rest API

### The Resource Categories

The API lets you query, change, and manage the resources in following categories:

- Token Resources
- Node Resources
- Osimage Resources
- Network Resources
- Policy Resources
- Group Resources
- Global Configuration Resources
- Service Resources
- Table Resources

### The Authentication Methods for REST API

xCAT REST API supports two ways to authenticate the access user: user account (username + password) and access token (acquired by username + password).

#### User Account

Follow the steps in *WEB Service Setup*, you can create an account for yourself. Then use that username and password to access the https server.

The general format of the URL used in the REST API call is:

```
https://<FQDN of xCAT MN>/xcatws/<resource>?userName=<user>&userPW=<pw>&<parameters>
```

where:

- **FQDN of xCAT MN:** the hostname of the xCAT management node. It also can be the IP of xCAT MN if you don't want to enable the web server certificate
- **resource:** one of the xCAT resources listed above
- **user:** the userid that the operation should be run on behalf of. See the previous section on how to add/authorize a userid.

- **pw**: the password of the userid (can be the salted version from /etc/shadow)

Example:

```
curl -X GET -k 'https://<FQDN of xCAT MN>/xcatws/nodes?userName=root&userPW=cluster'
```

## Access Token

xCAT also supports the use the Access Token to replace the using of username+password in every access. Before accessing any resource, you need get a token with your account (username+password)

```
# curl -X POST -k \
  'https://<FQDN of xCAT MN>/xcatws/tokens?pretty=1' -H Content-Type:application/json -
  -data \
  '{"userName":"root","userPW":"cluster"}'
{
  "token":{
    "id":"5cabd675-bc2e-4318-b1d6-831fd1f32f97",
    "expire":"2014-3-10 9:56:12"
  }
}
```

Then in the subsequent REST API access, the token can be used to replace the user account (username+password)

```
curl -X GET -k -H X-Auth-Token:5cabd675-bc2e-4318-b1d6-831fd1f32f97 'https://<FQDN of
  xCAT MN>/xcatws/<resource>?<parameters>'
```

The default validity of a token is 1 day. This default can be changed by the setting of *tokenexpiredays* attribute in *site* table.

```
chdef -t site clustersite tokenexpiredays=<days>
```

To make tokens valid forever use “never”.

```
chdef -t site clustersite tokenexpiredays=never
```

If an old token has expired, you will get a ‘Authentication failure’ error. You will need to reacquire a token for your account.

## The Common Parameters for Resource URI

xCAT REST API supports several common parameters in the resource URI to enable specific output:

- **pretty=1** - It is used to format the json output for easier viewing on the screen.

```
https://<xCAT MN>/xcatws/nodes?pretty=1
```

- **debug=1** - It is used to display more debug messages for a REST API request.

```
https://<xCAT MN>/xcatws/nodes?debug=1
```

- **xcoll=1** - It is used to specify that the output should be grouped with the values of objects.

```
GET https://<xCAT MN>/xcatws/nodes/node1,node2,node3/power?xcoll=1
{
  "node2": {
    "power": "off"
  },
  "node1,node3": {
    "power": "on"
  }
}
```

Note: All the above parameters can be used together like following:

```
https://<xCAT MN>/xcatws/nodes?pretty=1&debug=1
```

## The Output of REST API request

xCAT REST API only supports the [JSON](<http://www.json.org/>) formatted output.

## When an Error occurs during the operation

(i.e. there's error/errorcode in the output of xcat xml response):

When error happens, for all the GET/PUT/POST/DELETE methods, the output will only include 'error' and 'errorcode' properties:

```
{
  error: [
    msg1,
    msg2,
    ...
  ],
  errorcode: error_number
}
```

## When NO Error occurs during the operation

(i.e. there's no error/errorcode in the output of xcat xml response):

## For the GET method

If the output can be grouped by the object (resource) name, and the information being returned are attributes of the object, then use the object name as the hash key and make the value be a hash of its attributes/values:

```
{
  object1: {
    a1: v1,
    a2: v2,
    ...
  },
```

(continues on next page)



(continued from previous page)

```
object2: {
  a1: v1,
  a2: v2,
  ...
},
}
```

If the output can be grouped by the object (resource) name, but the information being returned is **not** attributes of the object, then use the object name as the hash key and make the value be an array of strings:

```
{
  object1: [
    msg1,
    msg2,
    ...
  ],
  object2: [
    msg1,
    msg2
    ...
  ],
}
```

An example of this case is the output of `reventlog`:

```
{
  "node1": [
    "09/07/2013 10:05:02 Event Logging Disabled, Log Area Reset/Cleared (SEL Fullness)",
    ...
  ],
}
```

If the output is not object related, put all the output in a list (array):

```
[
  msg1,
  msg2,
  ...
]
```

## For the PUT/DELETE methods

There will be no output for operations that succeeded. (We made this decision because the output for them not formatted, and no program will read it if xcat indicates the operation has succeeded.)

## For POST methods

Since POST methods can either be creates or general actions, there is not much consistency. In the case of a create, the rule is the same as PUT/DELETE (no output if successful). For actions that have output that matters (e.g. nodeshell, filesyncing, sw, postscript), the rules are like the GET method.

## Testing the API

Normally you will make REST API calls from your code. You can use any language that has REST API bindings (most modern languages do).

## An Example of How to Use xCAT REST API from Python

Refer to the file `/opt/xcat/ws/xcatws-test.py`:

```
./xcatws-test.py --user wsuser -password cluster_rest --xcatmn <FQDN of xCAT MN>
```

## An Example of How to Use xCAT REST API from PERL

Refer to the file `/opt/xcat/ws/xcatws-test.pl`:

```
./xcatws-test.pl -m GET -u "https://127.0.0.1/xcatws/nodes?userName=root&userPW=cluster"
```

## An Example Script of How to Use curl to Test Your xCAT REST API Service

It can be used as an example script to access and control xCAT resources. From the output message, you also could get the idea of how to access xCAT resources.

```
/opt/xcat/ws/xcatws-test.sh
./xcatws-test.sh -u root -p cluster
./xcatws-test.sh -u root -p cluster -h <FQDN of xCAT MN>
./xcatws-test.sh -u root -p cluster -h <FQDN of xCAT MN> -c
./xcatws-test.sh -u root -p cluster -h <FQDN of xCAT MN> -t
./xcatws-test.sh -u root -p cluster -h <FQDN of xCAT MN> -c -t
```

But for exploration and experimentation, you can make API calls from your browser or by using the `curl` command.

To make an API call from your browser, use the desired URL from this document. To simplify the test step, all the examples for the resources use `curl -k` for unsecure http connection and use the 'username+password' to authenticate the user.

```
curl -X GET -k 'https://myserver/xcatws/nodes?userName=xxx&userPW=xxx&pretty=1'
```

## Examples of making an API call using curl

- To query resources:

```
curl -X GET -k 'https://xcatmnhost/xcatws/nodes?userName=xxx&userPW=xxx&pretty=1'
```

- To change attributes of resources:

```
curl -X PUT -k 'https://xcatmnhost/xcatws/nodes/{noderange}?userName=xxx&userPW=xxx'
-H Content-Type:application/json --data '{"room":"hi","unit":"7"}'
```

- To run an operation on a resource:

```
curl -X POST -k 'https://xcatmnhost/xcatws/nodes/{noderange}?userName=xxx&userPW=xxx'
-H Content-Type:application/json --data '{"groups":"wstest"}'
```

- To delete a resource:

```
curl -X DELETE -k 'https://xcatmnhost/xcatws/nodes/{noderange}?userName=xxx&userPW=xxx'
```

## Web Service Status Codes

Here are the HTTP defined status codes that the Web Service can return:

- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 406 Not Acceptable
- 408 Request Timeout
- 417 Expectation Failed
- 418 I'm a teapot
- 503 Service Unavailable
- 200 OK
- 201 Created

## References

- REST: [http://en.wikipedia.org/wiki/Representational\\_State\\_Transfer](http://en.wikipedia.org/wiki/Representational_State_Transfer)
- REST: <http://rest.elkstein.org/2008/02/what-is-rest.html>
- HTTP Status codes: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>
- HTTP Request Methods: <http://tools.ietf.org/html/rfc2616#section-9.1>
- HTTP Request Tool: <http://soft-net.net/SendHTTPTool.aspx> (haven't tried it yet)
- HTTP PATCH: <http://tools.ietf.org/html/rfc5789>
- HTTP BASIC Security: [http://httpd.apache.org/docs/2.2/mod/mod\\_auth\\_basic.html](http://httpd.apache.org/docs/2.2/mod/mod_auth_basic.html)
- Asynchronous Rest: <http://www.infoq.com/news/2009/07/AsynchronousRest>
- General JSON: <http://www.json.org/>
- JSON wrapping: <http://search.cpan.org/~makamaka/JSON-2.27/lib/JSON.pm>
- Apache CGI: <http://httpd.apache.org/docs/2.2/howto/cgi.html>

## xCAT Rest API Resources

### Token Resources

The URI list which can be used to create tokens for account .

#### [URI:/tokens] - The authentication token resource.

#### POST - Create a token.

##### Returns:

- An array of all the global configuration list.

##### Example:

Acquire a token for user 'root'.

```
curl -X POST -k 'https://127.0.0.1/xcatws/tokens?userName=root&userPW=cluster&pretty=1' -  
-H Content-Type:application/json --data '{"userName":"root","userPW":"cluster"}'  
{  
  "token":{  
    "id":"a6e89b59-2b23-429a-b3fe-d16807dd19eb",  
    "expire":"2014-3-8 14:55:0"  
  }  
}
```

## Node Resources

The URI list which can be used to create, query, change and manage node objects.

### [URI:/nodes] - The node list resource.

This resource can be used to display all the nodes which have been defined in the xCAT database.

#### GET - Get all the nodes in xCAT.

The attributes details for the node will not be displayed.

Refer to the man page: *lsdef*

##### Returns:

- Json format: An array of node names.

##### Example:

Get all the node names from xCAT database.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes?userName=root&userPW=cluster&pretty=1'
[
  "node1",
  "node2",
  "node3",
]
```

### [URI:/nodes/{noderange}] - The node resource

#### GET - Get all the attributes for the node {noderange}.

The keyword ALLRESOURCES can be used as {noderange} which means to get node attributes for all the nodes.

Refer to the man page: *lsdef*

##### Returns:

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ... }' pairs.

##### Example:

Get all the attributes for node 'node1'.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1?userName=root&userPW=cluster&
↳pretty=1'
{
  "node1":{
    "profile":"compute",
    "netboot":"xnba",
    "arch":"x86_64",
    "mgt":"ipmi",
    "groups":"all",
```

(continues on next page)

(continued from previous page)

```

    ...
  }
}

```

## PUT - Change the attributes for the node {noderange}.

Refer to the man page: *chdef*

### Parameters:

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {attr1:v1,attr2:v2,...}.

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Change the attributes mgt=dfm and netboot=yaboot.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/nodes/node1?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"mgt":"dfm","netboot":"yaboot"}'
```

## POST - Create the node {noderange}.

Refer to the man page: *mkdef*

### Parameters:

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {options:{opt1:v1,opt2:v2},attr1:v1,attr2:v2,...}.

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Create a node with attributes groups=all, mgt=dfm and netboot=yaboot

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node1?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"options":{"--template":"x86_64kvmguest-template"},"mgt":"dfm","netboot":"yaboot"}'
```

## DELETE - Remove the node {noderange}.

Refer to the man page: *rmdef*

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Delete the node node1

```
curl -X DELETE -k 'https://127.0.0.1/xcatws/nodes/node1?userName=root&userPW=cluster&pretty=1'
```

[URI:/nodes/{noderange}/attrs/{attr1,attr2,attr3 ...}] - The attributes resource for the node {noderange}

## GET - Get the specific attributes for the node {noderange}.

The keyword ALLRESOURCES can be used as {noderange} which means to get node attributes for all the nodes.

Refer to the man page: *lsdef*

### Returns:

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ...}' pairs.

### Example:

Get the attributes {groups,mgt,netboot} for node node1

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/attrs/groups,mgt,netboot?userName=root&userPW=cluster&pretty=1'
{
  "node1":{
    "netboot":"xnba",
    "mgt":"ipmi",
    "groups":"all"
  }
}
```

[URI:/nodes/{noderange}/host] - The mapping of ip and hostname for the node {noderange}

## POST - Create the mapping of ip and hostname record for the node {noderange}.

Refer to the man page: *makehosts*

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Create the mapping of ip and hostname record for node 'node1'.

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node1/host?userName=root&userPW=cluster&pretty=1'
```

### [URI:/nodes/{noderange}/dns] - The dns record resource for the node {noderange}

#### POST - Create the dns record for the node {noderange}.

The prerequisite of the POST operation is the mapping of ip and noderange for the node has been added in the /etc/hosts.

Refer to the man page: *makedns*

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Create the dns record for node 'node1'.

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node1/dns?userName=root&userPW=cluster&pretty=1'
```

#### DELETE - Remove the dns record for the node {noderange}.

Refer to the man page: *makedns*

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Delete the dns record for node node1

```
curl -X DELETE -k 'https://127.0.0.1/xcatws/nodes/node1/dns?userName=root&userPW=cluster&pretty=1'
```

### [URI:/nodes/{noderange}/dhcp] - The dhcp record resource for the node {noderange}

#### POST - Create the dhcp record for the node {noderange}.

Refer to the man page: *makedhcp*

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Create the dhcp record for node 'node1'.



```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node1/dhcp?userName=root&userPW=cluster&pretty=1'
```

### DELETE - Remove the dhcp record for the node {noderange}.

Refer to the man page: *makedhcp*

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Delete the dhcp record for node node1

```
curl -X DELETE -k 'https://127.0.0.1/xcatws/nodes/node1/dhcp?userName=root&userPW=cluster&pretty=1'
```

### [URI:/nodes/{noderange}/nodestat]] - The attributes resource for the node {noderange}

#### GET - Get the running status for the node {noderange}.

Refer to the man page: *nodestat*

#### Returns:

- An object which includes multiple entries like: <noderange> : { nodestat : <node state> }

#### Example:

Get the running status for node node1

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/nodestat?userName=root&userPW=cluster&pretty=1'
{
  "node1":{
    "nodestat":"noping"
  }
}
```

### [URI:/nodes/{noderange}/nodes]] - Lists the nodes, noderange cannot start with /

#### GET - Lists the nodes.

Refer to the man page: *nodes*

#### Returns:

- Json format: An array of node names.

#### Example:

Get the node names from xCAT database.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node[1-3]/nodes?userName=root&
↪userPW=cluster&pretty=1'
[
  "node1",
  "node2",
  "node3",
]
```

**[URI:/nodes/{noderange}/subnodes] - The sub-nodes resources for the node {noderange}**

**GET - Return the Children nodes for the node {noderange}.**

Refer to the man page: *rscan*

**Returns:**

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ...}' pairs.

**Example:**

Get all the children nodes for node 'node1'.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/subnodes?userName=root&
↪userPW=cluster&pretty=1'
{
  "cmm01node09":{
    "mpa":"ngpcmm01",
    "parent":"ngpcmm01",
    "serial":"1035CDB",
    "mtm":"789523X",
    "cons":"fsp",
    "hwtype":"blade",
    "objtype":"node",
    "groups":"blade,all,p260",
    "mgt":"fsp",
    "nodetype":"ppc,osi",
    "slotid":"9",
    "hcp":"10.1.9.9",
    "id":"1"
  },
  ...
}
```

[URI:/nodes/{noderange}/power] - The power resource for the node {noderange}

**GET - Get the power status for the node {noderange}.**

Refer to the man page: *rpower*

**Returns:**

- An object which includes multiple entries like: <nodename> : { power : <powerstate> }

**Example:**

Get the power status.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/power?userName=root&userPW=cluster&pretty=1'
{
  "node1": {
    "power": "on"
  }
}
```

**PUT - Change power status for the node {noderange}.**

Refer to the man page: *rpower*

**Parameters:**

- Json Formatted DataBody: {action:on/off/reset ...}.

**Returns:**

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

**Example:**

Change the power status to on

```
curl -X PUT -k 'https://127.0.0.1/xcatws/nodes/node1/power?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"action":"on"}'
```

[URI:/nodes/{noderange}/energy] - The energy resource for the node {noderange}

**GET - Get all the energy status for the node {noderange}.**

Refer to the man page: *renergy*

**Returns:**

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ...}' pairs.

**Example:**

Get all the energy attributes.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/energy?userName=root&userPW=cluster&pretty=1'
{
  "node1":{
    "cappingmin":"272.3 W",
    "cappingmax":"354.0 W"
    ...
  }
}
```

### PUT - Change energy attributes for the node {noderange}.

Refer to the man page: *renergy*

#### Parameters:

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {powerattr:value}.

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errnum}.

#### Example:

Turn on the cappingstatus to [on]

```
curl -X PUT -k 'https://127.0.0.1/xcatws/nodes/node1/energy?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"cappingstatus":"on"}'
```

[URI:/nodes/{noderange}/energy/{cappingmaxmin,cappingstatus,cappingvalue ...}] - The specific energy attributes resource for the node {noderange}

### GET - Get the specific energy attributes cappingmaxmin,cappingstatus,cappingvalue ... for the node {noderange}.

Refer to the man page: *renergy*

#### Returns:

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ...}' pairs.

#### Example:

Get the energy attributes which are specified in the URI.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/energy/cappingmaxmin,cappingstatus?userName=root&userPW=cluster&pretty=1'
{
  "node1":{
    "cappingmin":"272.3 W",
    "cappingmax":"354.0 W"
  }
}
```

[URI:/nodes/{noderange}/sp/{community|ip|netmask|...}] - The attribute resource of service processor for the node {noderange}

**GET - Get the specific attributes for service processor resource.**

Refer to the man page: *rspconfig*

**Returns:**

- Json format: An object which includes multiple ‘<name> : {attr:value, attr:value ...}’ pairs.

**Example:**

Get the snmp community for the service processor of node1.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/sp/community?userName=root&
↳userPW=cluster&pretty=1'
{
  "node1":{
    "SP SNMP Community":"public"
  }
}
```

**PUT - Change the specific attributes for the service processor resource.**

Refer to the man page: *rspconfig*

**Parameters:**

- Json format: An object which includes multiple ‘attr:value’ pairs. DataBody: {community:public}.

**Returns:**

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

**Example:**

Set the snmp community to [mycommunity].

```
curl -X PUT -k 'https://127.0.0.1/xcatws/nodes/node1/sp/community?userName=root&
↳userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"value":"mycommunity
↳"}'
```

[URI:/nodes/{noderange}/nextboot] - The temporary bootorder resource in next boot for the node {noderange}

**GET - Get the next bootorder.**

Refer to the man page: *rsetboot*

**Returns:**

- Json format: An object which includes multiple ‘<name> : {attr:value, attr:value ...}’ pairs.

**Example:**

Get the bootorder for the next boot. (It's only valid after setting.)

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/nextboot?userName=root&
↪userPW=cluster&pretty=1'
{
  "node1": {
    "nextboot": "Network"
  }
}
```

**PUT - Change the next boot order.**

Refer to the man page: *rsetboot*

**Parameters:**

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {order:net/hd}.

**Returns:**

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

**Example:**

Set the bootorder for the next boot.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/nodes/node1/nextboot?userName=root&
↪userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"order":"net"}'
```

**[URI:/nodes/{noderange}/bootstate] - The boot state resource for node {noderange}.****GET - Get boot state.**

Refer to the man page: *nodeset*

**Returns:**

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ...}' pairs.

**Example:**

Get the next boot state for the node1.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/bootstate?userName=root&
↪userPW=cluster&pretty=1'
{
  "node1": {
    "bootstat": "boot"
  }
}
```

## PUT - Set the boot state.

Refer to the man page: [nodeset](#)

### Parameters:

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {osimage:xxx}/{state:offline}.

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Set the next boot state for the node1.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/nodes/node1/bootstate?userName=root&
↪userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"osimage":"rhels6.4-
↪x86_64-install-compute"}'
```

## [URI:/nodes/{noderange}/vitals] - The vitals resources for the node {noderange}

### GET - Get all the vitals attributes.

Refer to the man page: [rvitals](#)

### Returns:

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ...}' pairs.

### Example:

Get all the vitals attributes for the node1.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/vitals?userName=root&userPW=cluster&
↪pretty=1'
{
  "node1":{
    "SysBrd Fault":"0",
    "CPUs":"0",
    "Fan 4A Tach":"3330 RPM",
    "Drive 15":"0",
    "SysBrd Vol Fault":"0",
    "nvDIMM Flash":"0",
    "Progress":"0"
    ...
  }
}
```

[URI:/nodes/{noderange}/vitals/{temp|voltage|wattage|fanspeed|power|leds...}] - The specific vital attributes for the node {noderange}

**GET - Get the specific vitals attributes.**

Refer to the man page: *rvitals*

**Returns:**

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ...}' pairs.

**Example:**

Get the 'fanspeed' vitals attribute.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/vitals/fanspeed?userName=root&
↪userPW=cluster&pretty=1'
{
  "node1":{
    "Fan 1A Tach":"3219 RPM",
    "Fan 4B Tach":"2688 RPM",
    "Fan 3B Tach":"2560 RPM",
    "Fan 4A Tach":"3330 RPM",
    "Fan 2A Tach":"3293 RPM",
    "Fan 1B Tach":"2592 RPM",
    "Fan 3A Tach":"3182 RPM",
    "Fan 2B Tach":"2592 RPM"
  }
}
```

[URI:/nodes/{noderange}/inventory] - The inventory attributes for the node {noderange}

**GET - Get all the inventory attributes.**

Refer to the man page: *rinv*

**Returns:**

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ...}' pairs.

**Example:**

Get all the inventory attributes for node1.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/inventory?userName=root&
↪userPW=cluster&pretty=1'
{
  "node1":{
    "DIMM 21 ":"8GB PC3-12800 (1600 MT/s) ECC RDIMM",
    "DIMM 1 Manufacturer":"Hyundai Electronics",
    "Power Supply 2 Board FRU Number":"94Y8105",
    "DIMM 9 Model":"HMT31GR7EFR4C-PB",
    "DIMM 8 Manufacture Location":"01",
    "DIMM 13 Manufacturer":"Hyundai Electronics",
    "DASD Backplane 4":"Not Present",

```

(continues on next page)



(continued from previous page)

```

    ...
  }
}

```

**[URI:/nodes/{noderange}/inventory/{pci|model...}] - The specific inventory attributes for the node {noderange}**

**GET - Get the specific inventory attributes.**

Refer to the man page: *rinv*

**Returns:**

- Json format: An object which includes multiple ‘<name> : {attr:value, attr:value ...}’ pairs.

**Example:**

Get the ‘model’ inventory attribute for node1.

```

curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/inventory/model?userName=root&
↪userPW=cluster&pretty=1'
{
  "node1":{
    "System Description":"System x3650 M4",
    "System Model/MTM":"7915C2A"
  }
}

```

**[URI:/nodes/{noderange}/eventlog] - The eventlog resource for the node {noderange}**

**GET - Get all the eventlog for the node {noderange}.**

Refer to the man page: *reventlog*

**Returns:**

- Json format: An object which includes multiple ‘<name> : {attr:value, attr:value ...}’ pairs.

**Example:**

Get all the eventlog for node1.

```

curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/eventlog?userName=root&
↪userPW=cluster&pretty=1'
{
  "node1":{
    "eventlog":[
      "03/19/2014 15:17:58 Event Logging Disabled, Log Area Reset/Cleared (SEL_
↪Fullness)"
    ]
  }
}

```

## DELETE - Clean up the event log for the node {noderange}.

Refer to the man page: *reventlog*

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Delete all the event log for node1.

```
curl -X DELETE -k 'https://127.0.0.1/xcatws/nodes/node1/eventlog?userName=root&
↪userPW=cluster&pretty=1'
[
  {
    "eventlog": [
      "SEL cleared"
    ],
    "name": "node1"
  }
]
```

## [URI:/nodes/{noderange}/beacon] - The beacon resource for the node {noderange}

## GET - Get the beacon status for the node {noderange}.

Refer to the man page: *rbeacon*

### Returns:

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ...}' pairs.

### Example:

Get beacon for node1.

```
curl -X GET -k 'https://127.0.0.1/xcatws/nodes/node1/beacon?userName=root&userPW=cluster&
↪pretty=1'
{
  "node1": {
    "beacon": [
      "Front:Blink Rear:Blink"
    ]
  }
}
```

## PUT - Change the beacon status for the node {noderange}.

Refer to the man page: *rbeacon*

### Parameters:

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {action:on/off/blink}.

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Turn on the beacon.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/nodes/node1/beacon?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"action":"on"}'
```

## [URI:/nodes/{noderange}/updating] - The updating resource for the node {noderange}

## POST - Update the node with file syncing, software maintenance and rerun postscripts.

Refer to the man page: *updatenode*

### Returns:

- An array of messages for performing the node updating.

### Example:

Initiate an updatenode process.

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node2/updating?userName=root&userPW=cluster&pretty=1'
[
  "There were no syncfiles defined to process. File synchronization has completed.",
  "Performing software maintenance operations. This could take a while, if there are
  packages to install.",
  "node2: Wed Mar 20 15:01:43 CST 2013 Running postscript: ospkgs",
  "node2: Running of postscripts has completed."
]
```

## [URI:/nodes/{noderange}/filesyncing] - The filesyncing resource for the node {noderange}

## POST - Sync files for the node {noderange}.

Refer to the man page: *updatenode*

### Returns:

- An array of messages for performing the file syncing for the node.

**Example:**

Initiate an file syncing process.

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node2/filesyncing?userName=root&
↪userPW=cluster&pretty=1'
[
  "There were no syncfiles defined to process. File synchronization has completed."
]
```

**[URI:/nodes/{noderange}/sw] - The software maintenance for the node {noderange}****POST - Perform the software maintenance process for the node {noderange}.**

Refer to the man page: *updatenode*

**Returns:**

- Json format: An object which includes multiple ‘<name> : {attr:value, attr:value ...}’ pairs.

**Example:**

Initiate an software maintenance process.

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node2/sw?userName=root&userPW=cluster&
↪pretty=1'
{
  "node2": [
    " Wed Apr  3 09:05:42 CST 2013 Running postscript: ospkgs",
    " Unable to read consumer identity",
    " Postscript: ospkgs exited with code 0",
    " Wed Apr  3 09:05:44 CST 2013 Running postscript: otherpkgs",
    " ./otherpkgs: no extra rpms to install",
    " Postscript: otherpkgs exited with code 0",
    " Running of Software Maintenance has completed."
  ]
}
```

**[URI:/nodes/{noderange}/postscript] - The postscript resource for the node {noderange}****POST - Run the postscripts for the node {noderange}.**

Refer to the man page: *updatenode*

**Parameters:**

- Json format: An object which includes multiple ‘attr:value’ pairs. DataBody: {scripts:[p1,p2,p3,...]}.

**Returns:**

- Json format: An object which includes multiple ‘<name> : {attr:value, attr:value ...}’ pairs.

**Example:**

Initiate an updatenode process.

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node2/postscript?userName=root&
↪userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"scripts":["syslog",
↪"remoteshell"]}'
{
  "node2": [
    " Wed Apr  3 09:01:33 CST 2013 Running postscript: syslog",
    " Shutting down system logger: [ OK ]",
    " Starting system logger: [ OK ]",
    " Postscript: syslog exited with code 0",
    " Wed Apr  3 09:01:33 CST 2013 Running postscript: remoteshell",
    " Stopping sshd: [ OK ]",
    " Starting sshd: [ OK ]",
    " Postscript: remoteshell exited with code 0",
    " Running of postscripts has completed."
  ]
}
```

**[URI:/nodes/{noderange}/nodeshell] - The nodeshell resource for the node {noderange}**

**POST - Run the command in the shell of the node {noderange}.**

Refer to the man page: *xdsh*

#### Parameters:

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: set environment {ENV:{en1:v1,en2:v2}}, raw command {raw:[op1,op2]}, direct command {command:[cmd1,cmd2]}.

#### Returns:

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ...}' pairs.

#### Example:

Run the 'date' command on the node2.

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node2/nodeshell?userName=root&
↪userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"command":["date",
↪"ls"]}'
{
  "node2": [
    " Wed Apr  3 08:30:26 CST 2013",
    " testline1",
    " testline2"
  ]
}
```

## [URI:/nodes/{noderange}/nodecopy] - The nodecopy resource for the node {noderange}

### POST - Copy files to the node {noderange}.

Refer to the man page: *xdcp*

#### Parameters:

- **Json format:** An object which includes multiple ‘attr:value’ pairs. **RequestBody:** {src:[file1,file2],target:dir}.

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Copy files /tmp/f1 and /tmp/f2 from xCAT MN to the node2:/tmp.

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node2/nodecopy?userName=root&
↪userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"src":["/tmp/f1","/
↪tmp/f2"],"target":"/tmp"}'
no output for succeeded copy.
```

## [URI:/nodes/{noderange}/vm] - The virtualization node {noderange}.

The node should be a virtual machine of type kvm, esxi ...

### PUT - Change the configuration for the virtual machine {noderange}.

Refer to the man page: *chvm*

#### Parameters:

- **Json format: An object which includes multiple ‘attr:value’ pairs. DataBody:**  
 Set memory size - {"memorysize":"sizeofmemory(MB)"} Add new disk - {"ad-  
 ddisk":"sizeofdisk1(GB),sizeofdisk2(GB)"} Purge disk - {"purgedisk":"scsi\_id1,scsi\_id2"}

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example1:

Set memory to 3000MB.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/nodes/node1/vm?userName=root&userPW=cluster&
↪pretty=1' -H Content-Type:application/json --data '{"memorysize":"3000"}'
```

#### Example2:

Add a new 20G disk.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/nodes/node1/vm?userName=root&userPW=cluster&
↪pretty=1' -H Content-Type:application/json --data '{"adddisk":"20G"}'
```

### Example3:

Purge the disk 'hdb'.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/nodes/node1/vm?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"purgedisk":"hdb"}'
```

### POST - Create the vm node {noderange}.

Refer to the man page: *mkvm*

#### Parameters:

- **Json format: An object which includes multiple 'attr:value' pairs. DataBody:**  
Set CPU count - {"cpucount":"numberofcpu"} Set memory size - {"memorysize":"sizeofmemory(MB)"}  
Set disk size - {"disksize":"sizeofdisk"} Do it by force - {"force":"yes"}

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Create the vm node1 with a 30G disk, 2048M memory and 2 cpus.

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node1/vm?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"disksize":"30G","memorysize":"2048","cpucount":"2"}'
```

### DELETE - Remove the vm node {noderange}.

Refer to the man page: *rmvm*

#### Parameters:

- **Json format: An object which includes multiple 'attr:value' pairs. DataBody:**  
Purge disk - {"purge":"yes"} Do it by force - {"force":"yes"}

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Remove the vm node1 by force and purge the disk.

```
curl -X DELETE -k 'https://127.0.0.1/xcatws/nodes/node1/vm?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"force":"yes","purge":"yes"}'
```

**[URI:/nodes/{noderange}/vmclone] - The clone resource for the virtual node {noderange}.**

The node should be a virtual machine of kvm, esxi ...

**POST - Create a clone master from node {noderange}. Or clone the node {noderange} from a clone master.**

Refer to the man page: *clonevm*

**Parameters:**

- **Json format: An object which includes multiple 'attr:value' pairs. DataBody:**  
 Clone a master named "mastername" - {"tomaster":"mastername"} Clone a node from master "mastername" - {"frommaster":"mastername"} Use Detach mode - {"detach":"yes"} Do it by force - {"force":"yes"}

**Returns:**

- The messages of creating Clone target.

**Example1:**

Create a clone master named "vmmaster" from the node1.

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node1/vmclone?userName=root&
↳userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"tomaster":"vmmaster
↳","detach":"yes"}'
{
  "node1":{
    "vmclone":"Cloning of node1.hda.qcow2 complete (clone uses 9633.19921875 for a
↳disk size of 30720MB)"
  }
}
```

**Example2:**

Clone the node1 from the clone master named "vmmaster".

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node1/vmclone?userName=root&
↳userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"frommaster":
↳"vmmaster"}'
```

**[URI:/nodes/{noderange}/vmmigrate] - The virtualization resource for migration.**

The node should be a virtual machine of kvm, esxi ...



**POST - Migrate a node to target node.**

Refer to the man page: *rmigrate*

**Parameters:**

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {"target": "targethost"}.

**Example:**

Migrate node1 to target host host2.

```
curl -X POST -k 'https://127.0.0.1/xcatws/nodes/node1/vmmigrate?userName=root&
↪userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"target":"host2"}'
```

**Oimage resources**

URI list which can be used to query, create oimage resources.

**[URI:/osimages] - The oimage resource.****GET - Get all the oimage in xCAT.**

Refer to the man page: *lsdef*

**Returns:**

- Json format: An array of oimage names.

**Example:**

Get all the oimage names.

```
curl -X GET -k 'https://127.0.0.1/xcatws/osimages?userName=root&userPW=cluster&pretty=1'
[
  "sles11.2-x86_64-install-compute",
  "sles11.2-x86_64-install-iscsi",
  "sles11.2-x86_64-install-iscsiibft",
  "sles11.2-x86_64-install-service"
]
```

**POST - Create the oimage resources base on the parameters specified in the Data body.**

Refer to the man page: *copycds*

**Parameters:**

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {iso:isofilename:filename,params:[{attr1:value1,attr2:value2}]}

**Returns:**

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

**Example1:**

Create osimage resources based on the ISO specified

```
curl -X POST -k 'https://127.0.0.1/xcatws/osimages?userName=root&userPW=cluster&pretty=1'
↪ -H Content-Type:application/json --data '{"iso":"/iso/RHEL6.4-20130130.0-Server-ppc64-DVD1.iso"}'
```

**Example2:**

Create osimage resources based on an xCAT image or configuration file

```
curl -X POST -k 'https://127.0.0.1/xcatws/osimages?userName=root&userPW=cluster&pretty=1'
↪ -H Content-Type:application/json --data '{"file":"/tmp/sles11.2-x86_64-install-compute.tgz"}'
```

**[URI:/osimages/{imgname}] - The osimage resource****GET - Get all the attributes for the osimage {imgname}.**

The keyword ALLRESOURCES can be used as {imgname} which means to get image attributes for all the osimages.

Refer to the man page: *lsdef*

**Returns:**

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ...}' pairs.

**Example:**

Get the attributes for the specified osimage.

```
curl -X GET -k 'https://127.0.0.1/xcatws/osimages/sles11.2-x86_64-install-compute?userName=root&userPW=cluster&pretty=1'
{
  "sles11.2-x86_64-install-compute":{
    "provmethod":"install",
    "profile":"compute",
    "template":"/opt/xcat/share/xcat/install/sles/compute.sles11.tmpl",
    "pkglist":"/opt/xcat/share/xcat/install/sles/compute.sles11.pkglist",
    "osvers":"sles11.2",
    "osarch":"x86_64",
    "osname":"Linux",
    "imagetype":"linux",
    "otherpkgdir":"/install/post/otherpkgs/sles11.2/x86_64",
    "osdistrname":"sles11.2-x86_64",
    "pkgdir":"/install/sles11.2/x86_64"
  }
}
```

## PUT - Change the attributes for the osimage {imgname}.

Refer to the man page: *chdef*

### Parameters:

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {attr1:v1,attr2:v2...}

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Change the 'osvers' and 'osarch' attributes for the osimage.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/osimages/sles11.2-ppc64-install-compute/?
↪userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{
↪"osvers":"sles11.3","osarch":"x86_64"}'
```

## POST - Create the osimage {imgname}.

Refer to the man page: *mkdef*

### Parameters:

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {attr1:v1,attr2:v2}

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Create a osimage obj with the specified parameters.

```
curl -X POST -k 'https://127.0.0.1/xcatws/osimages/sles11.3-ppc64-install-compute?
↪userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{
↪"osvers":"sles11.3","osarch":"ppc64","osname":"Linux","provmethod":"install","profile":
↪"compute"}'
```

## DELETE - Remove the osimage {imgname}.

Refer to the man page: *rmdef*

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Delete the specified osimage.

```
curl -X DELETE -k 'https://127.0.0.1/xcatws/osimages/sles11.3-ppc64-install-compute?
↪userName=root&userPW=cluster&pretty=1'
```

**[URI:/osimages/{imgname}/attrs/attr1,attr2,attr3 ...] - The attributes resource for the osimage {imgname}**

**GET - Get the specific attributes for the osimage {imgname}.**

The keyword ALLRESOURCES can be used as {imgname} which means to get image attributes for all the osimages.

Refer to the man page: *lsdef*

**Returns:**

- Json format: An array of attr:value pairs for the specified osimage.

**Example:**

Get the specified attributes.

```
curl -X GET -k 'https://127.0.0.1/xcatws/osimages/sles11.2-ppc64-install-compute/attrs/
↪imagetype,osarch,osname,provmethod?userName=root&userPW=cluster&pretty=1'
{
  "sles11.2-ppc64-install-compute":{
    "provmethod":"install",
    "osname":"Linux",
    "osarch":"ppc64",
    "imagetype":"linux"
  }
}
```

**[URI:/osimages/{imgname}/instance] - The instance for the osimage {imgname}**

**POST - Operate the instance of the osimage {imgname}.**

Refer to the man page: :doc:` </guides/admin-guides/references/>`

**Parameters:**

- Json format: An object which includes multiple 'attr:value' pairs.      **RequestBody:** {action:genpackexport,params:[{attr1:value1,attr2:value2...}]}

**Returns:**

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

**Example1:**

Generates a stateless image based on the specified osimage

```
curl -X POST -k 'https://127.0.0.1/xcatws/osimages/sles11.2-x86_64-install-compute/
↪instance?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --
↪data '{"action":"gen"}'
```

**Example2:**

Packs the stateless image from the chroot file system based on the specified osimage

```
curl -X POST -k 'https://127.0.0.1/xcatws/osimages/sles11.2-x86_64-install-compute/instance?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"action":"pack"}'
```

### Example3:

Exports an xCAT image based on the specified osimage

```
curl -X POST -k 'https://127.0.0.1/xcatws/osimages/sles11.2-x86_64-install-compute/instance?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"action":"export"}'
```

## DELETE - Delete the stateless or statelite image instance for the osimage {imgname} from the file system

Refer to the man page: *rmimage*

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Delete the stateless image for the specified osimage

```
curl -X DELETE -k 'https://127.0.0.1/xcatws/osimages/sles11.2-x86_64-install-compute/instance?userName=root&userPW=cluster&pretty=1'
```

## Network Resources

The URI list which can be used to create, query, change and manage network objects.

### [URI:/networks] - The network list resource.

This resource can be used to display all the networks which have been defined in the xCAT database.

### GET - Get all the networks in xCAT.

The attributes details for the networks will not be displayed.

Refer to the man page: *lsdef*

### Returns:

- Json format: An array of networks names.

### Example:

Get all the networks names from xCAT database.

```
curl -X GET -k 'https://127.0.0.1/xcatws/networks?userName=root&userPW=cluster&pretty=1'
[
  "network1",
  "network2",
  "network3",
]
```

## POST - Create the networks resources base on the network configuration on xCAT MN.

Refer to the man page: *makenetworks*

### Parameters:

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {attr1:v1,attr2:v2,...}.

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Create the networks resources base on the network configuration on xCAT MN.

```
curl -X POST -k 'https://127.0.0.1/xcatws/networks?userName=root&userPW=cluster&pretty=1'
```

## [URI:/networks/{netname}] - The network resource

### GET - Get all the attributes for the network {netname}.

The keyword ALLRESOURCES can be used as {netname} which means to get network attributes for all the networks.

Refer to the man page: *lsdef*

### Returns:

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ...}' pairs.

### Example:

Get all the attributes for network 'network1'.

```
curl -X GET -k 'https://127.0.0.1/xcatws/networks/network1?userName=root&userPW=cluster&pretty=1'
{
  "network1":{
    "gateway":"<xcatmaster>",
    "mask":"255.255.255.0",
    "mgtifname":"eth2",
    "net":"10.0.0.0",
    "tftpserver":"10.0.0.119",
    ...
  }
}
```

## PUT - Change the attributes for the network {netname}.

Refer to the man page: *chdef*

### Parameters:

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {attr1:v1,attr2:v2,...}.

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Change the attributes mgtifname=eth0 and net=10.1.0.0.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/networks/network1?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"mgtifname":"eth0","net":"10.1.0.0"}'
```

## POST - Create the network {netname}. DataBody: {attr1:v1,attr2:v2...}.

Refer to the man page: *mkdef*

### Parameters:

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {attr1:v1,attr2:v2,...}.

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Create a network with attributes gateway=10.1.0.1, mask=255.255.0.0

```
curl -X POST -k 'https://127.0.0.1/xcatws/networks/network1?userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"gateway":"10.1.0.1","mask":"255.255.0.0"}'
```

## DELETE - Remove the network {netname}.

Refer to the man page: *rmdef*

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Delete the network network1

```
curl -X DELETE -k 'https://127.0.0.1/xcatws/networks/network1?userName=root&userPW=cluster&pretty=1'
```

**[URI:/networks/{netname}/attrs/attr1,attr2,...] - The attributes resource for the network {netname}**

**GET - Get the specific attributes for the network {netname}.**

The keyword ALLRESOURCES can be used as {netname} which means to get network attributes for all the networks.

Refer to the man page: *lsdef*

**Returns:**

- Json format: An object which includes multiple ‘<name> : { attr:value, attr:value ... }’ pairs.

**Example:**

Get the attributes {groups,mgt,netboot} for network network1

```
curl -X GET -k 'https://127.0.0.1/xcatws/networks/network1/attrs/gateway,mask,mgtifname,
↪net,tftpserver?userName=root&userPW=cluster&pretty=1'
{
  "network1":{
    "gateway":"9.114.34.254",
    "mask":"255.255.255.0",
  }
}
```

## Policy Resources

The URI list which can be used to create, query, change and manage policy entries.

**[URI:/policy] - The policy resource.**

**GET - Get all the policies in xCAT.**

It will display all the policy resource.

Refer to the man page: *lsdef*

**Returns:**

- Json format: An object which includes multiple ‘<name> : { attr:value, attr:value ... }’ pairs.

**Example:**

Get all the policy objects.

```
curl -X GET -k 'https://127.0.0.1/xcatws/policy?userName=root&userPW=cluster&pretty=1'
[
  "1",
  "1.2",
  "2",
  "4.8"
]
```



## [URI:/policy/{policy\_priority}] - The policy resource

### GET - Get all the attributes for a policy {policy\_priority}.

It will display all the policy attributes for one policy resource.

The keyword ALLRESOURCES can be used as {policy\_priority} which means to get policy attributes for all the policies.

Refer to the man page: *lsdef*

#### Returns:

- Json format: An object which includes multiple ‘<name> : {attr:value, attr:value ... }’ pairs.

#### Example:

Get all the attribute for policy 1.

```
curl -X GET -k 'https://127.0.0.1/xcatws/policy/1?userName=root&userPW=cluster&pretty=1'
{
  "1":{
    "name":"root",
    "rule":"allow"
  }
}
```

### PUT - Change the attributes for the policy {policy\_priority}.

It will change one or more attributes for a policy.

Refer to the man page: *chdef*

#### Parameters:

- Json format: An object which includes multiple ‘attr:value’ pairs. DataBody: {attr1:v1,attr2:v2,... }.

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Set the name attribute for policy 3.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/policy/3?userName=root&userPW=cluster&pretty=1'
-H Content-Type:application/json --data '{"name":"root"}'
```

**POST - Create the policy {policyname}. DataBody: {attr1:v1,attr2:v2...}.**

It will create a new policy resource.

Refer to the man page: *chdef*

**Parameters:**

- Json format: An object which includes multiple ‘attr:value’ pairs. DataBody: {attr1:v1,attr2:v2,...}.

**Returns:**

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

**Example:**

Create a new policy 10.

```
curl -X POST -k 'https://127.0.0.1/xcatws/policy/10?userName=root&userPW=cluster&pretty=1'
↪ -H Content-Type:application/json --data '{"name":"root","commands":"rpower"}
```

**DELETE - Remove the policy {policy\_priority}.**

Remove one or more policy resource.

Refer to the man page: *rmdef*

**Returns:**

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

**Example:**

Delete the policy 10.

```
curl -X DELETE -k 'https://127.0.0.1/xcatws/policy/10?userName=root&userPW=cluster&pretty=1'
```

**[URI:/policy/{policyname}/attrs/{attr1,attr2,attr3,...}] - The attributes resource for the policy {policy\_priority}****GET - Get the specific attributes for the policy {policy\_priority}.**

It will get one or more attributes of a policy.

The keyword ALLRESOURCES can be used as {policy\_priority} which means to get policy attributes for all the policies.

Refer to the man page: *lsdef*

**Returns:**

- Json format: An object which includes multiple ‘<name> : {attr:value, attr:value ...}’ pairs.

**Example:**

Get the name and rule attributes for policy 1.

```
curl -X GET -k 'https://127.0.0.1/xcatws/policy/1/attrs/name,rule?userName=root&
↪userPW=cluster&pretty=1'
{
  "1":{
    "name":"root",
    "rule":"allow"
  }
}
```

## Group Resources

The URI list which can be used to create, query, change and manage group objects.

### [URI:/groups] - The group list resource.

This resource can be used to display all the groups which have been defined in the xCAT database.

#### GET - Get all the groups in xCAT.

The attributes details for the group will not be displayed.

Refer to the man page: *lsdef*

#### Returns:

- Json format: An array of group names.

#### Example:

Get all the group names from xCAT database.

```
curl -X GET -k 'https://127.0.0.1/xcatws/groups?userName=root&userPW=cluster&pretty=1'
[
  "__mgmtnode",
  "all",
  "compute",
  "ipmi",
  "kvm",
]
```

### [URI:/groups/{groupname}] - The group resource

#### GET - Get all the attributes for the group {groupname}.

Refer to the man page: *lsdef*

#### Returns:

- Json format: An object which includes multiple '<name> : {attr:value, attr:value ... }' pairs.

#### Example:

Get all the attributes for group 'all'.

```
curl -X GET -k 'https://127.0.0.1/xcatws/groups/all?userName=root&userPW=cluster&pretty=1'
↪ '{
  "all": {
    "members": "zxnode2,nodexxx,node1,node4"
  }
}
```

### PUT - Change the attributes for the group {groupname}.

Refer to the man page: *chdef*

#### Parameters:

- Json format: An object which includes multiple ‘attr:value’ pairs. DataBody: {attr1:v1,attr2:v2,... }.

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Change the attributes mgt=dfm and netboot=yaboot.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/groups/all?userName=root&userPW=cluster&pretty=1'
↪ -H Content-Type:application/json --data '{"mgt":"dfm","netboot":"yaboot"}'
```

### [URI:/groups/{groupname}/attrs/{attr1,attr2,attr3 ...}] - The attributes resource for the group {group-name}

### GET - Get the specific attributes for the group {groupname}.

Refer to the man page: *lsdef*

#### Returns:

- Json format: An object which includes multiple ‘<name> : {attr:value, attr:value ... }’ pairs.

#### Example:

Get the attributes {mgt,netboot} for group all

```
curl -X GET -k 'https://127.0.0.1/xcatws/groups/all/attrs/mgt,netboot?userName=root&userPW=cluster&pretty=1'
↪ '{
  "all": {
    "netboot": "yaboot",
    "mgt": "dfm"
  }
}
```

## Global Configuration Resources

The URI list which can be used to create, query, change global configuration.

### [URI:/globalconf] - The global configuration resource.

This resource can be used to display all the global configuration which have been defined in the xCAT database.

#### GET - Get all the xCAT global configuration.

It will display all the global attributes.

Refer to the man page: *lsdef*

##### Returns:

- Json format: An object which includes multiple ‘<name> : {attr:value, attr:value ...}’ pairs.

##### Example:

Get all the global configuration

```
curl -X GET -k 'https://127.0.0.1/xcatws/globalconf?userName=root&userPW=cluster&pretty=1'
{
  "clustersite":{
    "xcatconfdir":"/etc/xcat",
    "tftpdir":"/tftpboot",
    ...
  }
}
```

### [URI:/globalconf/attrs/{attr1,attr2 ...}] - The specific global configuration resource.

#### GET - Get the specific configuration in global.

It will display one or more global attributes.

Refer to the man page: *lsdef*

##### Returns:

- Json format: An object which includes multiple ‘<name> : {attr:value, attr:value ...}’ pairs.

##### Example:

Get the ‘master’ and ‘domain’ configuration.

```
curl -X GET -k 'https://127.0.0.1/xcatws/globalconf/attrs/master,domain?userName=root&userPW=cluster&pretty=1'
{
  "clustersite":{
    "domain":"cluster.com",
    "master":"192.168.1.15"
```

(continues on next page)

(continued from previous page)

```
}  
}
```

### PUT - Change the global attributes.

It can be used for changing/adding global attributes.

Refer to the man page: *chdef*

#### Parameters:

- Json format: An object which includes multiple 'attr:value' pairs. DataBody: {attr1:v1,attr2:v2,...}.

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Change/Add the domain attribute.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/globalconf/attrs/domain?userName=root&  
↪userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"domain":"cluster.  
↪com"}'
```

### DELETE - Remove the site attributes.

Used to remove one or more global attributes.

Refer to the man page: *chdef*

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Remove the domain configure.

```
curl -X DELETE -k 'https://127.0.0.1/xcatws/globalconf/attrs/domain?userName=root&  
↪userPW=cluster&pretty=1'
```

## Service Resources

The URI list which can be used to manage the host, dns and dhcp services on xCAT MN.

**[URI:/services/dns] - The dns service resource.****POST - Initialize the dns service.**

Refer to the man page: *makedns*

**Returns:**

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

**Example:**

Initialize the dns service.

```
curl -X POST -k 'https://127.0.0.1/xcatws/services/dns?userName=root&userPW=cluster&
↳pretty=1'
```

**[URI:/services/dhcp] - The dhcp service resource.****POST - Create the dhcpd.conf for all the networks which are defined in the xCAT Management Node.**

Refer to the man page: *makedhcp*

**Returns:**

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

**Example:**

Create the dhcpd.conf and restart the dhcpd.

```
curl -X POST -k 'https://127.0.0.1/xcatws/services/dhcp?userName=root&userPW=cluster&
↳pretty=1'
```

**[URI:/services/host] - The hostname resource.****POST - Create the ip/hostname records for all the nodes to /etc/hosts.**

Refer to the man page: *makehosts*

**Returns:**

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

**Example:**

Create the ip/hostname records for all the nodes to /etc/hosts.

```
curl -X POST -k 'https://127.0.0.1/xcatws/services/host?userName=root&userPW=cluster&
↳pretty=1'
```

## [URI:/services/slpmnodes] - The nodes which support SLP in the xCAT cluster

**GET** - Get all the nodes which support slp protocol in the network.

Refer to the man page: *lsslp*

### Returns:

- Json format: An object which includes multiple '<name> : { attr:value, attr:value ... }' pairs.

### Example:

Get all the nodes which support slp in the network.

```
curl -X GET -k 'https://127.0.0.1/xcatws/services/slpmnodes?userName=root&userPW=cluster&pretty=1'
{
  "ngpcmm01":{
    "mpa":"ngpcmm01",
    "otherinterfaces":"10.1.9.101",
    "serial":"100037A",
    "mtm":"789392X",
    "hwtype":"cmm",
    "side":"2",
    "objtype":"node",
    "nodetype":"mp",
    "groups":"cmm,all,cmm-zet",
    "mgt":"blade",
    "hidden":"0",
    "mac":"5c:f3:fc:25:da:99"
  },
  ...
}
```

## [URI:/services/slpmnodes/{CEC|FRAME|MM|IVM|RSA|HMC|CMM|IMM2|FSP...}] - The slp nodes with specific service type in the xCAT cluster

**GET** - Get all the nodes with specific slp service type in the network.

Refer to the man page: *lsslp*

### Returns:

- Json format: An object which includes multiple '<name> : { attr:value, attr:value ... }' pairs.

### Example:

Get all the CMM nodes which support slp in the network.

```
curl -X GET -k 'https://127.0.0.1/xcatws/services/slpmnodes/CMM?userName=root&userPW=cluster&pretty=1'
{
  "ngpcmm01":{
    "mpa":"ngpcmm01",
    "otherinterfaces":"10.1.9.101",
```

(continues on next page)



(continued from previous page)

```

    "serial": "100037A",
    "mtm": "789392X",
    "hwtype": "cmm",
    "side": "2",
    "objtype": "node",
    "nodetype": "mp",
    "groups": "cmm,all,cmm-zet",
    "mgt": "blade",
    "hidden": "0",
    "mac": "5c:f3:fc:25:da:99"
  },
  "Server--SNY014BG27A01K": {
    "mpa": "Server--SNY014BG27A01K",
    "otherinterfaces": "10.1.9.106",
    "serial": "100CF0A",
    "mtm": "789392X",
    "hwtype": "cmm",
    "side": "1",
    "objtype": "node",
    "nodetype": "mp",
    "groups": "cmm,all,cmm-zet",
    "mgt": "blade",
    "hidden": "0",
    "mac": "34:40:b5:df:0a:be"
  }
}

```

## Table Resources

URI list which can be used to create, query, change table entries.

### [URI:/tables/{tablelist}/nodes/{noderange}] - The node table resource

For a large number of nodes, this API call can be faster than using the corresponding nodes resource. The disadvantage is that you need to know the table names the attributes are stored in.

#### GET - Get attributes of tables for a noderange.

##### Returns:

- An object containing each table. Within each table object is an array of node objects containing the attributes.

##### Example1:

Get all the columns from table nodetype for node1 and node2.

```

curl -X GET -k 'https://127.0.0.1/xcatws/tables/nodetype/nodes/node1,node2?userName=root&
↪userPW=cluster&pretty=1'
{
  "nodetype": [

```

(continues on next page)

(continued from previous page)

```
{
  "provmethod":"rhels6.4-x86_64-install-compute",
  "profile":"compute",
  "arch":"x86_64",
  "name":"node1",
  "os":"rhels6.4"
},
{
  "provmethod":"rhels6.3-x86_64-install-compute",
  "profile":"compute",
  "arch":"x86_64",
  "name":"node2",
  "os":"rhels6.3"
}
]
```

### Example2:

Get all the columns from tables nodetype and noderes for node1 and node2.

```
curl -X GET -k 'https://127.0.0.1/xcatsw/tables/nodetype,noderes/nodes/node1,node2?
↪userName=root&userPW=cluster&pretty=1'
{
  "noderes":[
    {
      "installnic":"mac",
      "netboot":"xnba",
      "name":"node1",
      "nfsserver":"192.168.1.15"
    },
    {
      "installnic":"mac",
      "netboot":"pxe",
      "name":"node2",
      "proxydhcp":"no"
    }
  ],
  "nodetype":[
    {
      "provmethod":"rhels6.4-x86_64-install-compute",
      "profile":"compute",
      "arch":"x86_64",
      "name":"node1",
      "os":"rhels6.4"
    },
    {
      "provmethod":"rhels6.3-x86_64-install-compute",
      "profile":"compute",
      "arch":"x86_64",
      "name":"node2",
      "os":"rhels6.3"
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```
]
}
```

## PUT - Change the node table attributes for {noderange}.

### Parameters:

- A hash of table names and attribute objects. DataBody: {table1:{attr1:v1,attr2:v2,...}}.

### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

### Example:

Change the nodetype.arch and noderes.netboot attributes for nodes node1,node2.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/tables/nodetype,noderes/nodes/node1,node2?
↪userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{
↪"nodetype":{"arch":"x86_64"},"noderes":{"netboot":"xnba"}}'
```

## [URI:/tables/{tablelist}/nodes/nodes/{noderange}/{attrlist}] - The node table attributes resource

For a large number of nodes, this API call can be faster than using the corresponding nodes resource. The disadvantage is that you need to know the table names the attributes are stored in.

## GET - Get table attributes for a noderange.

### Returns:

- An object containing each table. Within each table object is an array of node objects containing the attributes.

### Example:

Get OS and ARCH attributes from nodetype table for node1 and node2.

```
curl -X GET -k 'https://127.0.0.1/xcatws/tables/nodetype/nodes/node1,node2/os,arch?
↪userName=root&userPW=cluster&pretty=1'
{
  "nodetype": [
    {
      "arch": "x86_64",
      "name": "node1",
      "os": "rhels6.4"
    },
    {
      "arch": "x86_64",
      "name": "node2",
      "os": "rhels6.3"
    }
  ]
}
```

## **[URI:/tables/{tablelist}/rows] - The non-node table resource**

Use this for tables that don't have node name as the key of the table, for example: passwd, site, networks, policy, etc.

### **GET - Get all rows from non-node tables.**

#### **Returns:**

- An object containing each table. Within each table object is an array of row objects containing the attributes.

#### **Example:**

Get all rows from networks table.

```
curl -X GET -k 'https://127.0.0.1/xcatws/tables/networks/rows?userName=root&
→userPW=cluster&pretty=1'
{
  "networks": [
    {
      "netname": "192_168_13_0-255_255_255_0",
      "gateway": "192.168.13.254",
      "staticrangeincrement": "1",
      "net": "192.168.13.0",
      "mask": "255.255.255.0"
    },
    {
      "netname": "192_168_12_0-255_255_255_0",
      "gateway": "192.168.12.254",
      "staticrangeincrement": "1",
      "net": "192.168.12.0",
      "mask": "255.255.255.0"
    }
  ]
}
```

## **[URI:/tables/{tablelist}/rows/{keys}] - The non-node table rows resource**

Use this for tables that don't have node name as the key of the table, for example: passwd, site, networks, policy, etc.

{keys} should be the name=value pairs which are used to search table. e.g. {keys} should be [net=192.168.1.0,mask=255.255.255.0] for networks table query since the net and mask are the keys of networks table.

### **GET - Get attributes for rows from non-node tables.**

#### **Returns:**

- An object containing each table. Within each table object is an array of row objects containing the attributes.

#### **Example:**

Get rows from networks table where net=192.168.1.0,mask=255.255.255.0.

```
curl -X GET -k 'https://127.0.0.1/xcatws/tables/networks/rows/net=192.168.1.0,mask=255.255.255.0?userName=root&userPW=cluster&pretty=1'
{
  "networks": [
    {
      "mgtifname": "eth0",
      "netname": "192_168_1_0-255_255_255_0",
      "tftpserver": "192.168.1.15",
      "gateway": "192.168.1.100",
      "staticrangeincrement": "1",
      "net": "192.168.1.0",
      "mask": "255.255.255.0"
    }
  ]
}
```

### PUT - Change the non-node table attributes for the row that matches the {keys}.

#### Parameters:

- A hash of attribute names and values. DataBody: {attr1:v1,attr2:v2,...}.

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Create a route row in the routes table.

```
curl -X PUT -k 'https://127.0.0.1/xcatws/tables/routes/rows/routename=privnet?
  →userName=root&userPW=cluster&pretty=1' -H Content-Type:application/json --data '{"net":
  →"10.0.1.0","mask":"255.255.255.0","gateway":"10.0.1.254","ifname":"eth1"}'
```

### DELETE - Delete rows from a non-node table that have the attribute values specified in {keys}.

#### Returns:

- No output when execution is successful. Otherwise output the error information in the Standard Error Format: {error:[msg1,msg2...],errorcode:errornum}.

#### Example:

Delete rows from routes table where routename=privnet.

```
curl -X DELETE -k 'https://127.0.0.1/xcatws/tables/routes/rows/routename=privnet?
  →userName=root&userPW=cluster&pretty=1'
```

## **[URI:/tables/{tablelist}/rows/{keys}/{attrlist}] - The non-node table attributes resource**

Use this for tables that don't have node name as the key of the table, for example: passwd, site, networks, policy, etc.

### **GET - Get specific attributes for rows from non-node tables.**

#### **Returns:**

- An object containing each table. Within each table object is an array of row objects containing the attributes.

#### **Example:**

Get attributes mgtifname and tftpserver from networks table for each row where net=192.168.1.0,mask=255.255.255.0.

```
curl -X GET -k 'https://127.0.0.1/xcatws/tables/networks/rows/net=192.168.1.0,mask=255.255.255.0/mgtifname,tftpserver?userName=root&userPW=cluster&pretty=1'
{
  "networks": [
    {
      "mgtifname": "eth0",
      "tftpserver": "192.168.1.15"
    }
  ]
}
```

## **1.5.19 Security**

The security of a system covers a wide range of elements, from the security of system deployment and configuration, to the security of the system management, and the security of the software that is running on the system. This document will only discuss security features that are currently implemented in xCAT, not address the OS or any additional software packages that might be installed.

### **OpenSSL Configuration**

xCAT does not ship OpenSSL RPMS nor does it statically link to any OpenSSL libraries. Communication between the xCAT client and daemon utilizes OpenSSL and the administrator can configure the SSL\_version and SSL\_cipher that should be used by xCAT daemons.

The configuration is stored in the xCAT site table using the site.xcatsslversion and site.xcatsslciphers attributes.

### **Configuration**

site.xcatsslversion is the SSL\_version option used by xcatd and passed to IO::Socket::SSL->start\_SSL(). See <https://metacpan.org/pod/IO::Socket::SSL> for more information. By default, xCAT ships with an empty value for site.xcatsslversion. In this case, xcatd will use SSLv23:!SSLv2:!SSLv3:!TLSv1 internally.

Here is an example of changing site.xcatsslversion to a different value, TLSv1\_2, for example.

```
chtab key=xcatsslversion site.value=TLSv1_2
```

If running > TLSv1, it is possible to disable insecure ciphers. Here's an example of one possible configuration:

```
"xcatssliphers", "kDH:kEDH:kRSA:!SSLv3:!SSLv2:!aNULL:!eNULL:!MEDIUM:!LOW:!MD5:!EXPORT:!  
↪CAMELLIA:!ECDH", ,
```

After making any changes to these configuration values, xcatd must be restarted:

```
service restart xcatd
```

If any mistakes have been made and communication is lost to xCAT, use XCATABYPASS to fix the bad configuration:

```
XCATABYPASS=1 tabedit site
```

## Validation

Use the openssl command to validate the SSL configuration is valid and expected.

- To check whether TLSv1 is supported by xcatd:

```
openssl s_client -connect 127.0.0.1:3001 -tls1
```

- To check if SSLv3 is disabled on xcatd:

```
openssl s_client -connect localhost:3001 -ssl3
```

You should get a response similar to:

```
70367087597568:error:14094410:SSL routines:SSL3_READ_BYTES:ssl3 alert handshake_  
↪failure:s3_pkt.c:1259:SSL alert number 40  
70367087597568:error:1409E0E5:SSL routines:SSL3_WRITE_BYTES:ssl handshake_  
↪failure:s3_pkt.c:598:
```

## Transmission Channel

The xCAT daemon uses SSL to only allow authorized users to run xCAT commands. All xCAT commands are initiated as an xCAT **client**, even when run commands from the xCAT management node. This **client** opens an SSL socket to the xCAT daemon, sends the command and receives responses through this one socket. xCAT has configured the certificate for root, if you need to authorize other users, refer to the section below.

## The SSL Certificates in xCAT

The xCAT daemon on the management node and service node listens on a SSL socket on port 3001, the communications on the SSL socket include:

1. the xCAT requests from xCAT Clients
2. the xCAT requests forwarded from other xCAT daemons, for example, the requests forwarded between xCAT daemons on management node and service nodes
3. some special xCAT requests from compute nodes, such as `getcredentials`, `getpostscript`, `litefile`, etc.

xCAT creates 1 CA certificate and 2 credentials (private key and certificate pairs):

1. xCAT CA certificate(ca.pem):

- a self-signed certificate used as Certificate Authority in xcatd SSL communication;
- generated by `/opt/xcat/share/xcat/scripts/setup-xcat-ca.sh` script on xCAT installation;
- **will be generated (or updated) on xCAT management node when:**
  - install or update xCAT when “`/etc/xcat/ca`” directory does not exist
  - or run `xcatconfig -f|--force`
  - or run `xcatconfig -c|--credentials`
- **files on management node:**
  - `/etc/xcat/ca/ca-cert.pem`
  - `/etc/xcat/cert/ca.pem` ,copied by `/opt/xcat/share/xcat/scripts/setup-server-cert.sh`
  - `/root/.xcat/ca.pem` ,copied by `/opt/xcat/share/xcat/scripts/setup-local-client.sh`
- file on service node: `/root/.xcat/ca.pem`
- distribution path: `/etc/xcat/cert/ca.pem` (MN) ===(run `xcatconfig` command)====> `/install/postscripts/_xcat/ca.pem` (MN) ===(node provision/updatenode)====> `/xcatpost/_xcat/ca.pem` (SN and CN) ===(run “servicenode” postscript)====> `/root.xcat/ca.pem` (SN)

## 2. xCAT server credential(server-cred.pem):

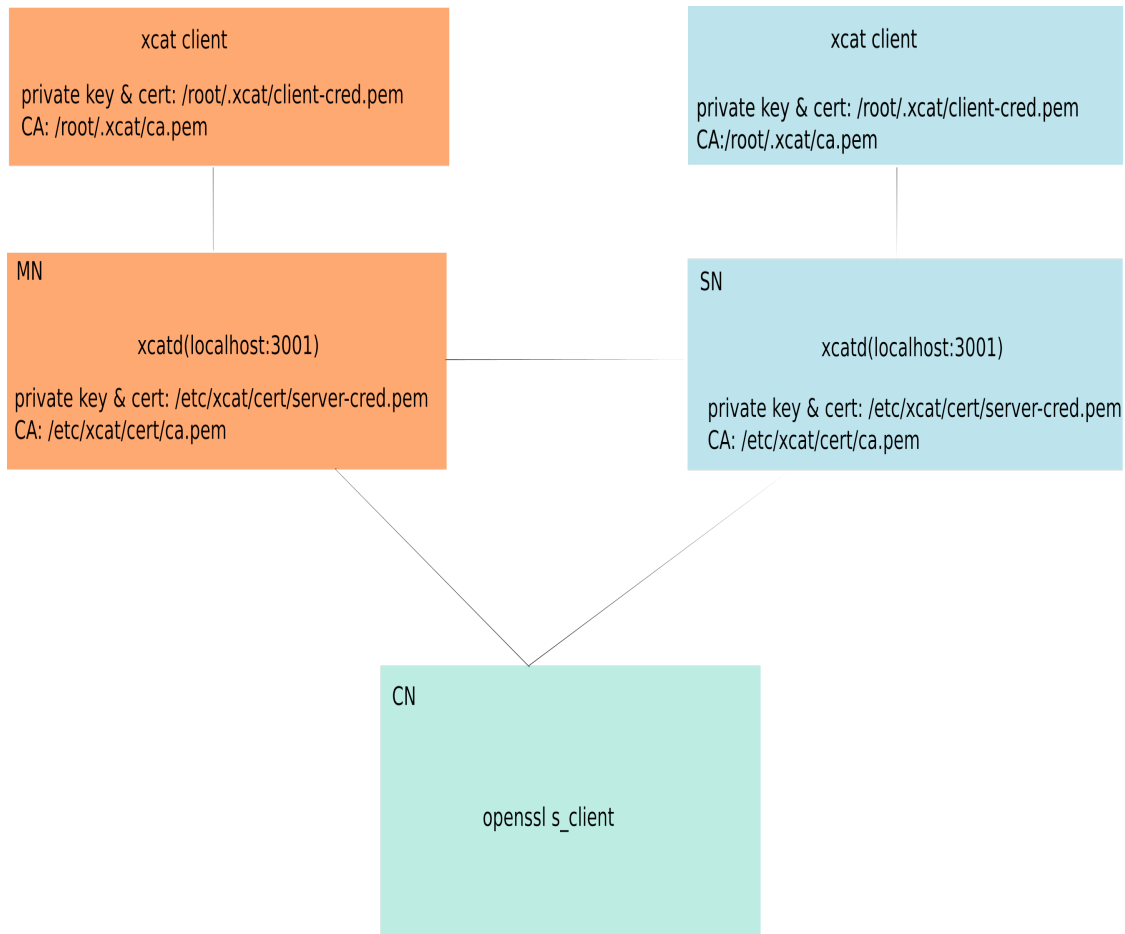
- a concatenation of server private key and certificate(signed with xCAT CA certificate)
- generated by `/opt/xcat/share/xcat/scripts/setup-server-cert.sh` on xCAT installation;
- **will be generated (or updated) on xCAT management node when:**
  - install or update xCAT when `/etc/xcat/cert` directory does not exist
  - or run `xcatconfig -f|--force`
  - or run `xcatconfig -c|--credentials`
- file on management node: `/etc/xcat/cert/server-cred.pem`
- file on service node: `/etc/xcat/cert/server-cred.pem`
- distribution path: `/etc/xcat/cert/server-cred.pem` (MN) ===(run `xcatserver` script called by servicenode postscript)====> `/etc/xcat/cert/server-cred.pem`(SN)

## 3. xCAT client credential(client-cred.pem):

- a concatenation of client private key and certificate (signed with xCAT CA certificate)
- generated by `/opt/xcat/share/xcat/scripts/setup-local-client.sh` on xCAT installation
- **will be generated (or updated) on xCAT management node when:**
  - install or update xCAT when `/root/.xcat/client-key.pem` does not exist;
  - or run `xcatconfig -f|--force`
  - or run `xcatconfig -c|--credentials`
- file on management node: `/root/.xcat/client-cred.pem`
- file on service node: `/root/.xcat/client-cred.pem`
- distribution path: `/root.xcat/client-cred.pem` (MN) ===(run `xcatclient` script called by servicenode postscript)====> `/root.xcat/client-cred.pem`(SN)



The usage of the credentials in the xCAT SSL communication is:



## Commands Access Control

Except SSL channel, xCAT only authorize root on the management node to run **xCAT** commands by default. But xCAT can be configured to allow both **non-root users** and **remote users** to run limited xCAT commands. For remote users, we mean the users who triggers the xCAT commands from other nodes and not have to login to the management node. xCAT uses the **policy** table to control who has authority to run specific xCAT commands. For a full explanation of the **policy** table, refer to [policy](#) man page.

## Granting Users xCAT Privileges

To give a non-root user all xCAT command privileges, run `tabedit policy` and add a line:

```
"6", "<username>", , , , , "allow", ,
```

Where `<username>` is the name of the user that you are granting privileges to. This user can now perform all xCAT commands, including changing the `policy` table to grant rights to other users, so this should be used with caution.

To grant a user ability to run `nodeIs` command:

```
"6", "<username>", , "nodeIs", , , "allow", ,
```

To grant all users the ability to run nodels:

```
"6.1", "*", "nodels", "allow",
```

CLI can also be used:

```
chdef -t policy -o 6.1 name=* commands=nodels rule=allow
```

**Note** Make sure the directories that contain the xCAT commands are in the user's \$PATH. If not, add them to \$PATH as appropriate in your system.

```
echo $PATH | grep xcat  
/opt/xcat/bin:/opt/xcat/sbin: .....
```

### Extra Setup for Remote Commands

To give a user the ability to run remote commands (xdsh, xdcp, psh, pcp) in some node, in addition to above steps, also need to run below steps:

```
su - <username>  
xdsh <noderange> -K
```

This will setup the user and root ssh keys for the user under the \$HOME/.ssh directory of the user on the nodes. The root ssh keys are needed for the user to run the xCAT commands under the xcatd daemon, where the user will be running as root. **Note:** the uid and the password for the user on the management node, should match the uid and password on the managed nodes.

### Set Up Login Node (Remote Client)

In some cases, you don't want your **non-root** user login to management node but still can run some xCAT commands. This time, you need setup a login node (i.e. remote client) for these users.

Below are the steps of how to set up a login node.

#### 1. Install the xCAT client

In order to avoid dependency problems on different distros, we recommend creating repository first by referring to links below.

- [Configure xCAT Software Repository in RHEL](#)
- [Configure the Base OS Repository in SUSE](#)
- [Configure the Base OS Repository in Ubuntu](#)

Then install xCAT-client.

**[RHEL]**

```
yum install xCAT-client
```

**[SUSE]**

```
zypper install xCAT-client
```

**[Ubuntu]**

```
apt-get install xCAT-client
```

## 2. Configure login node

When running on the login node, the environment variable **XCATHOST** must be export to the name or address of the management node and the port for connections (usually 3001).

```
export XCATHOST=<myManagmentServer>:3001
```

Using below command to add xCAT commands to your path.

```
source /etc/profile.d/xcat.sh
```

The userids and groupids of the non-root users should be kept the same on the login node, the management node, service nodes and compute nodes.

The remote not-root user still needs to set up the credentials for communication with management node. By running the `/opt/xcat/share/xcat/scripts/setup-local-client.sh <username>` command as root in management node, the credentials are generated in <username>'s `$HOME/.xcat` directory in management node. These credential files must be copied to the <username>'s `$HOME/.xcat` directory on the login node. **Note:** After scp, in the login node, you must make sure the owner of the credentials is <username>.

Setup your policy table on the management node with the permissions that you would like the non-root id to have.

At this time, the non-root id should be able to execute any commands that have been set in the policy table from the Login Node.

If any remote shell commands (psh,xdsh) are needed, then you need to follow [Extra Setup For Remote Commands](#).

## Auditing

xCAT logs all xCAT commands run by the xcatd daemon to both the syslog and the auditlog table in the xCAT database. The commands that are audited can be "ALL" xCAT commands or a list provided by the admin. The auditlog table allows the admin to monitor any attacks against the system or simply over use of resources. The auditlog table is store in the xCAT database and contains the following record.

```
# tabdump -d auditlog
recid:i      The record id.
audittime:   The timestamp for the audit entry.
userid:      The user running the command.
clientname:  The client machine, where the command originated.
clienttype:  Type of command: cli,java,webui,other.
command:     Command executed.
noderange:   The noderange on which the command was run.
args:        The command argument list.
status:      Allowed or Denied.
comments:    Any user-provided notes.
disable:     Do not use. tabprune will not work if set to yes or 1
```

## Password Management

xCAT is required to store passwords for various logons so that the application can login to the devices without having to prompt for a password. The issue is how to securely store these passwords.

Currently xCAT stores passwords in `passwd` table. You can store them as plain text, you can also store them as MD5 ciphertext.

Here is an example about how to store a MD5 encrypted password for root in `passwd` table.

```
tabch key=system passwd.username=root passwd.password=`openssl passwd -1 <password>`
```

During the provisioning, the root password will be set on the compute nodes. By default, xCAT stores the encrypted hash of password into installation files directly for better performance.

For example, `/etc/shadow` in stateless image for stateless nodes or installation files ( `/install/autoinst/<node>` ) for stateful nodes.

You can enable **secureroot** feature for more secure consideration.

```
chdef -t site secureroot=1
```

Then, after the new `packimage` or `nodeset` command, the root password hash can only be acquired on-the-fly with strict security control.

## Nodes Inter-Access in The Cluster

xCAT performs the setup for root to be able to ssh without password from the Management Node(MN) to all the nodes in the cluster. All nodes are able to ssh to each other without password or being prompted for a `known_host` entry, unless restricted. Nodes cannot ssh back to the Management Node or Service Nodes without a password by default.

xCAT generates, on the MN, a new set of ssh hostkeys for the nodes to share, which are distributed to all the nodes during install. If ssh keys do not already exist for root on the MN, it will generate an `id_rsa` public and private key pair.

During node install, xCAT sends the ssh hostkeys to `/etc/ssh` on the node, the `id_rsa` private key and `authorized_keys` file to root's `.ssh` directory on the node to allow root on the MN to ssh to the nodes without password. This key setup on the node allows the MN to ssh to the node with no password prompting.

On the MN and the nodes, xCAT sets the ssh configuration file to `strictHostKeyChecking no`, so that a `known_host` file does not have to be built in advanced. Each node can ssh to every other cluster node without being prompted for a password, and because they share the same ssh host keys there will be no prompting to add entries to `known_hosts`.

On the MN, you will be prompted to add entries to `known_hosts` file for each node once. See `makeknownhosts` command for a quick way to build a `known_hosts` file on the MN, if your nodes are defined in the xCAT database.

## Restricting Node to Node SSH

By default, all nodes installed by one management node are able to ssh to each without password. But there is an attribute `sshbetweennodes` in `site` table. This attribute defaults to `ALLGROUPS`, which means we setup ssh between all nodes during the install or when you run `xdsh -K`, or `updatenode -k` as in the past. This attribute can be used to define a comma-separated list of groups and only the nodes in those groups will be setup with ssh between the nodes. The attribute can be set to `NOGROUPS`, to indicate no nodes (groups) will be setup. Service Nodes will always be setup with ssh between service nodes and all nodes. It is unaffected by this attribute. This also only affects root userid setup and does not affect the setup of devices.

This setting of `site.sshbetweennodes` will only enable root ssh between nodes of the `compute1` and `compute 2` groups and all service nodes.

```
"sshbetweennodes", "compute1,compute2", ,
```

## Secure Zones

You can set up multiple zones in an xCAT cluster. A node in the zone can ssh without password to any other node in the zone, but not to nodes in other zones. Refer to [Zones](#) for more information.

## 1.5.20 Performance Tuning

xCAT supports clusters of all sizes. This document is a collection of hints, tips, and special considerations when working with large clusters, especially a single server (management node or service node) manages more than 128 nodes.

The information in this document should be viewed as example data only. Many of the suggestions are based on anecdotal experiences and may not apply to your particular environment. Suggestions in different sections of this document may recommend different or conflicting settings since they may have been provided by different people for different cluster environments. Often there is a significant amount of flexibility in most of these settings – you will need to resolve these differences in a way that works best for your cluster.

## System Tuning Settings for Linux

Adjusting Operating System tunables can improve large scale cluster performance, avoid bottlenecks, and prevent failures. The following sections are a collection of suggestions that have been gathered from various large scale HPC clusters. You should investigate and evaluate the validity of each suggestion before applying them to your cluster.

### 1. Tuning Linux ulimits:

The open file limits are important to high concurrence network services, such as `xcatd`. For a large cluster, it is required to increase the number of open file limit to avoid **Too many open files** error. The default value is `1024` in most OS distributions, to add below configuration in `/etc/security/limits.conf` to increase to `14096`.

```
* soft nofile 14096
* hard nofile 14096
```

### 2. Tuning Network kernel parameters:

There might be hundreds of hosts in a big network for large cluster, tuning the network kernel parameters for optimum throughput and latency could improve the performance of distributed application. For example, adding below configuration in `/etc/sysctl.conf` to increase the buffer size and queue length of **xCAT SSL listener** service access point ( port **3001** ).

```
net.core.rmem_max = 33554432
net.core.wmem_max = 33554432
net.core.rmem_default = 65536
net.core.wmem_default = 65536
net.core.somaxconn = 8192

net.ipv4.tcp_rmem = 4096 33554432 33554432
net.ipv4.tcp_wmem = 4096 33554432 33554432
net.ipv4.tcp_mem= 33554432 33554432 33554432
```

(continues on next page)

(continued from previous page)

```
net.ipv4.route.flush=1
net.core.netdev_max_backlog=1500
```

And if you encounter **Neighbour table overflow** error, it means there are too many ARP requests and the server cannot reply. Tune the ARP cache with below parameters.

```
net.ipv4.conf.all.arp_filter      = 1
net.ipv4.conf.all.rp_filter      = 1
net.ipv4.neigh.default.gc_thresh1 = 30000
net.ipv4.neigh.default.gc_thresh2 = 32000
net.ipv4.neigh.default.gc_thresh3 = 32768
net.ipv4.neigh.ib0.gc_stale_time  = 2000000
```

For more tunable parameters, you can refer to [Linux System Tuning Recommendations](#).

## Tuning xCAT Daemon Attributes

For large clusters, you consider changing the default settings in `site` table to improve the performance on a large-scale cluster or if you are experiencing timeouts or failures in these areas:

**consoleondemand** : When set to `yes`, `conserver` connects and creates the console output for a node only when the user explicitly opens the console using `rcons` or `wcons`. Default is `no` on Linux, `yes` on AIX. Setting this to `yes` can reduce the load `conserver` places on your xCAT management node. If you need this set to `no`, you may then need to consider setting up multiple servers to run the `conserver` daemon, and specify the correct server on a per-node basis by setting each node's `conserver` attribute.

**nodestatus** : If set to `n`, the `nodelist.status` column will not be updated during the node deployment, node discovery and power operations. Default is `y`, always update `nodelist.status`. Setting this to `n` for large clusters can eliminate one node-to-server contact and one xCAT database write operation for each node during node deployment, but you will then need to determine deployment status through some other means.

**precreatemypostscrip**ts : (`yes/1` or `no/0`, only for Linux). Default is `no`. If `yes`, it will instruct `xcat` at `nodeset` and `updatenode` time to query the database once for all of the nodes passed into the command and create the `myscript` file for each node, and put them in a directory in `site.tftpd`dir (such as: `/tftpboot`). This prevents `xcatd` from having to create the `myscript` files one at a time when each deploying node contacts it, so it will speed up the deployment process. (But it also means that if you change database values for these nodes, you must rerun `nodeset`.) If **precreatemypostscrip**ts is set to `no`, the `myscript` files will not be generated ahead of time. Instead they will be generated when each node is deployed.

**svloglocal** : if set to `1`, `syslog` on the service node will not get forwarded to the `mgmt` node. The default is to forward all `syslog` messages. The tradeoff on setting this attribute is reducing network traffic and log size versus having local management node access to all system messages from across the cluster.

**skiptables** : a comma separated list of tables to be skipped by `dumpxCATdb`. A recommended setting is `auditlog`, `eventlog` because these tables can grow very large. Default is to skip no tables.

**dhcplease** : The lease time for the DHCP client. The default value is `43200`.

**xcatmaxconnections** : Number of concurrent xCAT protocol requests before requests begin queueing. This applies to both client command requests and node requests, e.g. to get `postscrip`s. Default is `64`.

**xcatmaxbatchconnections** : Number of concurrent xCAT connections allowed from the nodes. Number must be less than **xcatmaxconnections**.

**useflowcontrol** : If `yes`, the `script` processing on each node contacts `xcatd` on the MN/SN using a lightweight UDP packet to wait until `xcatd` is ready to handle the requests associated with `postscrip`s. This prevents deploying nodes from flooding `xcatd` and locking out admin interactive use. This value works with the **xcatmaxconnections** and

**xcatmaxbatch** attributes. If the value is **no**, nodes sleep for a random time before contacting **xcatd**, and retry. The default is **no**. Not supported on AIX.

These attributes may be changed based on the size of your cluster. For a large cluster, it is better to enable **useflowcontrol** and set **xcatmaxconnection** = 356, **xcatmaxbatchconnections** = 300. Then the daemon will only allow 300 concurrent connections from the nodes. This will allow 56 connections still to be available on the management node for xCAT commands (e.g **node1s**).

## Tuning the Database Server

### 1. MariaDB database

MariaDB: [Tuning Server Parameters](#)

According to this documentation, the two most important variables to configure are **key\_buffer\_size** and **table\_open\_cache**.

### 2. PostgreSQL database

PostgreSQL: [Server Configuration](#)

## Tuning httpd for xCAT node deployments

In xCAT, the Operation System provisioning over network is heavily relying on the web server (Apache 2.x). However, Apache 2.x is a general-purpose web server, the default settings may not allow enough simultaneous HTTP client connections to support a large cluster.

### 1. Tuning MaxRequestWorkers directive

By default, httpd is configured to use **prefork** module for **MPM**, which has a limit of 256 simultaneous requests. If slow httpd response observed during OS provisioning, you can increase **MaxRequestWorkers** directive for better performance.

For example, to avoid some nodes provisioning failure when rebooting all nodes in a large hierarchy stateless cluster ( one service node is serving 270 compute nodes ), increase the value from 256 to 1000.

On Red Hat, change (or add) these directives in

```
/etc/httpd/conf/httpd.conf
```

On SLES (with Apache2), change (or add) these directives in

```
/etc/apache2/server-tuning.conf
```

### 1. Having httpd Cache the Files It Is Serving

**Note:** this information was contributed by Jonathan Dye and is provided here as an example. The details may have to be changed for distro or apache version.

This is simplest if you set **noderes.nfsserver** to a separate apache server, and then you can configure it to reverse proxy and cache. For some reason **mod\_mem\_cache** doesn't seem to behave as expected, so you can use **mod\_disk\_cache** to achieve a similar result: make a **tmpfs** on the apache server and configure its mountpoint to be the directory that **CacheRoot** points to. Also tell it to ignore **/install/autoinst** since the caching settings are really aggressive. Do a recursive **wget** to warm the cache and watch the **tmpfs** fill up. Then do a bunch of kickstart installs. Before this, the apache server on the xcat management node may have been a bottleneck during kickstart installs. After this change, it no longer should be.

Here is the apache config file:

```
ProxyRequests Off # don't be a proxy, just allow the reverse proxy

CacheIgnoreCacheControl On
CacheStoreNoStore On
CacheIgnoreNoLastMod On

CacheRoot /var/cache/apache2/tmpfs
CacheEnable disk /install
CacheDisable /install/autoinst
CacheMaxFileSize 1073741824

# CacheEnable mem / # failed attempt to do in-memory caching
# MCacheSize 20971520
# MCacheMaxObjectSize 524288000

# through ethernet network
# ProxyPass /install http://172.21.254.201/install

# through IB network
ProxyPass /install http://192.168.111.2/install
```

For more Apache 2.x tuning, see the external web page: [Apache Performance Tuning](#)

## 1.5.21 SoftLayer

Using xCAT in SoftLayer

## 1.5.22 System Clone (sysclone)

Using System Clone to Deploy Diskful Node

---

**Note:** This feature has been deprecated

---

When we want to deploy large numbers of nodes which have the same configuration, the simplest way is to clone. This means the user can customize and tweak one node's configuration according to his needs. They can verify it's proper operation, then make this node as template. They can capture an osimage from this template node, and deploy the rest of the nodes with this osimage quickly. xCat (2.8.2 and above) provides this feature which we call Sysclone to help you handle this scenario.



## List of Supported Arch and OS

xCAT version	OS	Tested Version	ARCH	Feature
2.8.2 and later	CentOS	6.3 5.9	x86_64	Basic clone node
	RHEL	6.4 5.9	x86_64	Basic clone node
2.8.3 and later	SLES	11.3 10.4	x86_64	Basic clone node
2.8.5 and later	CentOS	6.3	x86_64	Add feature: update delta changes(has limitation)
	RHEL	6.4	x86_64	Add feature: update delta changes(has limitation)
	SLES	11.3	x86_64	Add feature: update delta changes
	SLES	10.x	x86_64	Not support any more
2.9 and later	RHEL	6.4	ppc64	Basic clone node/update delta changes/LVM
	SLES	11.3	ppc64	Basic clone node/update delta changes
	RHEL	7.0	ppc64	Basic clone node/update delta changes/LVM
	RHEL	6.4 7.0	x86_64	support LVM

## Using Sysclone to Install Nodes

This document describes how to install and configure a template node (called golden client), capture an image from this template node, then using this image to deploy other nodes (called target nodes) quickly.

## Prepare the xCAT Management Node for Support Sysclone

To configure xCAT management node refer to section *Install Guides*

To support Sysclone, we need to install some extra rpms on management node and the golden client.

1. Download the xcat-dep tarball (xcat-dep-XXX.tar.bz2) which includes extra rpms needed by Sysclone. (You might already have the xcat-dep tarball on management node. If not, go to [xcat-dep](#) and get the latest xCAT dependency tarball.)
2. Install systemimager server on management node

- [RH/CentOS]:

```
yum -y install systemimager-server
```

- [SLES]:

```
zypper -n install systemimager-server
```

[Note] You may ignore the following messages when installing systemimager-server:

```
Can't locate AppConfig.pm in @INC (@INC contains: /usr/lib/systemimager/perl /usr/local/
↳ lib64/perl5 /usr/local/share/perl5 /usr/lib64/perl5/vendor_perl /usr/share/perl5/
↳ vendor_perl /usr/lib64/perl5 /usr/share/perl5 .) at /usr/lib/systemimager/perl/
↳ SystemImager/Config.pm line 13.
BEGIN failed--compilation aborted at /usr/lib/systemimager/perl/SystemImager/Config.pm
↳ line 13.
Compilation failed in require at /usr/lib/systemimager/perl/SystemImager/Server.pm line
↳ 17.
BEGIN failed--compilation aborted at /usr/lib/systemimager/perl/SystemImager/Server.pm
↳ line 17.
```

(continues on next page)

(continued from previous page)

```
Compilation failed in require at /usr/sbin/si_mkrsyncd_conf line 28.
BEGIN failed--compilation aborted at /usr/sbin/si_mkrsyncd_conf line 28.
```

3. Copy the xcat-dep-XXX.tar.bz2 file to directory /install/post/otherpkgs/<os>/<arch>/xcat/ of the management node according your golden client's OS version and system architecture, then decompress it:

- [CentOS6.3 and x86\_64]:

```
mkdir -p /install/post/otherpkgs/CentOS6.3/x86_64/xcat
cp xcat-dep-*.tar.bz2 /install/post/otherpkgs/CentOS6.3/x86_64/xcat
cd /install/post/otherpkgs/CentOS6.3/x86_64/xcat
tar jxvf xcat-dep-*.tar.bz2
```

- [SLES11.3 and x86\_64]:

```
mkdir -p /install/post/otherpkgs/SLES11.3/x86_64/xcat
cp xcat-dep-*.tar.bz2 /install/post/otherpkgs/SLES11.3/x86_64/xcat
cd /install/post/otherpkgs/SLES11.3/x86_64/xcat
tar jxvf xcat-dep-*.tar.bz2
```

- [RHEL6.4 and ppc64]:

```
mkdir -p /install/post/otherpkgs/rhels6.4/ppc64/xcat
cp xcat-dep-*.tar.bz2 /install/post/otherpkgs/rhels6.4/ppc64/xcat
cd /install/post/otherpkgs/rhels6.4/ppc64/xcat
tar jxvf xcat-dep-*.tar.bz2
```

## Install and Configure the Golden Client

The Golden Client acts as a regular node for xCAT, it just has some extra rpms to support clone. When you deploy golden client with xCAT, you just need to add a few additional definitions to the image which will be used to deploy golden client.

For information of how to install a regular node, refer to section *Diskful Installation*

To support clone, add 'otherpkglist' and 'otherpkgdir' attributes to the image definition which will be used to deploy golden client, then deploy golden client as normal. Once deployed, the golden client will have extra rpms to support cloning. If you have deployed your golden client already, use **updatenode** command to push these extra rpms to golden client. CentOS shares the same pkglist file with RHEL. For example:

- [RH6.4 and x86\_64]:

```
chdef -t osimage -o <osimage-name> otherpkglist=/opt/xcat/share/xcat/install/rh/
↪sysclone.rhels6.x86_64.otherpkgs.pkglist
chdef -t osimage -o <osimage-name> -p otherpkgdir=/install/post/otherpkgs/rhels6.4/
↪x86_64
updatenode <golden-client> -S
```

- [CentOS6.3 and x86\_64]:

```
chdef -t osimage -o <osimage-name> otherpkglist=/opt/xcat/share/xcat/install/rh/
↪sysclone.rhels6.x86_64.otherpkgs.pkglist
chdef -t osimage -o <osimage-name> -p otherpkgdir=/install/post/otherpkgs/CentOS6.3/
```

(continues on next page)

(continued from previous page)

```
↪x86_64
updatenode <golden-client> -S
```

- [SLES11.3 and x86\_64]:

```
chdef -t osimage -o <osimage-name> otherpkglist=/opt/xcat/share/xcat/install/sles/
↪sysclone.sles11.x86_64.otherpkgs.pkglist
chdef -t osimage -o <osimage-name> -p otherpkgdir=/install/post/otherpkgs/SLES11.3/
↪x86_64
updatenode <golden-client> -S
```

- [RH6.3 and ppc64]:

```
chdef -t osimage -o <osimage-name> otherpkglist=/opt/xcat/share/xcat/install/rh/
↪sysclone.rhels6.ppc64.otherpkgs.pkglist
chdef -t osimage -o <osimage-name> -p otherpkgdir=/install/post/otherpkgs/rhels6.3/
↪ppc64
updatenode <golden-client> -S
```

[Note]: If you install systemimager RPMs on CentOS 6.5 node by above steps, you maybe hit a failure. This is a known issue with CentOS6.5. Refer to known issue section for help.

## Capture Image from Golden Client

On Management node, use xCAT command **imgcapture** to capture an image from the golden-client.:

```
imgcapture <golden-client> -t sysclone -o <mycomputeimage>
```

When **imgcapture** is running, it pulls the image from the golden-client, and creates an image files system and a corresponding osimage definition on the xcat management node. You can use command below to check the osimage attributes.:

```
lsdef -t osimage <mycomputeimage>
```

## Install the target nodes with the image from the golden-client

To install the target nodes with the image captured from golden client.

- [x86\_64]:

```
nodeset <target-node> osimage=<mycomputeimage>
rsetboot <target-node> net
rpower <target-node> boot
```

- [ppc64]:

```
nodeset <target-node> osimage=<mycomputeimage>
rnetboot <target-node>
```

## Update Nodes Later On

If, at a later time, you need to make changes to the golden client (install new rpms, change config files, etc.), you can capture the changes and push them to the already cloned nodes without a need to restart cloned nodes. This process will only transfer the deltas, so it will be much faster than the original cloning.

**[Limitation]:** In xcat2.8.5, this feature has limitation on RHEL and CentOS. When your delta changes related boot-loader, it would encounter error. This issue will be fixed in xcat higher version. So up to now, in RHEL and CentOS, this feature just update files not related bootloader.

Update delta changes:

1. Make changes to your golden node (install new rpms, change config files, etc.).
2. From the mgmt node, capture the image using the same command as before. Assuming <myimagename> is an existing image, this will only sync the changes to the image on the Management node:

```
imgcapture <golden-client> -t sysclone -o <myimagename>
```

3. To synchronize the changes to your target nodes do the following:

- a) If you are running xCAT 2.8.4 or older:

From one of the nodes you want to update, test the update to see which files will be updated:

```
xdsh <target-node> -s 'si_updateclient --server <mgmtnode-ip> --dry-run --yes'
```

If it lists files and directories that you do not think should be updated, you need to add them to the exclude list in 3 places

- On the golden node: /etc/systemimager/updateclient.local.exclude
- On the mgmt node: /install/sysclone/images/<myimagename>/etc/systemimager/updateclient.local.exclude
- On all of the nodes to be updated: /etc/systemimager/updateclient.local.exclude

From the mgmt node, push the updates out to the other nodes:

```
xdsh <target-node-range> -s 'si_updateclient --server <mgmtnode-ip> --yes'
```

- b) If you are running xCAT 2.8.5 or later:

You could push the updates out to the other nodes quickly by below command:

```
updatenode <target-node-range> -S
```

## Known Issue

### Can not install systemimager RPMs in CentOS6.5 by yum

If you install systemimager RPMs on CentOS 6.5 node using **yum**, you may experience some problems due to CentOS6.5 itself. If that happens, copy related RPMs to CentOS 6.5 node and install them by hand.

- **On management node:**

```
[root@MN]# cd /<path-to-xcat-dep>/xcat-dep
[root@MN xcat-dep]# scp systemimager-client-4.3.0-0.1.noarch.rpm \
    systemconfigurator-2.2.11-1.noarch.rpm \
    systemimager-common-4.3.0-0.1.noarch.rpm \
    perl-AppConfig-1.52-4.noarch.rpm <CentOS-node-ip>:/
↪<savepath>
```

- On golden client:

```
[root@CentOS6.5 node]# cd /<savepath>
[root@CentOS6.5 node]# rpm -ivh perl-AppConfig-1.52-4.noarch.rpm
[root@CentOS6.5 node]# rpm -ivh systemconfigurator-2.2.11-1.noarch.rpm
[root@CentOS6.5 node]# rpm -ivh systemimager-common-4.3.0-0.1.noarch.rpm
[root@CentOS6.5 node]# rpm -ivh systemimager-client-4.3.0-0.1.noarch.rpm
```

## Kernel panic at times when install target node with rhels7.0 in Power 7 server

When you clone rhels7.0 image to target node which is Power 7 server lpar, you may hit Kernel panic problem at times after boot loader grub2 download kernel and initrd. This is an known issue but without a resolution. For now, we recommend you try again.

## 1.5.23 Zones

### Overview

XCAT supports the concept of zones within a single xCAT cluster managed by one Management Node. The nodes in the cluster can be divided up into multiple zones that have different ssh keys managed separately.

Each defined zone has it own root's ssh RSA keys, so that any node can ssh without a password to any other node in the same zone, cannot ssh without being prompted for a password to nodes in another zone.

Currently xCAT changes root ssh keys on the service nodes (SN) and compute nodes (CN) that are generated at install time to the root ssh keys from the Management node. It also changes the ssh **hostkeys** on the SN and CN to a set of pre-generated hostkeys from the MN. Putting the RSA public key in the **authorized-keys** file on the service nodes and compute nodes allows passwordless ssh to the Service Nodes (SN) and the compute nodes from the Management Node (MN). Today, by default, all nodes in the xCAT cluster are setup to be able to passwordless ssh to other nodes except when using the site **sshbetweennodes** attribute. More on that later. The pre-generated hostkey makes all nodes look like the same to ssh, so you are never prompted for updates to **known\_hosts**.

The new support only addresses the way we generate and distribute root's ssh RSA keys. Hostkey generation and distribution is not affected. It only supports setting up zones for the root userid. Non-root users are not affected. The Management node (MN) and Service Nodes (SN) are still setup so that root can ssh without password to the nodes from the MN and SN's for xCAT command to work. Also, the SN's should be able to ssh to each other with a password. Compute nodes and Service Nodes are not setup by xCAT to be able to ssh to the Management Node without being prompted for a password. This is to protect the Management Node.

In the past, the setup allowed compute nodes to be able to ssh to the SN's without a password. Using zones, will no longer allow this to happen. Using zones only allows compute nodes to ssh without password to compute node, unless you add the service node into the zone which is not considered a good idea.

But add service node into a zone is not a good idea. Because:

- IF you put the service node in a zone, it will no longer be able to ssh to the other servicenodes with being prompted for a password.

- Allowing the compute node to ssh to the service node, could allow the service node to be compromised, by anyone who gained access to the compute node.
- It is recommended to not put the service nodes in any zones and then they will use the default zone which today will assign the root's home directory ssh keys as in previous releases. More on the default zone later.

If you do not wish to use zones, your cluster will continue to work as before. The root ssh keys for the nodes will be taken from the Management node's root's home directory ssh keys or the Service node's root's home directory ssh keys (hierarchical case) and put on the nodes when installing, running `xdsh -K` or `updatenode -k`. To continue to operate this way, do not define a zone. The moment you define a zone in the database, you will begin using zones in xCAT.

## Configure Zones

Setting up zones only applies to nodes. We will still use the MN root ssh keys on any devices, switches, hardware control. All ssh access to these devices is done from the MN or SN. The commands that distribute keys to these entities will not recognize zones (e.g. `rspconfig`, `xdsh -K --devicetype`). You should never define, the Management Node in a zone. The zone commands will not allow this.

The ssh keys will be generated and store in `/etc/xcat/sshkeys/<zonenumber>/` .ssh directory. You must not change this path. XCAT will manage and sync this directory to the service nodes as need for hierarchy.

When using zones, the `site` table `sshbetweennodes` attribute is no longer used. You will get a warning that it is no longer used, if it is set. You can just remove the setting to get rid of the warning. The `zone` table `sshbetweennodes` attribute is used so this can be assigned for each zone. When using zones, the attribute can only be set to yes/no. Lists of nodegroups are not supported as was supported in the `site` table `sshbetweennodes` attributes. With the ability of creating zones, you should be able to setup your nodes groups to allow or not allow passwordless root ssh as before.

There are three commands to support zones:

- `mkzone` - creates the zones
- `chzone` - changes a previously created zone
- `rmzone` - removes a zone

**Note:** It is highly recommended that you only use the zone commands for creating and maintaining your zones. They do a lot of maintaining of tables and directories for the zones when they are running.

## Create Zones

The first time you run `mkzone`, it is going to create two zones. It will create the zone you request, but automatically add the xCAT default zone. This command creates the two zones, but does not assign it to any nodes. There is a new attribute on the nodes called `zonename`. As long as it is not defined for the node, then the node will use what is currently defined in the database as the defaultzone.

**Note:** if zones are defined in the zone table, there must be one and only one default zone. If a node does not have a zonename defined and there is no defaultzone in the zone table, it will get an error and no keys will be distribute.

For example:

```
#mkzone zone1

#lsdef -t zone -l
Object name: xcatdefault
  defaultzone=yes
  sshbetweennodes=yes
  sshkeydir=/root/.ssh
```

(continues on next page)

(continued from previous page)

```
Object name: zone1
defaultzone=no
sshbetweennodes=yes
sshkeydir=/etc/xcat/sshkeys/zone1/.ssh
```

Another example which makes the zone and defines the nodes in the mycompute group in the zone and also automatically creates a group on each node by the zonename is the following:

```
#makezone zone2 -a mycompute -g

#lsdef mycompute
Object name: node1
groups=zone2,mycompute
postbootscripts=otherpkgs
postscripts=syslog,remoteshell,syncfiles
zonename=zone2
```

At this time we have only created the zone, assigned the nodes and generated the SSH RSA keys to be distributed to the node. To setup the ssh keys on the nodes in the zone, run the following `updatenode` command. It will distribute the new keys to the nodes, it will automatically sync the zone key directory to any service nodes and it will regenerate your `mypostscript.<nodename>` files to include the zonename, if you are using `precreatemypostscripts` enabled.

```
updatenode mycompute -k
```

You can also use the following command but it will not regenerate the `mypostscript.<nodename>` file.

```
xdsh mycompute -K
```

If you need to install the nodes, then run the following commands. They will do everything during the install that the `updatenode` did. Running `nodeset` is very important, because it will regenerate the `mypostscript` file to include the `zonename` attribute.

```
nodeset mycompute osimage=<mycomputeimage>
rsetboot mycompute net
rpower mycompute boot
```

## Change Zones

After you create a zone, you can use the `chzone` command to make changes. Some of the things you can do are the following:

- Add nodes to the zone
- Remove nodes from the zone
- Regenerated the keys
- Change `sshbetweennodes` setting
- Make it the default zone

The following command will add `node1-node10` to `zone1` and create a group called `zone1` on each of the nodes.

```
chzone zone1 -a node1-node10 -g
```

The following command will remove `node20-node30` from `zone1` and remove the group `zone1` from those nodes.

```
chzone zone1 -r node2--node30 -g
```

The following command will change zone1 such that root cannot ssh between the nodes without entering a password.

```
#chzone zone1 -s no

#lsdef -t zone zone1
Object name: zone1
  defaultzone=no
  sshbetweennodes=no
  sshkeydir=/etc/xcat/sshkeys/zone1/.ssh
```

The following command will change zone1 to the default zone.

**Note:** you must use the `-f` flag to force the change. There can only be one default zone in the zone table.

```
#chzone zone1 -f --defaultzone

#lsdef -t zone -l
Object name: xcatdefault
  defaultzone=no
  sshbetweennodes=yes
  sshkeydir=/root/.ssh
Object name: zone1
  defaultzone=yes
  sshbetweennodes=no
  sshkeydir=/etc/xcat/sshkeys/zone1/.ssh
```

Finally, if your root ssh keys become corrupted or compromised you can regenerate them.

```
chzone zone1 -K
```

or

```
chzone zone1 -k <path to SSH RSH private key>
```

As with the `mkzone` commands, these commands have only changed the definitions in the database, you must run the following to distribute the keys.

```
updatenode mycompute -k
```

or

```
xdsh mycompute -K
```



## Remove Zones

The `rmzone` command will remove a zone from the database. It will also remove the zone name from the `zonename` attribute on all the nodes currently defined in the zone and as an option `-g` will remove the group `zonename` from the nodes. The `zonename` attribute will be undefined, which means the next time the keys are distributed, they will be picked up from the defaultzone. It will also remove the `/etc/xcatsshkeys/<zonename>` directory.

**Note:** `rmzone` will always remove the `zonename` defined on the nodes in the zone. If you use other xCAT commands and end up with a `zonename` defined on the node that is not defined in the zone table, when you try to distribute the keys you will get errors and the keys will not be distributed.

```
rmzone zone1 -g
```

If you want to remove the default zone, you must use the `-f` flag. You probably only need this to remove all the zones in the zone table. If you want to change the default zone, you should use the `chzone` command.

**Note:** if you remove the default zone and nodes have the `zonename` attribute undefined, you will get errors when you try to distribute keys.

```
rmzone zone1 -g -f
```

As with the other zone commands, after the location of a nodes root ssh keys has changed you should use one of the following commands to update the keys on the nodes:

```
updatenode mycompute -k
```

or

```
xdsh mycompute -K
```

## 1.5.24 xcat-inventory

`xcats-inventory` is an inventory tool for the cluster managed by xCAT. Its features includes:

- a object based view of the cluster inventory, which is flexible, extensible and well formatted
- interfaces to export/import the cluster inventory data in yaml/json format, which can be then managed under source control
- inventory templates for typical clusters, which help user to defines a cluster easily
- ability to integrate with Ansible(Coming Soon)

This section presents 2 typical user case of `xcats-inventory`

### Manage the xCAT Cluster Definition under Source Control

The xCAT cluster inventory data, including global configuration and object definitions(`node/osimage/passwd/policy/network/router`), and the relationship of the objects, can be exported to a YAML/JSON file(**inventory file**) from xCAT Database, or be imported to xCAT Database from the inventory file.

By managing the inventory file under source control system, you can manage the xCAT cluster definition under source control. This section presents a typical step-by-step scenario on how to manage cluster inventory data under `git`.

1. create a directory `/git/cluster` under `git` directory to hold the cluster inventory

```
mkdir -p /git/cluster
cd /git/cluster
git init
```

2. export the current cluster configuration to a inventory file “mycluster.yaml” under the git directory created above

```
xcat-inventory export --format=yaml >/git/cluster/mycluster.yaml
```

3. check diff and commit the cluster inventory file(commit no: c95673)

```
cd /git/cluster
git diff
git add /git/cluster/mycluster.yaml
git commit /git/cluster/mycluster.yaml -m "$(date +%Y_%m_%d_%H_%M_%S): initial_
↪cluster inventory data; blah-blah"
```

4. ordinary cluster maintenance and operation: replaced bad nodes, turn on xcatdebugmode...

5. cluster setup is stable now, export and commit the cluster configuration(commit no: c95673)

```
xcat-inventory export --format=yaml >/git/cluster/mycluster.yaml
cd /git/cluster
git diff
git add /git/cluster/mycluster.yaml
git commit /git/cluster/mycluster.yaml -m "$(date +%Y_%m_%d_%H_%M_%S): replaced_
↪bad nodes; turn on xcatdebugmode; blah-blah"
```

6. ordinary cluster maintenance and operation, some issues are founded in current cluster, need to restore the current cluster configuration to commit no c95673<sup>1</sup>

```
cd /git/cluster
git checkout c95673
xcat-inventory import -f /git/cluster/mycluster.yaml
```

*Notice:*

1. The cluster inventory data exported by xcat-inventory does not include intermediate data, transient data and historical data in xCAT DataBase, such as node status, auditlog table
2. We suggest you backup your xCAT database by dumpxCATdb before your trial on this feature, although we have run sufficient test on this

## Define and create your first xCAT cluster easily

The inventory templates for 2 kinds of typical xCAT cluster is shipped. You can create your first xCAT cluster easily by making several modifications on the template. The templates can be found under /opt/xcat/share/xcat/inventory\_templates on management node with xcat-inventory installed.

Currently, the inventory templates includes:

1. flat\_cluster\_template.yaml:

a flat baremetal cluster, including **openbmc controlled PowerLE servers, IPMI controlled Power servers(commented out), X86\_64 servers(commented out)**

<sup>1</sup> When you import the inventory data to xCAT Database in step 6, there are 2 modes: clean mode and update mode. If you choose the clean mode by xcat-inventory import -c|--clean, all the objects definitions that are not included in the inventory file will be removed; Otherwise, only the objects included in the inventory file will be updated or inserted. Please choose the proper mode according to your need

2. flat\_kvm\_cluster\_template.yaml: a flat KVM based Virtual Machine cluster, including **PowerKVM based VM nodes, KVM based X86\_64 VM nodes(commented out)**

The steps to create your first xCAT cluster is:

1. create a customized cluster inventory file “mycluster.yaml” based on flat\_cluster\_template.yaml

```
cp /opt/xcat/share/xcat/inventory_templates/flat_cluster_template.yaml /git/cluster/
↪mycluster.yaml
```

2. customize the cluster inventory file “mycluster.yaml” by modifying the attributes in the line under token #CHANGEME according to the setup of your physical cluster. You can create new node definition by duplicating and modifying the node definition in the template.

3. import the cluster inventory file

```
xcat-inventory import -f /git/cluster/mycluster.yaml
```

Now you have your 1st xCAT cluster, you can start bring up the cluster by provision nodes.

## 1.6 Questions & Answers

### 1.6.1 DNS, Hostname, Alias

**Q: When there are multiple NICs, how to generate /etc/hosts records?**

When there are multiple NICs, and you want to use confignetwork to configure these NICs, suggest to use hosts table to configure the installation NIC (installnic) and to use nics table to configure secondary NICs. Refer to the following example to generate /etc/hosts records.

**Best practice example:**

- There are 2 networks in different domains: mgtnetwork and pubnetwork
- mgtnetwork is xCAT management network
- There are 2 adapters in system node1: eth0 and eth1
- Add installnic eth0 10.5.106.101 record in /etc/hosts, its alias is mgtnic
- hostnames node1-pub and node1.public.com are for nic eth1, IP is 192.168.30.101

**Steps:**

1. Add networks entry in networks table:

```
chdef -t network mgtnetwork net=10.0.0.0 mask=255.0.0.0 domain=cluster.com
chdef -t network pubnetwork net=192.168.30.0 mask=255.255.255.0 domain=public.com
```

2. Create node1 with installnic IP 10.5.106.101, its alias is mgtnic:

```
chdef node1 ip=10.5.106.101 hostnames=mgtnic groups=all
```

3. Configure eth1 in nics table:

```
chdef node1 nicips.eth1=192.168.30.101 nichostnamesuffixes.eth1=-pub nicaliases.
↪eth1=node1.public.com nictypes.eth1=Ethernet nicnetworks.eth1=pubnetwork
```

4. Check node1 definition:

```
lsdef node1
  Object name: node1
  groups=all
  ip=10.5.106.101
  hostnames=mgtnic
  nicaliases.eth1=node1.public.com
  nichostnamesuffixes.eth1=-pub
  nicips.eth1=192.168.30.101
  nicnetworks.eth1=pubnetwork
  nictypes.eth1=Ethernet
  postbootscripts=otherpkgs
  postscripts=syslog,remoteshell,syncfiles
```

5. Execute `makehosts -n` to generate `/etc/hosts` records:

```
makehosts -n
```

6. Check results in `/etc/hosts`:

```
10.5.106.101 node1 node1.cluster.com mgtnic
192.168.30.101 node1-pub node1.public.com
```

7. Edit `/etc/resolv.conf`, xCAT management node IP like `10.5.106.2` is nameserver:

```
search cluster.com public.com
nameserver 10.5.106.2
```

8. Execute `makedns -n` to configure DNS

### Q: How to configure aliases?

There are 3 methods to configure aliases:

1. Use `hostnames` in `hosts` table to configure aliases for the installnic.
2. If you want to use script `confignetwork` to configure secondary NICs, suggest to use `aliases` in `nics` table to configure aliases. Refer to [Configure Aliases](#)
3. If you want to generate aliases records in `/etc/hosts` for secondary NICs and you don't want to use the script `confignetwork` to configure these NICs, suggest to use `otherinterfaces` in `hosts` table to configure aliases. Refer to following example:

- If you want to add `node1-hd 20.1.1.1` in `hosts` table, and don't use `confignetwork` to configure it, you can add `otherinterfaces` like this:

```
chdef node1 otherinterfaces="node1-hd:20.1.1.1"
```

- After executing `makehosts -n`, you can get records in `/etc/hosts` like following:

```
20.1.1.1 node1-hd
```

**Note:** If suffixes or aliases for the same IP are configured in both `hosts` table and `nics` table, will cause conflicts. `makehosts` will use values from `nics` table. The values from `nics` table will over-write that from `hosts` table to create `/etc/hosts` records.

**Q: How to handle the same short hostname in different domains?**

You can follow the best practice example.

**Best practice example:**

- There are 2 networks in different domains: `mgtnetwork` and `pubnetwork`
- `mgtnetwork` is xCAT management network
- Generate 2 records with the same hostname in `/etc/hosts`, like:

```
10.5.106.101 node1.cluster.com
192.168.20.101 node1.public.com
```

- Nameserver is xCAT management node IP

**Steps:**

1. Add networks entry in `networks` table:

```
chdef -t network mgtnetwork net=10.0.0.0 mask=255.0.0.0 domain=cluster.com
chdef -t network pubnetwork net=192.168.30.0 mask=255.255.255.0 domain=public.com
```

2. Create `node1` with `ip=10.5.106.101`, xCAT can manage and install this node:

```
chdef node1 ip=10.5.106.101 groups=all
```

3. Create `node1-pub` with `ip=192.168.30.101`, this node is only used to generate `/etc/hosts` records for public network, can use `_unmanaged` group name to label it:

```
chdef node1-pub ip=192.168.30.101 hostnames=node1.public.com groups=_unmanaged
```

4. Execute `makehosts -n` to generate `/etc/hosts` records:

```
makehosts -n
```

5. Check results in `/etc/hosts`:

```
10.5.106.101 node1 node1.cluster.com
192.168.30.101 node1-pub node1.public.com
```

6. Edit `/etc/resolv.conf`, for example, xCAT management node IP is `10.5.106.2`:

```
search cluster.com public.com
nameserver 10.5.106.2
```

7. Execute `makedns -n` to configure DNS

### Q: When to use `hosts` table and `nics` table?

`hosts` table is used to store IP addresses and hostnames of nodes. `makehosts` use these data to create `/etc/hosts` records. `nics` table is used to stores secondary NICs details. Some scripts like `confignetwork` use data from `nics` table to configure secondary NICs. `makehosts` also use these data to create `/etc/hosts` records for each NIC.

## 1.7 Reference Implementation

Reference documentation created for management of large clusters using xCAT.

### 1.7.1 CORAL

CORAL stands for Collaboration of Oak Ridge, Argonne, and Livermore and is solution that IBM is building for the Department of Energy to supersede its current cluster of Supercomputers.

#### Cluster Management

#### Scalability

#### Python framework

When testing the scale up of xCAT commands against OpenBMC REST API, it was evident that the Perl framework of xCAT did not scale well and was not sending commands to the BMCs in a true parallel fashion.

The team investigated the possibility of using Python framework. This support is implemented using Python 2.x framework for RHEL 7 and Python 3.x framework for RHEL 8.

#### Installation

A new RPM is created that contains the Python code: `xCAT-openbmc-py`. The Python code requires additional Python libraries that may not be available as an operating system provided package. The following will help resolve the dependencies.

#### Using RPM (recommended)

---

**Note:** Supported only on RHEL 7 and RHEL 8 for POWER9

---

---

**Note:** In a herarchical environment `xCAT-openbmc-py` must be installed on both Management and Service nodes. On Service node `xCAT-openbmc-py` can be installed directly by following instructions in **Install xCAT-openbmc-py on MN**, or `xCAT-openbmc-py` can be installed on Service node from Management node by following instructions in **Install xCAT-openbmc-py on SN from MN**

---

## Install xCAT-openbmc-py on RHEL 8 MN

1. Configure RHEL 8 EPEL repository

```
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

2. Install xCAT-openbmc-py

```
yum install xCAT-openbmc-py
```

## Install xCAT-openbmc-py on RHEL 8 SN from MN

**Attention:** Instructions below assume Service node has access to the Internet.

1. Choose one of the 2 methods below to complete the installation

### Install on diskful SN using updatenode or reinstall

1. Make the target repository directory on the MN:

```
mkdir -p /install/post/otherpkgs/rhels8.5.0/ppc64le/epel/Packages
```

2. Download the RHEL 8 EPEL rpm:

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -O /
↪install/post/otherpkgs/rhels8.5.0/ppc64le/epel/Packages/epel-release-latest-8.
↪noarch.rpm
```

3. Create a repository in that directory:

```
cd /install/post/otherpkgs/rhels8.5.0/ppc64le/epel/Packages/
createrepo .
```

4. Configure otherpkglist attribute of the osimage

```
chdef -t osimage rhels8.5.0-ppc64le-install-service otherpkglist=/opt/xcat/share/
↪xcat/install/rh/service.rhels8.ppc64le.otherpkgs.pkglist
```

5. Add the following entries to the contents of /opt/xcat/share/xcat/install/rh/service.rhels8.ppc64le.otherpkgs.pkglist

```
...
#NEW_INSTALL_LIST#
epel/Packages/epel-release-latest-8
#NEW_INSTALL_LIST#
xcat/xcat-core/xCAT-openbmc-py
```

6. If SN was initially installed without xCAT-openbmc-py package, updatenode can be used to install that package

```
updatenode <SN> -S
```

7. If this is a new SN installation

```
rinstall <SN> osimage=rhels8.5.0-ppc64le-install-service
```

### Install on diskless SN using rinstall

1. Add EPEL online repository <https://dl.fedoraproject.org/pub/epel/8/ppc64le> to `pkgdir` attribute of existing diskless RHEL 8 service osimage:

```
chdef -t osimage -o rhels8.5.0-ppc64le-netboot-service -p pkgdir=https://dl.fedoraproject.org/pub/epel/8/Everything/ppc64le
```

2. Configure `otherpkglist` attribute of the osimage

```
chdef -t osimage rhels8.5.0-ppc64le-netboot-service otherpkglist=/opt/xcat/share/xcat/install/rh/service.rhels8.ppc64le.otherpkgs.pkglist
```

3. Add the following entries to the contents of `/opt/xcat/share/xcat/install/rh/service.rhels8.ppc64le.otherpkgs.pkglist`

```
...  
#NEW_INSTALL_LIST#  
xcat/xcat-core/xCAT-openbmc-py
```

4. Install diskless SN

```
genimage rhels8.5.0-ppc64le-netboot-service  
packimage rhels8.5.0-ppc64le-netboot-service  
rinstall <SN> osimage=rhels8.5.0-ppc64le-netboot-service
```

### Install xCAT-openbmc-py on RHEL 7 MN

The following repositories should be configured on your Management Node.

- RHEL 7.5 OS repository
- RHEL 7.5 Extras repository
- RHEL 7 EPEL repository (<https://fedoraproject.org/wiki/EPEL>)
- Fedora28 repository (for `gevent` and `greenlet`)

1. Configure RHEL 7.5 OS repository
2. Configure RHEL 7.5 Extras repository
3. Configure RHEL 7 EPEL repository

```
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

4. Create a local Fedora28 repository and configure the MN to the FC28 Repo

Here's an example to configure the Fedora 28 repository at `/install/repos/fc28`

1. Make the target repository directory on the MN:



```
mkdir -p /install/repos/fc28/ppc64le/Packages
```

2. Download the rpms:

```
cd /install/repos/fc28/ppc64le/Packages
wget https://www.rpmfind.net/linux/fedora-secondary/releases/28/Everything/
↳ ppc64le/os/Packages/p/python2-gevent-1.2.2-2.fc28.ppc64le.rpm
wget https://www.rpmfind.net/linux/fedora-secondary/releases/28/Everything/
↳ ppc64le/os/Packages/p/python2-greenlet-0.4.13-2.fc28.ppc64le.rpm
```

3. Create a repository in that directory:

```
cd /install/repos/fc28/ppc64le/
createrepo .
```

4. Create a repo file /etc/yum.repos.d/fc28.repo and set its contents:

```
[fc28]
name=Fedora28 yum repository for gevent and greenlet
baseurl=file:///install/repos/fc28/ppc64le/
enabled=1
gpgcheck=0
```

5. Download and install xCAT-openbmc-py :

```
wget https://xcat.org/files/xcat/xcat-dep/2.x_Linux/beta/xCAT-openbmc-py-RH7-2.14.6-
↳ snap202204090016.noarch.rpm -O /tmp/xCAT-openbmc-py-RH7.noarch.rpm
yum install /tmp/xCAT-openbmc-py-RH7.noarch.rpm
```

## Install xCAT-openbmc-py on RHEL 7 SN from MN

**Attention:** Instructions below assume Service node has access to the Internet. If not, a local EPEL repository would need to be configured on the Management node, similar to the RHEL Extras repository.

1. Copy Packages directory containing gevent and greenlet rpms from /install/repos/fc28/ppc64le to the directory pointed to by otherpkgdir attribute of the osimage.

```
# Display the directory of otherpkgdir
lsdef -t osimage rhels7.5-ppc64le-install-service -i otherpkgdir -c

# Create Packages directory
mkdir /install/post/otherpkgs/rhels7.5-alternate/ppc64le/xcat/Packages

# Copy rpms
cp /install/repos/fc28/ppc64le/Packages/*.rpm /install/post/otherpkgs/rhels7.5-
↳ alternate/ppc64le/xcat/Packages
```

2. Configure otherpkglist attribute of the osimage

```
chdef -t osimage rhels7.5-ppc64le-install-service otherpkglist=/opt/xcat/share/xcat/
↳ install/rh/service.rhels7.ppc64le.otherpkgs.pkglist
```

3. Add the following entries to the contents of `/opt/xcat/share/xcat/install/rh/service.rhels7.ppc64le.otherpkgs.pkglist`

```
...
xcat/Packages/python2-gevent
xcat/Packages/python2-greenlet
xcat/xcat-core/xCAT-openbmc-py
```

4. Choose one of the 3 methods below to complete the installation

### Install on diskful SN using updatenode

If SN was installed without `xCAT-openbmc-py` package, `updatenode` can be used to install that package.

1. Sync EPEL repository and key file

```
rsync -v /etc/yum.repos.d/epel.repo root@<SN>:/etc/yum.repos.d/
rsync -v /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7 root@<SN>:/etc/pki/rpm-gpg/
```

2. Update packages on SN

```
updatenode <SN> -S
```

### Install on diskful SN using rinstall

1. Configure `synclists` attribute of `osimage`

```
chdef -t osimage rhels7.5-ppc64le-install-service synclists=/install/custom/netboot/
↪compute.synclist
```

2. Add the following to the contents of `/install/custom/netboot/compute.synclist`

```
...
/etc/yum.repos.d/epel.repo -> /etc/yum.repos.d/epel.repo
/etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7 -> /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
```

3. Install SN

```
rinstall <SN> osimage=rhels7.5-ppc64le-install-service
```

### Install on diskless SN using rinstall

1. Add EPEL online repository <https://dl.fedoraproject.org/pub/epel/7/ppc64le> to `pkgdir` attribute of `osimage`:

```
chdef -t osimage -o rhels7.5-ppc64le-netboot-service -p pkgdir=https://dl.
↪fedoraproject.org/pub/epel/7/ppc64le
```

2. Install diskless SN

```
genimage rhels7.5-ppc64le-netboot-service
packimage rhels7.5-ppc64le-netboot-service
rinstall <SN> osimage=rhels7.5-ppc64le-netboot-service
```

## Disable Python Framework

By default, if xCAT-openbmc-py is installed and Python files are there, xCAT will default to running the Python framework.

A site table attribute is created to allow the ability to control between Python and Perl.

- To disable all Python code and revert to the Perl implementation:

```
chdef -t site clustersite openbmcp Perl=ALL
```

- To disable single commands, specify a command separated lists:

```
chdef -t site clustersite openbmcp Perl="rpower,rbeacon"
```

## Performance

### Supported Commands

The following commands are currently supported:

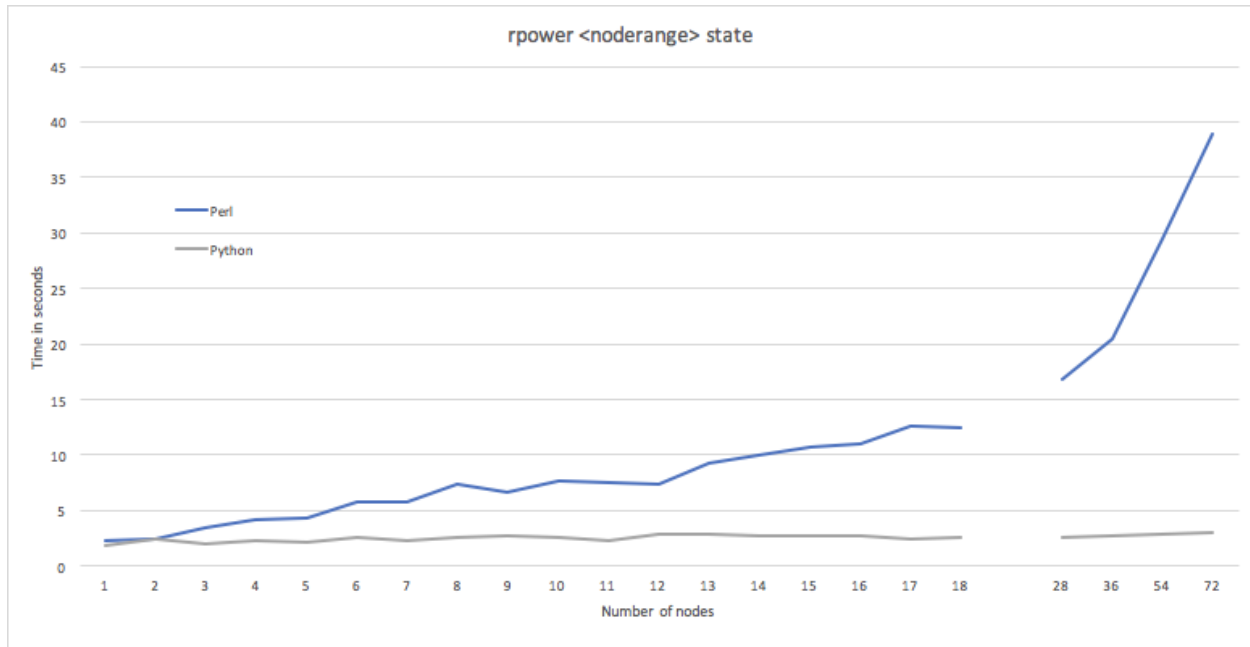
Command	Support	Release	Notes
rpower	Yes	2.13.11	
rinv	Yes	2.13.11	
rbeacon	Yes	2.13.11	
rspconfig	No	2.14	
rsetboot	Yes	2.13.11	
rvitals	Yes	2.13.11	
rflash	No		
reventlog	No	2.14	

## Data

The following graph shows performance gains in the Python code implementation of `mgt=openbmc` when compared to the Perl implementation.

### rpower <noderange> state

This chart gathers data points on a single 18-node frame, up to 4 frames (72-nodes).



## Power9 Firmware Update

### IPMI Firmware Update

The process for updating firmware on the IBM Power9 Server (Boston) is documented below.

### Collect the required files

Collect the following files and put them into a directory on the Management Node.

- pUpdate utility
- .pnor for host
- .bin for bmc

If running `rflash` in Hierarchy, the firmware files/directory must be accessible on the Service Nodes.

### Flash Firmware

Using xCAT `rflash` command, specify the directory containing the files with the `-d` option.

```
rflash <noderange> -d /path-to-directory/
```

The `pUpdate` utility is leveraged in doing the firmware update against the target node and will do the following:

- power off the host
- flash bmc and reboot
- flash host
- power on the host

Monitor the progress for the nodes by looking at the files under `/var/log/xcat/rflash/`.

## Validation

Use the `rinv` command to validate firmware level:

```
rinv <noderange> firm | xcoll
```

## OpenBMC Firmware Update

### Unattended Firmware Flash

Unattended flash of OpenBMC firmware will do the following events:

1. Upload both BMC firmware file and Host firmware file
2. Activate both BMC firmware and Host firmware
3. If BMC firmware becomes activate, reboot BMC to apply new BMC firmware, or else, `rflash` will exit
4. If BMC itself state is `NotReady`, `rflash` will exit
5. If BMC itself state is `Ready`, `rflash` will reboot the compute node to apply Host firmware

Use the following command to flash the firmware unattended:

```
rflash <noderange> -d /path/to/directory
```

If there are errors encountered during the flash process, take a look at the manual steps to continue flashing the BMC.

## Validation

Use one of the following commands to validate firmware levels are in sync:

- Use the `rinv` command to validate firmware level:

```
rinv <noderange> firm -V | grep -i ibm | grep "\*" | xcoll
```

- Use the `rflash` command to validate the firmware level:

```
rflash <noderange> -l | grep "\*" | xcoll
```

## Manual Firmware Flash

The sequence of events that must happen to flash OpenBMC firmware is the following:

1. Power off the Host
2. Upload and Activate BMC
3. Reboot the BMC (applies BMC)
4. Upload and Activate Host
5. Power on the Host (applies Host)

## Power off Host

Use the `rpower` command to power off the host:

```
rpower <noderange> off
```

## Upload and Activate BMC Firmware

Use the `rflash` command to upload and activate the Host firmware:

```
rflash <noderange> -a /path/to/obmc-phosphor-image-witherspoon.ubi.mtd.tar
```

If running `rflash` in Hierarchy, the firmware files must be accessible on the Service Nodes.

**Note:** If a `.tar` file is provided, the `-a` option does an upload and activate in one step. If an ID is provided, the `-a` option just does activate the specified firmware. After firmware is activated, use the `rflash <noderange> -l` to view. The `rflash` command shows (\*) as the active firmware and (+) on the firmware that requires reboot to become effective.

## Reboot the BMC

Use the `rpower` command to reboot the BMC:

```
rpower <noderange> bmcreboot
```

The BMC will take 2-5 minutes to reboot, check the status using: `rpower <noderange> bmcstate` and wait for `BMCReady` to be returned.

**Known Issue:** On reboot, the first call to the BMC after reboot, xCAT will return `Error: BMC did not respond within 10 seconds, retry the command..` Please retry.

## Upload and Activate Host Firmware

Use the `rflash` command to upload and activate the Host firmware:

```
rflash <noderange> -a /path/to/witherspoon.pnor.squashfs.tar
```

If running `rflash` in Hierarchy, the firmware files must be accessible on the Service Nodes.

**Note:** The `-a` option does an upload and activate in one step, after firmware is activated, use the `rflash <noderange> -l` to view. The `rflash` command shows (\*) as the active firmware and (+) on the firmware that requires reboot to become effective.

## Power on Host

Use the `rpower` command to power on the Host:

```
rpower <noderange> on
```

## Validation

Use one of the following commands to validate firmware levels are in sync:

- Use the `rinv` command to validate firmware level:

```
rinv <noderange> firm -V | grep -i ibm | grep "\*" | xcoll
```

- Use the `rflash` command to validate the firmware level:

```
rflash <noderange> -l | grep "\*" | xcoll
```

## Known Issues

### xCAT Genesis Base

xCAT ships a `xCAT-genesis-base` package as part of `xcat-deps`. This is a light-weight diskless linux image based currently on Fedora28, that is used by xCAT to do hardware discovery.

Follow the steps below to build your own version of the `xCAT-genesis-base` on-site. You can include additional drivers or modules or you can build your own version of the `xCAT-genesis-base` on-site using a server running OS other than Fedora28, like Red Hat Enterprise Linux 8. Building `xCAT-genesis-base` on a server running Red Hat Enterprise Linux 7 or earlier, is no longer supported.

1. Download the latest timestamp version of the `xCAT-genesis-builder` RPM provided here: [http://xcat.org/files/xcat/xcat-dep/2.x\\_Linux/beta/](http://xcat.org/files/xcat/xcat-dep/2.x_Linux/beta/)
2. Install the `xCAT-genesis-builder` RPM on a node installed with desired OS (currently verified with Fedora28 and Red Hat Enterprise Linux 8). For more details: <https://github.com/xcat2/xcat-core/tree/master/xCAT-genesis-builder#readme>
3. If additional drivers or modules need to be loaded when genesis kernel boots, edit `xCAT-genesis-builder/xcat-cmdline.sh` and add `modprobe` statements for each one.
4. Build the new `xCAT-genesis-base` RPM:

```
/opt/xcat/share/xcat/netboot/genesis/builder/buildrpm
```

To use the generated RPM from the step above for node discovery:

1. Uninstall the old `xCAT-genesis-base` RPM and install the newly built `xCAT-genesis-base` RPM on the xCAT Management node
2. Execute: `mknb ppc64`
3. Follow *discover nodes*

## HA

### HA Solution Overview

While a xCAT management node `xcatmn1` is running as a primary management node, another node - `xcatmn2` can be configured to act as primary management node in case `xcatmn1` becomes unavailable. The process is manual and requires disabling primary `xcatmn1` and activating backup `xcatmn2`. Both nodes require access to shared storage described below. Use of Virtual IP is also required.

An interactive sample script `xcatha.py` is available to guide through the steps of disabling and activation of xCAT management nodes. Dryrun option in that scrip allows viewing the actions without executing them.

### Configure and Activate Primary xCAT Management Node

#### Disable And Stop All Related Services on Primary xCAT Management Node

Before configuring Virtual IP and shared data, make sure to stop related services. Since primary management node may become unavailable at any time, all related services should be configured to not auto start at boot time.

Use `xcatha.py -d` to disable and stop all related services:

```
./xcatha.py -d
2018-06-22 03:43:51,600 - INFO - [xCAT] Shutting down services:
... goconserver
... conserver
... ntpd
... dhcpd
... named
... xcatd
... postgresql
Continue? [[Y]es/[N]o/[D]ryrun]:
Y
... ..
[xCAT] Disabling services from starting on reboot:
... goconserver
... conserver
... ntpd
... dhcpd
... named
... xcatd
... postgresql
Continue? [[Y]es/[N]o/[D]ryrun]:
Y
```



## Configure Virtual IP

Existing xCAT management node IP should be configured as Virtual IP address, the Virtual IP address should be non-persistent, it needs to be re-configured right after the management node is rebooted. This non-persistent Virtual IP address is designed to avoid ip address conflict when the original primary management node is recovered with this Virtual IP address configured. Since the Virtual IP is non-persistent, the network interface should have a persistent IP address.

1. Configure another IP on primary management node for network interface as static IP, for example, 10.5.106.70:

1. Configure 10.5.106.70 as static IP:

```
ip addr add 10.5.106.70/8 dev eth0
```

2. Edit ifcfg-eth0 file as:

```
DEVICE="eth0"
BOOTPROTO="static"
NETMASK="255.0.0.0"
IPADDR="10.5.106.70"
ONBOOT="yes"
```

3. If want to take new static ip effect immediately, login xcatmn1 using 10.5.106.70, and restart network service, then add original static IP on primary management node 10.5.106.7 as Virtual IP

```
ssh 10.5.106.70 -l root
service network restart
ip addr add 10.5.106.7/8 brd + dev eth0 label eth0:0
```

2. Add 10.5.106.70 into postgresql configuration file on primary management node

1. Add 10.5.106.70 into /var/lib/pgsql/data/pg\_hba.conf:

```
host      all             all             10.5.106.7/32      md5
```

2. Add 10.5.106.70 into listen\_addresses variable in /var/lib/pgsql/data/postgresql.conf:

```
listen_addresses = 'localhost,10.5.106.7,10.5.106.70'
```

3. Modify provision network entry mgtifname as eth0:0 on primary management node:

```
tabedit networks
"10_0_0_0-255_0_0_0","10.0.0.0","255.0.0.0","eth0:0","10.0.0.103",,"<xcatmaster>",,,
↪,,,,,,,,,"1500",,
```

## Configure Shared Data

The following xCAT directory structure should be accessible from primary xCAT management node:

```
/etc/xcat
/install
~/.xcat
/var/lib/pgsql
/tftpboot
```

## Synchronize /etc/hosts

Since the /etc/hosts is used by xCAT commands, the /etc/hosts should be synchronized between the primary management node and backup management node.

## Synchronize Clock

It is recommended that the clocks are synchronized between the primary management node and backup management node.

## Activate Primary xCAT Management Node

Use `xcatha.py` interactive activate `xcatmn1`:

```
./xcatha.py -a
[Admin] Verify VIP 10.5.106.7 is configured on this node
Continue? [[Y]es/[N]o]:
Y
[Admin] Verify that the following is configured to be saved in shared storage and
↪accessible from this node:
... /install
... /etc/xcat
... /root/.xcat
... /var/lib/pgsql
... /tftpboot
Continue? [[Y]es/[N]o]:
Y
[xCAT] Starting up services:
... postgresql
... xcatd
... named
... dhcpd
... ntpd
... conserver
... goconserver
Continue? [[Y]es/[N]o/[D]ryrun]:
Y
2018-06-24 22:13:09,428 - INFO - ===> Start all services stage <===
2018-06-24 22:13:10,559 - DEBUG - systemctl start postgresql [Passed]
2018-06-24 22:13:13,298 - DEBUG - systemctl start xcatd [Passed]
    domain=cluster.com
2018-06-24 22:13:13,715 - DEBUG - lsdef -t site -i domain|grep domain [Passed]
Handling bybc0607 in /etc/hosts.
Handling localhost in /etc/hosts.
Handling bybc0609 in /etc/hosts.
Handling localhost in /etc/hosts.
Getting reverse zones, this may take several minutes for a large cluster.
Completed getting reverse zones.
Updating zones.
Completed updating zones.
Restarting named
```

(continues on next page)

(continued from previous page)

```
Restarting named complete
Updating DNS records, this may take several minutes for a large cluster.
Completed updating DNS records.
DNS setup is completed
2018-06-24 22:13:17,320 - DEBUG - makedns -n [Passed]
Renamed existing dhcp configuration file to /etc/dhcp/dhcpd.conf.xcatbak

Warning: No dynamic range specified for 10.0.0.0. If hardware discovery is being used, a
dynamic range is required.
2018-06-24 22:13:17,811 - DEBUG - makedhcp -n [Passed]
2018-06-24 22:13:18,746 - DEBUG - makedhcp -a [Passed]
2018-06-24 22:13:18,800 - DEBUG - systemctl start ntpd [Passed]
2018-06-24 22:13:19,353 - DEBUG - makeconservercf [Passed]
2018-06-24 22:13:19,449 - DEBUG - systemctl start conserver [Passed]
```

## Activate Backup xCAT Management Node to be Primary Management Node

1. Install xCAT on backup xCAT management node xcatmn2 with local disk
2. Switch to PostgreSQL database
3. Disable and deactivate services using `xcatha.py -d` on both xcatmn2 and xcatmn1
4. Remove Virtual IP from primary xCAT Management Node xcatmn1:

```
ip addr del 10.5.106.7/8 dev eth0:0
```

5. Configure Virtual IP on xcatmn2
6. Add Virtual IP into `/etc/hosts` file

```
10.5.106.7 xcatmn1 xcatmn1.cluster.com
```

7. Connect the following xCAT directories to shared data on xcatmn2:

```
/etc/xcat
/install
~/.xcat
/var/lib/pgsql
/tftpboot
```

8. Add static management node network interface IP 10.5.106.5 into PostgreSQL configuration file

1. Add 10.5.106.5 into `/var/lib/pgsql/data/pg_hba.conf`:

```
host      all          all          10.5.106.5/32      md5
```

2. Add 10.5.106.5 into `listen_addresses` variable in `/var/lib/pgsql/data/postgresql.conf`:

```
listen_addresses = 'localhost,10.5.106.7,10.5.106.70,10.5.105.5'
```

9. Use `xcatha.py -a` to start all related services on xcatmn2
10. Modify provision network entry `mgtifname` as `eth0:0`:

```
tabedit networks
"10_0_0_0-255_0_0_0","10.0.0.0","255.0.0.0","eth0:0","10.0.0.103",,"<xcatmaster>",,,
→,,,,,,,,,"1500",,
```

## Unplanned failover: primary xCAT management node is not accessible

If primary xCAT management node becomes not accessible before being deactivated and backup xCAT management node is activated, it is recommended that the primary node is disconnected from the network before being rebooted. This will ensure that when services are started on reboot, they do not interfere with the same services running on the backup xCAT management node.

### 1.7.2 PERCS

See documentation here: [https://sourceforge.net/p/xcat/wiki/XCAT\\_Power\\_775\\_Hardware\\_Management/](https://sourceforge.net/p/xcat/wiki/XCAT_Power_775_Hardware_Management/)

## 1.8 Troubleshooting

This chapter introduces the methods of debugging and troubleshooting a xCAT cluster. [General xCAT troubleshooting and debugging suggestions](#)

Additional recommendations:

### 1.8.1 Operating System Installation

The ability to access the installer or to collect logs during the installation process can be helpful when debugging installation problems.

A new attribute is provided in the **site** table called **xcatdebugmode**.

- **xcatdebugmode=0**: Diagnostic entries will be shown in corresponding log files.
- **xcatdebugmode=1**: Diagnostic entries will be shown in corresponding log files and debug port will be opened.
- **xcatdebugmode=2**: Diagnostic entries will be shown in corresponding log files, debug port will be opened and SSH access is enabled.

Supported OS:

- RHEL: 6.7 and above
- SLES: 11.4 and above
- UBT: 14.04.3 and above

The following behavior is supported during OS installation:

<b>xcatdebugmode</b>	0			1			2		
	RHEL	SLES	UBT	RHEL	SLES	UBT	RHEL	SLES	UBT
Log Collecting	Y	Y	Y	Y	Y	Y	Y	Y	Y
Enable Debug Port	N	N	N	Y	Y	Y	Y	Y	Y
SSH Access	N	N	N	N	N	N	Y	Y	Y

- Y - the behavior is supported by OS at specified **xcatdebugmode** level.

- N - the behavior is not supported.

Next chapter introduces the procedures on how to troubleshoot operating system installation.

## Log Collection: Collecting logs of the whole installation process

The ability to collect logs during the installation (diskful and diskless) can be enabled by setting the “site.xcatdebugmode” to different levels (0,1,2), which is quite helpful when debugging installation problems.

### The diskful provision logs:

xcatdebug-mode		0			1			2		
OS Distribution		RHEL	SLES	UBT	RHEL	SLES	UBT	RHEL	SLES	UBT
<b>Pre-Install</b>	MN	N			N			N		
	log CN	C1			C1 C2			C1 C2		
<b>Installer</b>	MN	N	N	N	M1	M1	M1	M1	M1	M1
	log CN	C3	C3	C3	C3	C3	C3	C3	C3	C3
<b>Post-Install</b>	MN	M2			M3			M3		
	log CN	C1			C1 C2			C1 C2		
<b>PostBoot</b>	MN	M2			M3			M3		
	log CN	C1			C1 C2			C1 C2		

### The diskless provision logs:

xcatdebug-mode		0			1			2		
OS Distribution		RHEL	SLES	UBT	RHEL	SLES	UBT	RHEL	SLES	UBT
<b>Provision</b>	MN	N			M3			M3		
	log CN	N			N			N		
<b>PostBoot</b>	MN	M3			M3 M4			M3 M4		
	log CN	C1			C1 C2			C1 C2		

- **Pre-Install** logs: the logs of pre-installation scripts, including:
  - %pre section in anaconda,
  - <pre-scripts/> section for SUSE and partman/early\_command and preseed/early\_command sections for ubuntu.
  - STDOUT and STDERR of the scripts
  - debug trace output of bash scripts with set -x
- **Installer** logs: the logs from the os installer itself, i.e, the logs of installation program (anaconda, autoyast and preseed,etc.)
- **Post-Install** logs: the logs of post-installation scripts, including
  - %post section in anaconda,

- <chroot-scripts/> and <post-scripts/> sections for SUSE and preseed/late\_command section for ubuntu.
- STDOUT and STDERR of the scripts
- debug trace output of bash scripts with `set -x`
- **Provision** logs: the logs during the diskless provision.
- **PostBootScript** logs: the logs during the post boot scripts execution, which are specified in `postbootscripts` attribute of node and osimage definition and run during the 1st reboot after installation.

MN: the logs forwarded to management node:

- M1: the installer logs will be forwarded to the MN in `/var/log/xcat/computes.log` file.
- M2: the error messages will be forwarded to `/var/log/xcat/computes.log` file on MN only when critical error happens.
- M3: the installation logs will be forwarded to `/var/log/xcat/computes.log` file on MN.
- M4: the debug trace(`set -x` or `-o xtrace`) of bash scripts enabled.
- N: the logs will not be forwarded to MN.

CN: the logs on compute node:

- C1 - the installation logs will be saved to `/var/log/xcat/xcat.log` file on CN.
- C2 - the debug trace(`set -x` or `-o xtrace`) of bash scripts enabled.
- C3 - the installer logs will be saved to the CN in `/var/log/anaconda` for RHEL, `/var/log/YaST2` for SLES, `/var/log/installer` for UBT.
- N - the logs will not be saved to CN.

## Enabling Debug Port: Running commands in the installer from MN

**This mode is supported with debug level set to 1 or 2**

xCAT creates a service inside the **installer**, listening on port 3054. It executes commands sent to it from the xCAT MN and returns the response output.

The command `runcmdinstaller` can be used to send request to installer:

Usage: `runcmdinstaller <node> "<command>"`

Note: Make sure all the commands are quoted by ""

To list all the items under the `/etc` directory in the installer: `runcmdinstaller c910f03c01p03 "ls /etc"`

## SSH Access: Accessing the installer via “ssh”

**This mode is supported with debug level set to 2**

When ssh access to the installer is enabled, the admin can login into the installer through:

1. For RHEL, the installation won't halt, just login into the installer with `ssh root@<node>`.
2. For SLES, the installation will halt after the ssh server is started, the console output looks like:

```
*** sshd has been started ***

*** login using 'ssh -X root@<node>' ***
*** run 'yast' to start the installation ***
```

Just as the message above suggests, the admin can open 2 sessions and run `ssh -X root@<node>` with the configured system password in the `passwd` table to login into the installer, then run `yast` to continue installation in one session and inspect the installation process in the installer in the other session.

After the installation is finished, the system requires a reboot. The installation will halt again before the system configuration, the console output looks like:

```
*** Preparing SSH installation for reboot ***
*** NOTE: after reboot, you have to reconnect and call yast.ssh ***
```

Just as the message above suggests, the admin should run `ssh -X root@<node>` to access the installer and run `yast.ssh` to finish the installation.

**Note:** For `sles12`, during the second stage of an SSH installation YaST freezes. It is blocked by the SuSE-Firewall service because the `SYSTEMCTL_OPTIONS` environment variable is not set properly. Workaround: When logged in for the second time to start the second stage of the SSH installation, call `yast.ssh` with the `--ignore-dependencies` as follows:

```
SYSTEMCTL_OPTIONS=--ignore-dependencies yast.ssh
```

- For UBT, the installation will halt on the message in the console similar to:

```
----- [!!] Continue installation remotely using SSH |-----
|
|                               Start SSH
|
| To continue the installation, please use an SSH client to connect to
| the IP address <node> and log in as the "installer" user. For
| example:
|
|     ssh installer@<node>
|
| The fingerprint of this SSH server's host key is:
| <SSH_host_key>
|
| Please check this carefully against the fingerprint reported by your
| SSH client.
|
|                               <Continue>
```

Just as the message above suggests, the admin can run `ssh installer@<node>` with the password “cluster” to login into the installer, the following message shows on login:

```
----- [!!] Configuring d-i |-----
|
| This is the network console for the Debian installer. From
| here, you may start the Debian installer, or execute an
```

(continues on next page)

(continued from previous page)

```
interactive shell.  
  
To return to this menu, you will need to log in again.  
  
Network console option:  
  
        Start installer  
        Start installer (expert mode)  
        Start shell
```

The admin can open 2 sessions and then select “Start installer” to continue installation in one session and select “Start shell” in the other session to inspect the installation process in the installer.

## 1.9 Developers

This page is for developers interested in working with xCAT.

### 1.9.1 Contributor and Maintainer Agreements

We welcome developers willing to contribute to the xCAT project to help make it better.

Follow the guidelines below.

#### Contributors

- Sign our Contributor License Agreement (CLA). You can find the CLA [here](#)
- If you are contributing on behalf of your employer, we’ll need a signed copy of our Corporate Contributor License Agreement (CCLA). You can find the CCLA [here](#).
- Fork the repo, make your changes, and open pull requests!

#### Maintainers

If you are an experienced xCAT user and plan to contribute to the xCAT code regularly, you can request to become an xCAT Maintainer (includes git push access) by first sending an email to the xCAT users mailing list ([xcat-user@lists.sourceforge.net](mailto:xcat-user@lists.sourceforge.net)).

- If approved, complete and sign our **xCAT Maintainer License Agreement**, then scan and email a PDF file to [xcat-legal@lists.sourceforge.net](mailto:xcat-legal@lists.sourceforge.net).
- Additional information can be found in our [maintainers guide](#) below:



## Maintainers Guide

The roles and responsibilities of the maintainers are:

- set the direction for the xCAT project, including architectural and design decisions
- commit code (new function or fixes) to the xCAT Git repository (either their own code, or on behalf of another contributor - see below)
- review requests for xCAT contributors to become maintainers (All such requests will be subject to a vote by current maintainers)
- review and help resolve technical concerns or problems regarding the project

All decisions by the maintainers are made by consensus.

When a **maintainer** merges a pull request for a **contributor** they must:

- Require that each code contributor complete and sign the CLA [here](#) or CCLA [here](#) email a PDF copy to [xcat-legal@lists.sourceforge.net](mailto:xcat-legal@lists.sourceforge.net).
- Require that all code be contributed under the EPL.

xCAT is licensed under the [Eclipse Public License](#).

## 1.9.2 GitHub

Developers are encouraged to fork the [xcat2](#) repositories, make changes, and submit pull requests. This mechanism allows for better collaboration and gives the community a chance to review and provide feedback before the code is pulled into the product.

GitHub provides excellent documentation on [Collaborating](#) and the following is only provided as a quick reference.

### Getting Started

#### Fork a repository

Forking a repository is taking a copy of a GitHub repository and associating it to your own account space so you can make changes to the code base. Since you own this copy, you will have fetch/push access to the repository.

We will first create a fork on the [xcat2/xcat-core](#) project so that we can work on functions without affecting the mainline (or *upstream*) code. Additionally, by creating a fork, you are able to generate *pull request* so that the changes can be easily seen and reviewed by other community members.

- In GitHub UI, find a project that you want to fork and click on the “Fork” icon.



**Note:** The target is your own account space

- After the fork is created, there is a copy of the repository under your own account

```
<userid>/xcat-core <- forked copy
```

```
xcat2/xcat-core <- upstream
```

## Clone the forked repository

- On your development machine, clone the forked repository from **your** account:

**Note:** Ensure the clone is from `<userid>/xcat-core` and **not** `xcat2/xcat-core`

```
$ git clone git@github.com:<userid>/xcat-core.git
```

This now becomes the origin remote repository:

```
$ git remote -v
origin  git@github.com:<userid>/xcat-core.git (fetch)
origin  git@github.com:<userid>/xcat-core.git (push)
```

## Configure an upstream repository

- In order to get updates from the upstream project: `xcat2/xcat-core`, you will need to add another remote repository to fetch from.

```
$ git remote add upstream git@github.com:xcat2/xcat-core.git
```

View the configured repositories:

```
$ git remote -v
origin  git@github.com:<userid>/xcat-core.git (fetch)
origin  git@github.com:<userid>/xcat-core.git (push)
upstream      git@github.com:xcat2/xcat-core.git (fetch)
upstream      git@github.com:xcat2/xcat-core.git (push)
```

## Changing Code and Pull Requests

### Syncing a Fork

**Note:** *The examples below all reference the master branch*

### Before the Syncing

References: <https://help.github.com/articles/syncing-a-fork/>

From time to time, your master branch will start to **fall behind** the upstream/master because changes are being pulled into the `xcat2/xcat-core` project from other developers.



### Temporarily Stashing work (optional)

If working on changes in another branch, it's recommended to stash the work before switching to the `master` branch so the pull does not wipe out any uncommitted changes. To stash work in the current branch:

1. Add (*not commit*) any untracked files and directories:

```
$ git add <untracked_files_and_directories>
```

2. Stash the work:

```
$ git stash
```

### Switch and Update the master branch of your forked copy

1. Switch to the master branch.

```
$ git checkout master
```

2. Pull the commits from the upstream `master` (`xcat2/xcat-core`) to your local master branch.

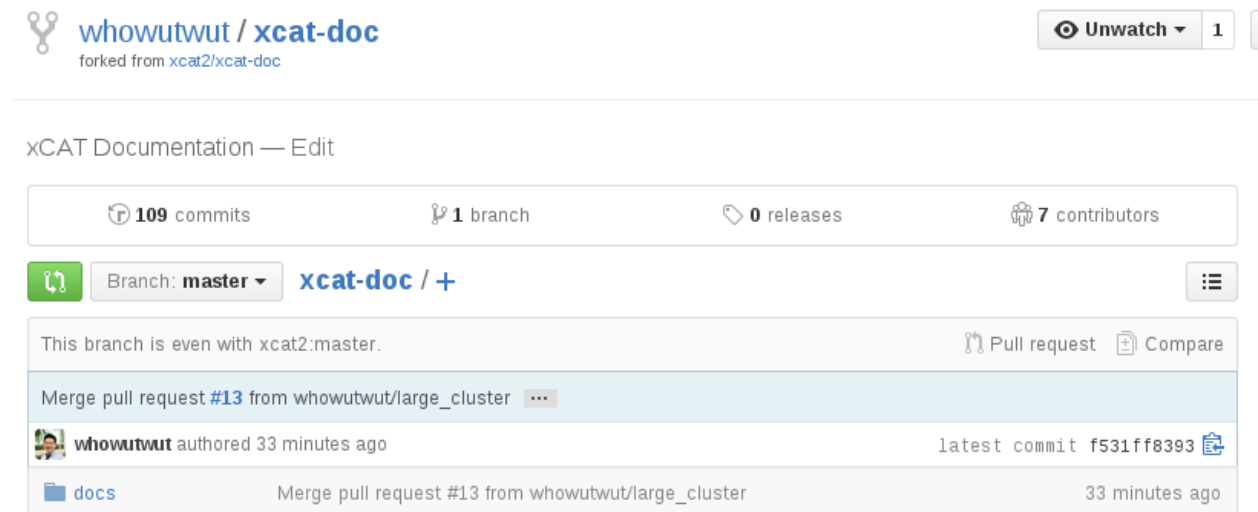
```
$ git pull upstream master
remote: Counting objects: 38, done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 38 (delta 14), reused 9 (delta 9), pack-reused 14
Unpacking objects: 100% (38/38), done.
From github.com:xcat2/xcat-core
* branch          master      -> FETCH_HEAD
   8f0cb07..d0651b5 master      -> upstream/master
Updating 8f0cb07..d0651b5
Fast-forward
...
```

3. Push the commits from upstream merged to your local master to your forked copy in GitHub:

```
$ git push origin master
```

### After the Syncing

Your fork master branch should now be **even** with `xcat2/xcat-core`



The screenshot shows the GitHub interface for the repository 'whowutwut / xcat-doc', which is a fork of 'xcat2/xcat-doc'. The repository has 109 commits, 1 branch, 0 releases, and 7 contributors. The current branch is 'master'. A pull request is open, titled 'Merge pull request #13 from whowutwut/large\_cluster', which was authored 33 minutes ago. The pull request includes a commit 'f531ff8393' and a file named 'docs'. The status bar at the bottom indicates 'Merge pull request #13 from whowutwut/large\_cluster' and '33 minutes ago'.

### Unstashing work (optional)

If work has been stashed, to continue back where you left off, switch to the target branch and run : `git stash pop`

### Creating Branches

**Note:** To more easily keep the forked repository and upstream repositories in sync, we recommended that you keep your master branch free from changes and create branches to contain the changes in function you are working on.

### Local Branches

Git branches are very light weight and easy to create. Simply use the `git branch <branch_name>` command.

Since we are using pull requests to merge changes back to the upstream project, you will need to have your changes committed to a branch on GitHub (calling this a “remote branch”) so that GitHub can detect differences and pull requests can be created.

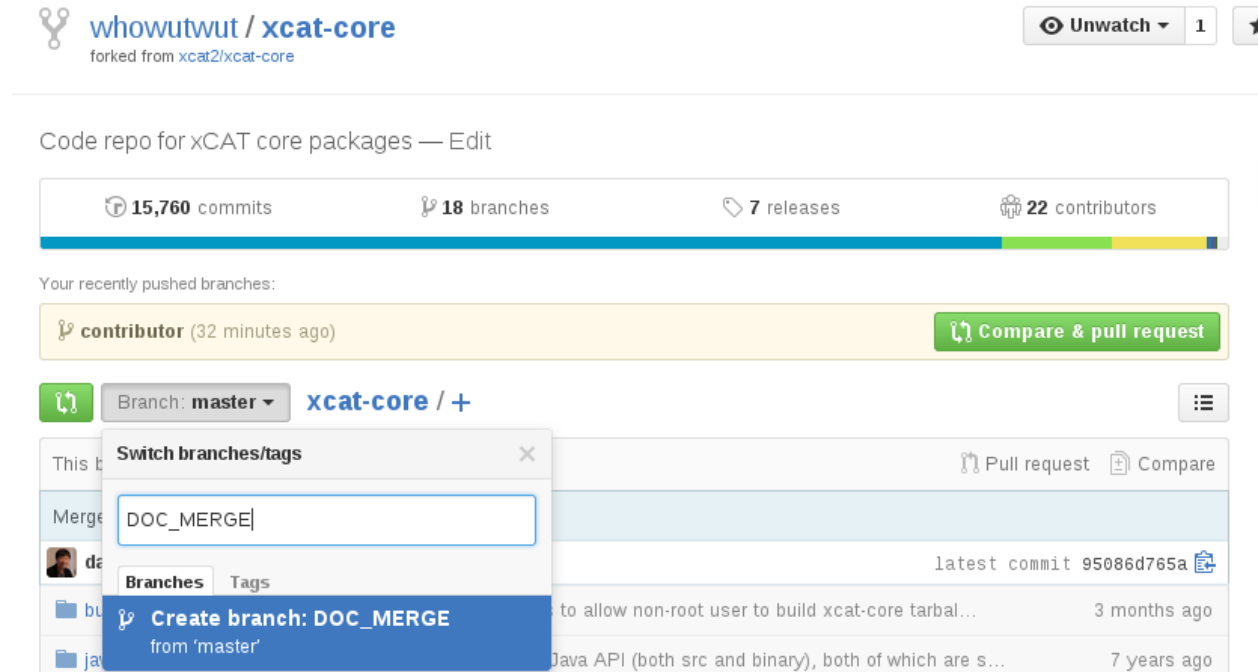
### Remote Branches

Reference articles: \* <http://www.gitguys.com/topics/adding-and-removing-remote-branches/>

## From GitHub

- Under your repository, click on the **Branch** dropdown and type in a name for the new branch, then hit enter.

In the example below, creating a new branch off 'master' called DOC\_MERGE



- Since we created the branch from the UI, a refresh needs to be done in order to see the new branch. Refresh by fetching from the origin repository:

```
$ git fetch origin
Enter passphrase for key '/home/vhu/.ssh/github/id_rsa':
From github.com:whowutwut/xcat-doc
* [new branch]      DOC_MERGE -> origin/DOC_MERGE
```

- Show the remote branches: `git branch -r`

```
$ git branch -r
origin/HEAD -> origin/master
origin/large_cluster
origin/master
origin/DOC_MERGE <=== NEW BRANCH
origin/sync
upstream/master
```

## From Command Line (CLI)

- Create a branch:

```
$ git branch
* master

$ git branch skeleton

$ git branch
* master
  skeleton
```

- Push the newly created branch to origin (on GitHub):

```
$ git push origin skeleton
Enter passphrase for key '/home/vhu/.ssh/github/id_rsa':
Total 0 (delta 0), reused 0 (delta 0)
To git@github.com:whowutwut/xcat-doc.git
 * [new branch]      skeleton -> skeleton
```

Verify that the branch is there by looking at the remote branches:

```
$ git branch -r
origin/HEAD -> origin/master
origin/master
origin/skeleton <=== HERE IT IS
upstream/master
```

## Changing Code

### Checkout Branch

Checkout and switch to the branch using `git checkout -b`

```
$ git checkout -b mybranch origin/mybranch
Branch mybranch set up to track remote branch mybranch from origin.
Switched to a new branch 'mybranch'
```

### Changing the code

Now you are ready to make changes related to your function in this branch

## Multiple Remote Branches

It may take days before your pull request is properly reviewed and you want to keep changes out of that branch so in the event that you are asked to fix something, you can push directly to the branch with the active pull request.

Creating additional branches will allow you to work on different tasks/enhancements at the same time. You can easily manage your working changes between branches with `git stash..`

## Committing code and pushing to remote branch

Once your code is ready...

1. Commit the code to your local branch:

```
$ git add <files>
$ git commit | git commit -m "<comments>"
```

2. Push the changes to your remote branch:

```
$ git push origin <branch name>
```

## Pull Requests

### Creating Pull Requests

Once your changes are ready to be submitted to the xcat team, the easiest way to generate a pull request is from the GitHub UI.

1. Under your project, click on the “branches” link



2. Find the branch that contains your changes and click on the “New pull request” button

Overview
Yours
Active
Stale
All branches

Search branches...

Default branch

master Updated 22 hours ago by whowutwut
Default
Change default branch

Your branches

doc\_cleanup Updated 21 hours ago by whowutwut
0 | 1
New pull request

DOC\_MERGE Updated 5 days ago by whowutwut
44 | 0
New pull request

manpage Updated 11 days ago by daniceexi
120 | 0
New pull request

3. Submit the pull request!

## Changing Pull Request

After submitting a pull request, you may get comments from reviewer that something needs to be changed. Then you can use following steps to change your pull request.

1. Change any code and add them to be tracked in git

```
$ git checkout <mybranch>
$ vi <files>
$ git add <files>
```

2. Commit the change to last commit in the branch instead of creating a new commit. This step is useful to keep the change of this pull request in one commit instead of multiple ones.

```
$ git commit --amend
```

3. Push the new commit to remote repository. Then the commit will be displayed in pull request automatically.

```
$ git push origin <mybranch> -f
```

## Resolving Conflicts in the Pull Request

During the reviewing of your pull request, another pull request may be merged which contains changes that conflicts with your change so that your pull request can no longer be merged automatically. You can use following steps to resolve the conflicts.

1. Update the upstream, replace <upstream> with the name if your upstream repo

```
$ git fetch <upstream>
```

2. Checkout to your working branch

```
$ git checkout <mybranch>
```

3. rebase your branch to the master branch in the <upstream>



```
$ git rebase <upstream>/master
```

4. In the previous step, If there are conflicts, the rebase will stop and you will see CONFLICT messages for certain files. Edit the files to resolve the conflicts manually and then use **git add** the re-add the files to be tracked in git.

```
$ vi <files>
$ git add <files>
```

5. Continue the rebase repeat the step above for any additional conflicts

```
$ git rebase --continue
```

6. Once the rebase is complete and conflicts are resolved, **force** push the change to your repository

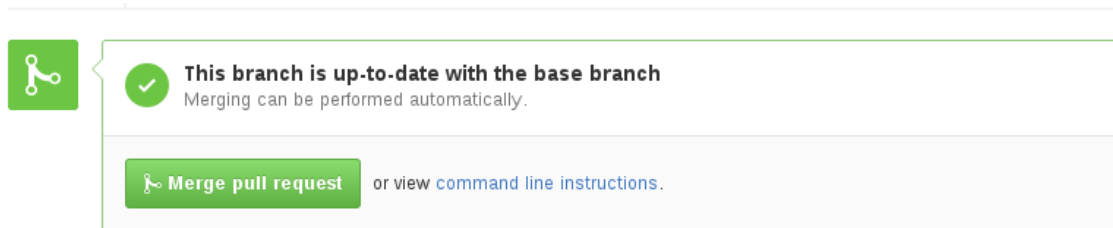
```
$ git push origin <mybranch> -f
```

If all the conflicts are resolved, the pull request should automatically turn green again and is able to be merged automatically.

## Reviewing Pull Requests as a Maintainer

When you are looking over a pull request, you can merge the changes into your own temporary branch to give the code or changes a try before merging into the parent project.

1. From the merge request, click on the `command line instructions` link:



2. Then under **Step 1:**, there are instruction for creating a temp branch and pulling the changes from the pull request:

### Merging via command line

If you do not want to use the merge button or an automatic merge cannot be performed, you can perform a manual merge on the command line.

HTTPS
Git
Patch

**Step 1:** From your project repository, check out a new branch and test the changes.

```
git checkout -b immarvin-master master
git pull git://github.com/immarvin/xcat-core.git master
```

**Step 2:** Merge the changes and update on GitHub.

```
git checkout master
git merge --no-ff immarvin-master
git push origin master
```

## Deleting Branches

### From Command Line

Switch off the branch that you want to delete. Most of the time, switching to master will allow you to delete a working branch:

```
$ git checkout master
$ git branch -D mybranch
```

Delete the remote branch off GitHub:

```
$ git push origin --delete mybranch
Enter passphrase for key '/home/vhu/.ssh/github/id_rsa':
To git@github.com:whowutwut/xcat-doc.git
- [deleted]          mybranch
```

Verify branch is gone:

```
$ git branch -r
origin/HEAD -> origin/master
origin/large_cluster
origin/master
origin/sync
upstream/master
```

### Sync up GitHub and Local Machine

There are times when you delete the branch off your local machine or from GitHub and it's become out of sync, you sync up the list, run the following:

```
git remote prune origin
```

## 1.9.3 Guides

### Documentation

#### Guidelines for xCAT Documentation

The following guidelines should be followed when making changes to the xCAT documentation to help create consistency in the documentation.

## Document Structure

### Section Structure

xCAT doc page may have 4 levels of title at most:

```
The First Title
=====

The Second Title
-----

The Third Title
.....

The Forth Title
.....
```

### List Structure

#### Bullet Lists

- Bullet one  
The Content.
- Bullet Two  
The Content.

```
* Bullet one
The Content.
* Bullet Two
The Content.
```

#### Enumerated List

1. Item 1
  - a) item a
  - b) item b
2. Item 2
  - a) item a

```
1. Item 1
a) item a
b) item b
2. Item 2
a) item a
```

## Include another file

To add contents of a document file inside another file, use `.. include::`. This is useful when a common information needs to be displayed in multiple files, without the use of a hyperlink.

```
.. include:: config_common.rst
```

**Note:** Do not put customized link targets, such as `.. _my_link_target:` inside the file to be included. If you do, a warning for a duplicate label will be displayed during the documentation build process.

## Index file

Index.rst files contain the `.. toctree::` tag. Files listed under that tag will have links to them displayed in the left side navigation area. If a documentation file does not wish to be accessible from the navigation area, do not list it under the `.. toctree::`.

**Note:** If a file is not listed under the `.. toctree::` it might generate a warning during the documentation build **WARNING: document isn't included in any toctree.** To eliminate such warning, add the file to the `exclude_patterns` list in the `docs/source/conf.py` file. However, do not add a file to the `exclude_patterns` list if it contains a customized link target, such as `.. _my_link_target:`. This link target will not be visible to other files and a **WARNING: undefined label:** will be displayed during the documentation build.

## Hyperlinks -> Internal Links -> External Links

Add links to refer other web page is a very common way in writing document, it's very helpful to reduce the doc duplication and make docs easy to understand. Following are several ways to add a link in the xCAT documentation.

- **Add an Internal Link to ``Customized Link Target``**

Customized Link Target means a user defined **Link Target**.

Define a **Link Target** named `my_link_target`:

```
.. _my_link_target:

**Customized Link Target**

This part of content is a link target which can be linked by other content.
```

Link to the customized link target `my_link_target` *my link*:

```
:ref:`my link <my_link_target>`
```

Usage: This method is used to add a **Link Target** in any page that can be referred by any other pages.

- **Add an Internal Link to Current Page**

Link to an internal section in current page: *Guidelines for xCAT Documentation*:

```
`Guidelines for xCAT Documentation`_
```

Usage: Every title of a section is an auto-generated 'link target', so you can use it directly. But it's only available inside the current page.

- **Add an Internal Link to Other Page via File Path**

Link to page *http://server/overview/suport\_list.html* with **absolute file path**

```
:doc:`support list </overview/support_list>`
```

Link to page [http://server/overview/suport\\_list.html](http://server/overview/suport_list.html) with **relative file path**

```
:doc:`support list <../overview/support_list>`
```

Usage: When you want to link to another whole page but don't want to make a Customized Link Target in that source page, you can use the file path to link it directly.

- **Add an External Link**

Link to an external web page: [google](http://www.google.com):

```
`google <http://www.google.com>`_
```

Usage: When you want to link to a page which does not belong to xCAT documentation.

Note: The `https://` keyword must be added before the web page URL.

- **Add a Link with Explicit URL Displayed**

Link to <http://www.google.com>:

```
http://www.google.com
```

Usage: Make a link and display the URL.

## Add OS or ARCH Specific Contents

When writing a common xCAT doc, we always encounter the case that certain small part of content needs to be OS or ARCH specific. In this case, use the following format to add specific branches.

The keyword in the [] can be an OS name or ARCH name, or any name which can distinguish the content from other part.

The valid keyword includes: **RHEL**, **SLES**, **UBUNTU**, **CENTOS**, **X86\_64**, **PPC64**, **PPC64LE**. If the keyword is an OS, it can be postfixed with an OS version e.g. RHEL7.

- **[RHEL7]**

This part of description is for [rh7] specific.

- **[SLES]**

This part of description is for [sles] specific.

- **[PPC64LE]**

This part of description is for [ppc64le] specific.

```
* ** [RHEL7] **
```

This part of description **is for** [rh7] specific.

Miscellaneous

Add a Comment

The sentence started with `..` will be a comment that won't be displayed in the doc.

```
.. This is a comment
```

Add Literal Block

If you want to add a paragraph of code or something that don't want to be interpreted by browser:

```
If you want to add a paragraph of code or something that don't want to be interpreted by browser: ::  
    #lsdef node1  
    #tabdump
```

Decorate Word

If you want to display one or several words to be Literal Word:

```
If you want to display one or several words to be ``Literal Word``
```

If you want to make a **strong emphasis** of the word:

```
If you want to make a strong emphasis of the word:
```

Add a Table

Add a table in the doc:

Header 1	Header 2	Header 3
body row 1	column 2	column 3
body row 2	Cells may span columns.	
body row 3	Cells may span rows.	<ul style="list-style-type: none"><li>• Cells</li></ul>
body row 4		<ul style="list-style-type: none"><li>• contain</li><li>• blocks.</li></ul>

```
+-----+-----+-----+  
| Header 1 | Header 2 | Header 3 |  
+-----+-----+-----+  
| body row 1 | column 2 | column 3 |  
+-----+-----+-----+  
| body row 2 | Cells may span columns. |  
+-----+-----+-----+  
| body row 3 | Cells may | - Cells |  
+-----+ span rows. | - contain |
```

(continues on next page)

(continued from previous page)

```
| body row 4 |           | - blocks. |
+-----+-----+-----+
```

## Add Footnotes

This is the first example of footnotes<sup>1</sup>.

This is the second example of footnotes<sup>2</sup>.

```
This is the first example of footnotes [1]_.
This is the second example of footnotes [2]_.

.. [1] First footnote
.. [2] Second footnote
```

## Code Development

### Code Standard for Perl

This document gives out a Code Standard for Perl programming in xCAT. All the Perl code which is checked in to xCAT code repository should follow this standard.

This document does not give the coding rules one by one, but give a piece of example code. You need to follow the example code strictly.

This standard referred to the Perl code style from perldoc: <http://perldoc.perl.org/perlstyle.html>

### Tidy Your Code

Meanwhile, you are recommended to use following command line to tidy your code:

```
perltidy -w -syn -g -opt -i=4 -nt -io -nbbc -kbl=2 -pscf=-c -aws \
-pt=2 -bbc -nolc <orig_code> -o <formatted_code>
```

How to install perltidy tool:

- [RHEL]

```
yum install perltidy.noarch
```

- [UBUNTU]

```
apt-get install perltidy
```

<sup>1</sup> First footnote

<sup>2</sup> Second footnote

**Code Standard Example:**

```
#!/usr/bin/perl

# This is a perl program example to demo the recommended
# style to write perl code

use strict;
use warnings;

#-----

=head3    subroutine_example
    Descriptions:
        This a subroutine to demo how to write perl code
    Arguments:
        param1: The first parameter
        param2: The second parameter
        param3: The third parameter
    Returns:
        0 - success
        0 - fail
=cut

#-----

sub subroutine_example {
    my ($param1, $param2, $param3) = @_;

    print "In the subroutine subroutine_example.\n";

    return 0;
}

# Declare variables
my $a_local_scale;
my @a_local_array;
my %a_local_hash;

$a_local_scale = 1;

@a_local_array = ("a", "b", "c");

%a_local_hash = (
    "v1" => 1,
    "v2" => 2,
    "v3" => 3,
);

# Demo how to check the key of hash
if (%a_local_hash and
    defined($a_local_hash{v1}) and
    defined($a_local_hash{v2}) and
```

(continues on next page)



(continued from previous page)

```
defined($a_local_hash{v3})) {

    # Calculate the sum of values
    my $sum_of_values = $a_local_hash{v1}
        + $a_local_hash{v2}
        + $a_local_hash{v3};

    print "The sum of values: $sum_of_values.\n";
}

# Demo how to check whether the array is empty
if (@a_local_array) {
    print "Has element in array: " . join(',', @a_local_array) . "\n";
}
elsif ($a_local_scale) {
    print "The value of a scale variable: $a_local_scale.\n";
}
else {
    # None of above
    print "Get into the default path.\n";
}

# Call the subroutine subroutine_example()
subroutine_example($a_local_scale, \@a_local_array, %a_local_hash);

exit 0;
```

## Building Source Code

### xcat-core

Clone the xCAT project from [GitHub](#):

```
cd xcat-core
./buildcore.sh
```

## **xcat-deps**

The `xcat-deps` package is currently owned and maintained by the core development on our internal servers. Use the packages created at: <http://xcat.org/download.html#xcat-dep>

## **man pages**

The xCAT man pages are written in Perl POD files and automatically get built into the xCAT rpms. The content in the .pod files are always the master.

In the past, the man pages were converted into html files and uploaded to SourceForge. In moving to [ReadTheDocs](#) we want to also provide the man pages as references in the documentation. To convert the pods to `rst`, we are using The Perl module: `pod2rst`.

The following steps will help configure `pod2rst` and be able to generate the changes `.rst` files to push to GitHub.

1. Download the following Perl modules:

- [Pod-POM-View-Restructured-0.02](#)
- [Pod-POM-2.00](#)

2. For each of the above Perl modules:

- **[as root]** Extract and build the Perl module

```
perl Makefile.PL
make
make install
```

- **[as non-root]** Extract and build the Perl module using `PREFIX` to specify a directory that you have write permission

```
mkdir ~/perl1lib
perl Makefile.PL PREFIX=~/perl1lib
make
make install
```

3. Execute the script `create_man_pages.py` to generate the `.rst` files into `xcat-core/docs` :

- **[as root]**

```
cd xcat-core
./create_man_pages.py
```

- **[as non root]**

```
cd xcat-core
./create_man_pages.py --prefix=~/perl1lib
```

## Debug Problems

### Tips

## 1.10 Need Help

xCAT is now on [GitHub](#)!

For support, we encourage the use of the [GitHub issues system](#).

- [xcat-core](#)
- [confluent](#)
- [documentation](#) issues can be opened against xcat-core project

The older email list is still available: [xcat-user@list.sourceforge.net](mailto:xcat-user@list.sourceforge.net)

## 1.11 Security Notices

### 1.11.1 2023 Notices

#### 2023-03-08 - xCAT Vulnerabilities

*Mar 8, 2023*, xCAT announced the following security advisory: <https://github.com/xcat2/xcat-core/security/advisories/GHSA-hpxg-7428-6jvv>

#### Advisory CVEs

- CVE-2023-27486 - **Insufficient authorization validation between zones when xCAT zones are enabled** (Severity: High)

Please see the security bulletin above for patch, upgrade, or suggested work around information.

#### Action

The issue described in CVE-2023-27486 only impacts users making use of the optional xCAT zones feature. xCAT zones are not enabled by default. Users making use of xCAT zones should upgrade to xCAT 2.16.5 or newer. Users that do not use xCAT zones are not impacted and do not need to upgrade.

### 1.11.2 2018 Notices

#### 2018-06-12 - OpenSSL Vulnerabilities

*Jun 12, 2018*, OpenSSL announced the following security advisories: <https://www.openssl.org/news/secadv/20180612.txt>

## Advisory CVEs

- CVE-2018-0732 - **Client DoS due to large DH parameter** (Severity: Low)

Please see the security bulletin above for patch, upgrade, or suggested work around information.

## Action

xCAT uses OpenSSL for client-server communication but **does not** ship it.

It is highly recommended to keep your OpenSSL levels up-to-date with the indicated versions in the security bulletins to prevent any potential security threats. Obtain the updated software packages from your Operating system distribution channels.

## 1.11.3 2017 Notices

### 2017-12-12 - TLS Vulnerabilities

*Dec 12, 2017*, TLS implementations may disclose side channel information via discrepancies between valid and invalid PKCS#1 padding

## Advisory CVEs

- CWE-203 - <http://cwe.mitre.org/data/definitions/203.html>

## Summary

Transport Layer Security (TLS) is a mechanism for a security transport over network connections, and is defined in RFC 5246. TLS may utilize RSA cryptography to secure the connection, and section 7.4.7 describes how client and server may exchange keys. Implementations that don't closely follow the descriptions in RFC 5246 may leak information to an attacker when they handle PKCS #1 v1.5 padding errors in ways that lets the attacker distinguish between valid and invalid messages. An attacker may utilize discrepancies in TLS error messages to obtain the pre-master secret key private RSA key used by TLS to decrypt sensitive data. This type of attack has become known as a Bleichenbacher attack. CERT/CC previously published CERT Advisory CA-1998-07 for this type of attack.

## Action

Consider the following recommended actions:

1. Disable TLS RSA
2. Apply an update (if available)

xCAT uses OpenSSL for client-server communication but **does not** ship it.

It is highly recommended to keep your OpenSSL levels up-to-date. Obtain the updated software packages from your Operating system distribution channels.

## 2017-12-07 - OpenSSL Vulnerabilities

*Dec 07, 2017*, OpenSSL announced the following security advisories: <https://www.openssl.org/news/secadv/20171207.txt>

### Advisory CVEs

- CVE-2017-3737 - **Read/write after SSL object in error state** (Severity: Moderate)

This issue does not affect OpenSSL 1.1.0.

OpenSSL 1.0.2 users should upgrade to 1.0.2n

- CVE-2017-3738 - **rsaz\_1024\_mul\_avx2 overflow bug on x86\_64** (Severity: Low)

Due to the low severity of this issue we are not issuing a new release of OpenSSL 1.1.0 at this time. The fix will be included in OpenSSL 1.1.0h when it becomes available. The fix is also available in commit e502cc86d in the OpenSSL git repository.

OpenSSL 1.0.2 users should upgrade to 1.0.2n

Please see the security bulletin above for patch, upgrade, or suggested work around information.

### Action

xCAT uses OpenSSL for client-server communication but **does not** ship it.

It is highly recommended to keep your OpenSSL levels up-to-date with the indicated versions in the security bulletins to prevent any potential security threats. Obtain the updated software packages from your Operating system distribution channels.

## 2017-08-28 - OpenSSL Vulnerabilities

*Aug 28, 2017*, OpenSSL announced the following security advisories: <https://www.openssl.org/news/secadv/20170828.txt>

### Advisory CVEs

- CVE-2017-3735 - **Malformed X.509 IPAddressFamily could cause OOB read** (Severity: Low)

If an X.509 certificate has a malformed IPAddressFamily extension, OpenSSL could do a one-byte buffer overread. The most likely result would be an erroneous display of the certificate in text format.

As this is a low severity fix, no release is being made. The fix can be found in the source repository (1.0.2, 1.1.0, and master branches); see <https://github.com/openssl/openssl/pull/4276>. This bug has been present since 2006.

Please see the security bulletin above for patch, upgrade, or suggested work around information.

## Action

xCAT uses OpenSSL for client-server communication but **does not** ship it.

It is highly recommended to keep your OpenSSL levels up-to-date with the indicated versions in the security bulletins to prevent any potential security threats. Obtain the updated software packages from your Operating system distribution channels.

### 2017-02-16 - OpenSSL Vulnerabilities

*Feb 16, 2017*, OpenSSL announced the following security advisories: <https://www.openssl.org/news/secadv/20170216.txt>

#### Advisory CVEs

- CVE-2017-3733 - **Encrypt-Then-Mac renegotiation crash** (Severity:High)

OpenSSL 1.1.0 users should upgrade to 1.1.0e

This issue does not affect OpenSSL version 1.0.2.

Please see the security bulletin above for patch, upgrade, or suggested work around information.

## Action

xCAT uses OpenSSL for client-server communication but **does not** ship it.

It is highly recommended to keep your OpenSSL levels up-to-date with the indicated versions in the security bulletins to prevent any potential security threats. Obtain the updated software packages from your Operating system distribution channels.

### 2017-01-27 - OpenSSL Vulnerabilities

*Jan 26, 2017*, OpenSSL announced the following security advisories: <https://www.openssl.org/news/secadv/20170126.txt>

#### Advisory CVEs

- CVE-2017-3731 - **Truncated packet could crash via OOB read** (Severity:Moderate)
- CVE-2017-3730 - **Bad (EC)DHE parameters cause a client crash** (Severity: Moderate)
- CVE-2017-3732 - **BN\_mod\_exp may produce incorrect results on x86\_64** (Severity: Moderate)
- CVE-2016-7055 - **Montgomery multiplication may produce incorrect results** (Severity: Low)

Please see the security bulletin above for patch, upgrade, or suggested work around information.

## Action

xCAT uses OpenSSL for client-server communication but **does not** ship it.

It is highly recommended to keep your OpenSSL levels up-to-date with the indicated versions in the security bulletins to prevent any potential security threats. Obtain the updated software packages from your Operating system distribution channels.

### 1.11.4 2016 Notices

#### 2016-11-30 - Removal of Service Stream Password

It has been brought to our attention that the xCAT product has hard-coded default passwords for the HMC/FSP to allow for IBM Service to connect to customer machines for L2/L3 support activities. This creates a security vulnerability where third parties could potentially gain root level access using these weak, hard coded passwords.

Example:

```
create_pwd => "netsDynPwdTool --create dev FipSdev",
password => "FipSdev"
```

In response, xCAT will remove these hard-coded password and interfaces from the xCAT code.

## Action

No action is required for xCAT 2.12.3, and higher.

If running older versions of xCAT, update xCAT to a higher level code base that has the hard-coded default passwords removed.

The following table describes the recommended update path:

xCAT Version	Action	Release Notes
<b>2.13</b> , or newer	No applicable	
<b>2.12.x</b>	Update to <b>2.12.3</b> , or higher	<a href="#">2.12.3 Release Notes</a>
<b>2.11.x</b>	Update to <b>2.12.3</b> , or higher	<a href="#">2.12.3 Release Notes</a>
<b>2.10.x</b>	Update to <b>2.12.3</b> , or higher	<a href="#">2.12.3 Release Notes</a>
<b>2.9.x</b> , or older	Update to: <ul style="list-style-type: none"> <li>• <b>2.9.4</b>, or higher for <b>AIX</b></li> <li>• <b>2.12.3</b>, or higher for <b>LINUX</b></li> </ul>	<a href="#">2.9.4 Release Notes</a>

#### 2016-08-24 - OpenSSL Vulnerabilities (Sweet32)

##### SWEET32: Birthday attacks against TLS ciphers with 64bit block size

##### CVE-2016-2183

Description: OpenSSL could allow a remote attacker to obtain sensitive information, caused by an error in the in the Triple-DES on 64-bit block cipher, used as a part of the SSL/TLS protocol. By capturing large amounts of encrypted traffic between the SSL/TLS server and the client, a remote attacker able to conduct a man-in-the-middle attack could exploit this vulnerability to recover the plaintext data and obtain sensitive information.

### CVE-2016-6329

Description: OpenVPN could allow a remote attacker to obtain sensitive information, caused by an error in the in the Triple-DES on 64-bit block cipher, used as a part of the SSL/TLS protocol. By capturing large amounts of encrypted traffic between the SSL/TLS server and the client, a remote attacker able to conduct a man-in-the-middle attack could exploit this vulnerability to recover the plaintext data and obtain sensitive information.

A detailed description of this issue can be seen in the following blog posting: <https://www.openssl.org/blog/blog/2016/08/24/sweet32/>

### Advisory CVEs

- CVE-2016-2183
- CVE-2016-6329

### Action

xCAT uses OpenSSL for client-server communication but **does not** ship it.

It is highly recommended to keep your OpenSSL levels up-to-date with the indicated versions in the security bulletins to prevent any potential security threats. Obtain the updated software packages from your Operating system distribution channels.

### 2016-08-16 - OpenSSL Vulnerabilities

This vulnerability has no fix available at this time (other then mentioned patches below)

Issue: [https://bugzilla.redhat.com/show\\_bug.cgi?id=1359615](https://bugzilla.redhat.com/show_bug.cgi?id=1359615)

Patch: <https://github.com/openssl/openssl/commit/0ed26acce328ec16a3aa635f1ca37365e8c7403a>

### Advisory CVEs

[CVE-2016-2180](#) - OpenSSL is vulnerable to a denial of service, caused by an out-of-bounds read in the TS\_OBJ\_print\_bio function.

### Action

xCAT uses OpenSSL for client-server communication but **does not** ship it. It is highly recommended to keep your OpenSSL levels up-to-date to prevent any potential security threats.



## 2016-05-03 - OpenSSL Vulnerabilities (ANS.1 encoder)

May 3, 2016 OpenSSL announced the following security advisories: <https://www.openssl.org/news/secadv/20160503.txt>

### Advisory CVEs

- CVE-2016-2108 - **Memory corruption in the ASN.1 encoder** (Severity:High)  
This issue affects all OpenSSL version prior to April 2015  
OpenSSL 1.0.2 users should upgrade to 1.0.2c OpenSSL 1.0.1 users should upgrade to 1.0.1o
- CVE-2016-2107 - **Padding oracle in AES-NI CBC MAC check** (Severity: High)  
This issue was introduced as part of the fix for Lucky 13 padding attack (CVE-2013-0169)  
OpenSSL 1.0.2 users should upgrade to 1.0.2h OpenSSL 1.0.1 users should upgrade to 1.0.1t
- CVE-2016-2105 - **EVP\_EncodeUpdate overflow** (Severity:Low)  
OpenSSL 1.0.2 users should upgrade to 1.0.2h OpenSSL 1.0.1 users should upgrade to 1.0.1t
- CVE-2016-2106 - **EVP\_EncryptUpdate overflow** (Severity:Low)  
OpenSSL 1.0.2 users should upgrade to 1.0.2h OpenSSL 1.0.1 users should upgrade to 1.0.1t
- CVE-2016-2109 - **ASN.1 BIO excessive memory allocation** (Severity:Low)  
OpenSSL 1.0.2 users should upgrade to 1.0.2h OpenSSL 1.0.1 users should upgrade to 1.0.1t
- CVE-2016-2176 - **EBCDIC overread** (Severity:Low)  
OpenSSL 1.0.2 users should upgrade to 1.0.2h OpenSSL 1.0.1 users should upgrade to 1.0.1t

### Action

xCAT uses OpenSSL for client-server communication but **does not** ship it.

It is recommended to keep your OpenSSL levels up-to-date with the indicated versions in the security bulletins to prevent any potential security threats.

## 2016-03-01 - OpenSSL Vulnerabilities (DROWN)

March 1, 2016 OpenSSL announced the following security advisories: <https://www.openssl.org/news/secadv/20160301.txt>

### Advisory CVEs

- CVE-2016-0800 - **Cross-protocol attack on TLS using SSLv2 (DROWN)** (Severity:High)  
This issue affects OpenSSL versions 1.0.1 and 1.0.2.  
OpenSSL 1.0.2 users should upgrade to 1.0.2g OpenSSL 1.0.1 users should upgrade to 1.0.1s
- CVE-2016-0705 - **Double-free in DSA code** (Severity:Low)  
This issue affects OpenSSL versions 1.0.2 and 1.0.1.  
OpenSSL 1.0.2 users should upgrade to 1.0.2g OpenSSL 1.0.1 users should upgrade to 1.0.1s

- CVE-2016-0798 - **Memory leak in SRP database lookups** (Severity:Low)

This issue affects OpenSSL versions 1.0.2 and 1.0.1.

OpenSSL 1.0.2 users should upgrade to 1.0.2g OpenSSL 1.0.1 users should upgrade to 1.0.1s

- CVE-2016-0797 - **BN\_hex2bn/BN\_dec2bn NULL pointer deref/heap corruption** (Severity:Low)

This issue affects OpenSSL versions 1.0.2 and 1.0.1.

OpenSSL 1.0.2 users should upgrade to 1.0.2g OpenSSL 1.0.1 users should upgrade to 1.0.1s

- CVE-2016-0797 - **Fix memory issues in BIO\_\*printf functions** (Severity:Low)

This issue affects OpenSSL versions 1.0.2 and 1.0.1.

OpenSSL 1.0.2 users should upgrade to 1.0.2g OpenSSL 1.0.1 users should upgrade to 1.0.1s

- CVE-2016-0702 - **Side channel attack on modular exponentiation** (Severity:Low)

This issue affects OpenSSL versions 1.0.2 and 1.0.1.

OpenSSL 1.0.2 users should upgrade to 1.0.2g OpenSSL 1.0.1 users should upgrade to 1.0.1s

- CVE-2016-0703 - **Divide-and-conquer session key recovery in SSLv2** (Severity:High)

This issue affected OpenSSL versions 1.0.2, 1.0.1l, 1.0.0q, 0.9.8ze and all earlier versions. It was fixed in OpenSSL 1.0.2a, 1.0.1m, 1.0.0r and 0.9.8zf

- CVE-2016-0704 - **Bleichenbacher oracle in SSLv2** (Severity:Moderate)

This issue affected OpenSSL versions 1.0.2, 1.0.1l, 1.0.0q, 0.9.8ze and all earlier versions. It was fixed in OpenSSL 1.0.2a, 1.0.1m, 1.0.0r and 0.9.8zf

## Action

xCAT uses OpenSSL for client-server communication but **does not** ship it. Regarding CVE-2016-0800, xCAT also does not use SSLv2 layer but uses the newer TLS transport layer security.

It is recommended to keep your OpenSSL levels up-to-date with the indicated versions in the security bulletins to prevent any potential security threats.

## 2016-01-28 - OpenSSL Vulnerabilities

Jan 28, 2016 OpenSSL announced the following security advisories: <https://mta.openssl.org/pipermail/openssl-announce/2016-January/000061.html>

## Advisory CVEs

- CVE-2016-0701 - **DH small subgroups** (Severity:High)

This issue affects OpenSSL version 1.0.2. OpenSSL 1.0.2 users should upgrade to 1.0.2f

- CVE-2015-3197 - **SSLv2 doesn't block disabled ciphers** (Severity:Low)

This issue affects OpenSSL versions 1.0.2 and 1.0.1.

OpenSSL 1.0.2 users should upgrade to 1.0.2f OpenSSL 1.0.1 users should upgrade to 1.0.1r

## Action

xCAT uses OpenSSL for client-server communication but **does not** ship it.

It is recommended to keep your OpenSSL levels up-to-date with the indicated versions in the security bulletins to prevent any potential security threats.

### 2016-01-15 - OpenSSL Vulnerabilities (SLOTH)

A detailed description of this issue can be seen in the following blog posting: <http://www.mitls.org/pages/attacks/SLOTH>

## Advisory CVEs

**CVE-2015-7575** - TLS 1.2 Transcript Collision attacks against MD5 in key exchange protocol (SLOTH)

## Action

xCAT uses OpenSSL for client-server communication but **does not** ship it.

It is highly recommended to keep your OpenSSL levels up-to-date with the indicated versions in the security bulletins to prevent any potential security threats. Obtain the updated software packages from your Operating system distribution channels.

Disable MD5 authentication in the cipher list using the site table keyword `xcatsslciphers`.

1. Check if MD5 is already disabled: `tabdump site | grep xcatssl`
2. If nothing is set, add `ALL:!MD5` to the cipher list: `chtab key=xcatsslciphers site.value='ALL:!MD5'`
3. Restart xcat: `service xcatd restart`

## 1.11.5 2015 Notices

### 2015-12-03 - OpenSSL Vulnerabilities

OpenSSL announced security fixes on 12/03/15 in the following bulletin: <http://openssl.org/news/secadv/20151203.txt>

## Advisory CVEs

- **CVE-2015-3193 - BN\_mod\_exp may produce incorrect results on x86\_64** (Severity:Moderate)  
OpenSSL 1.0.2 users should upgrade to 1.0.2e
- **CVE-2015-3194 - Certificate verify crash with missing PSS parameter** (Severity:Moderate)  
OpenSSL 1.0.2 users should upgrade to 1.0.2e OpenSSL 1.0.1 users should upgrade to 1.0.1q
- **CVE-2015-3195 - X509\_ATTRIBUTE memory leak** (Severity:Moderate)  
OpenSSL 1.0.2 users should upgrade to 1.0.2e OpenSSL 1.0.1 users should upgrade to 1.0.1q  
OpenSSL 1.0.0 users should upgrade to 1.0.0t OpenSSL 0.9.8 users should upgrade to 0.9.8zh
- **CVE-2015-3196 - Race condition handling PSK identify hint** (Severity:Low)

OpenSSL 1.0.2 users should upgrade to 1.0.2d  
OpenSSL 1.0.1 users should upgrade to 1.0.1p  
OpenSSL 1.0.0 users should upgrade to 1.0.0t

- CVE-2015-1794 - **Anon DH ServerKeyExchange with 0 p parameter** (Severity:Low)

OpenSSL 1.0.2 users should upgrade to 1.0.2e

### Action

xCAT uses OpenSSL for client-server communication but **does not** ship it.

It is recommended to keep your OpenSSL levels up-to-date with the indicated versions in the security bulletins to prevent any potential security threats.

### 2015-05-20 - OpenSSL Vulnerabilities (LOGJAM)

A Logjam vulnerability attacks openssl and web services on weak (512-bit) Diffie-Hellman key groups. Refer to the following documents for details.

Main site: <https://weakdh.org/>

Server test: <https://weakdh.org/sysadmin.html>

Refer to the following openssl link for more details regarding the fix: <https://www.openssl.org/blog/blog/2015/05/20/logjam-freak-upcoming-changes/>

OpenSSL 1.0.2 users should upgrade to 1.0.2b  
OpenSSL 1.0.1 users should upgrade to 1.0.1n

### Action

xCAT uses OpenSSL for client-server communication but **does not** ship it. It uses the default ciphers from openssl. It also allows the user to customize it through site.xcatsslversion and site.xcatsslciphers. Make sure you do not enable DH or DHE ciphers.

Get the latest openssl package from the os distros and upgrade it on all the xCAT management nodes, the service nodes and xCAT client nodes.

### 2015-05-19 - OpenSSL Vulnerabilities (VENOM)

#### Advisory CVEs

- CVE-2015-3456 - (aka **VENOM**) is a security flaw in the QEMU's Floppy Disk Controller (FDC) emulation.

VENOM vulnerability could expose virtual machines on unpatched host systems

A new vulnerability known as VENOM has been discovered, which could allow an attacker to escape a guest virtual machine (VM) and access the host system along with other VMs running on this system. The VENOM bug could potentially allow an attacker to steal sensitive data on any of the virtual machines on this system and gain elevated access to the host's local network and its systems.

The VENOM bug (CVE-2015-3456) exists in the virtual Floppy Disk Controller for the open-source hypervisor QEMU, which is installed by default in a number of virtualization infrastructures such as Xen hypervisors, the QEMU client, and Kernel-based Virtual Machine (KVM). VENOM does not affect VMware, Microsoft Hyper-V, and Bochs hypervisors.

- QEMU: <http://git.qemu.org/?p=qemu.git;a=commitdiff;h=e907746266721f305d67bc0718795fedee2e824c>
- Xen Project: <http://xenbits.xen.org/xsa/advisory-133.html>

- Red Hat: <https://access.redhat.com/articles/1444903>
- SUSE: <https://www.suse.com/support/kb/doc.php?id=7016497>
- Ubuntu: <http://www.ubuntu.com/usn/usn-2608-1/>

### Action

xCAT does not ship any rpms that have QEMU component directly. However xCAT does make system calls to QEMU when doing KVM/Xen visualization. If you are using xCAT to manage KVM or Xen hosts and guests, get the latest rpms that have QEMU component from the os distro and do a upgrade on both xCAT management node and the KVM/Xen hosts.

### 2015-04-03 - OpenSSL Vulnerabilities (BAR MITZVAH)

The RC4 Bar mitzvah attack is an attack on the SSL/TLS protocols when both the client and server have RC4 enabled.

- <http://www.darkreading.com/attacks-breaches/ssl-tls-suffers-bar-mitzvah-attack-/d/d-id/1319633>
- [http://www.imperva.com/docs/HII\\_Attacking\\_SSL\\_when\\_using\\_RC4.pdf](http://www.imperva.com/docs/HII_Attacking_SSL_when_using_RC4.pdf).

### Action

xCAT uses OpenSSL shipped with OS distribution for client-server communication. It does not use RC4 ciphers explicitly. However, it allows user to specify xcatsslciphers on the site table for ssl communication. It is recommended that the user not specify RC4 ciphers to avoid the Bar mitzvah attack.

It is also recommended that the user go to the OS distribution to get latest OpenSSL package for the fix of this problem.

### 2015-03-24 - OpenSSL Vulnerabilities

OpenSSL 1.0.2 introduced the “multiblock” performance improvement. This feature only applies on 64 bit x86 architecture platforms that support AES NI instructions. A defect in the implementation of “multiblock” can cause a segmentation fault within OpenSSL, thus enabling a potential DoS attack.

This issue affects OpenSSL version: 1.0.2

### Action

xCAT uses OpenSSL for client-server communication but **does not** ship it. Upgrade OpenSSL to 1.0.2a or higher.

### 2015-03-12 - OpenSSL Vulnerabilities (FREAK)

OpenSSL announced security fixes on 01/08/15 in the following bulletin: <https://www-origin.openssl.org/news/secadv/20150108.txt>

## **Advisory CVEs**

- CVE-2015-0204 **RSA silently downgrades to EXPORT\_RSA [Client]** (Severity: Low)

FREAK vulnerability CVE-2015-0204 is involved when 'RSA\_EXPORT' ssl cipher suit is used in ssl server/client.

## **Action**

xCAT does not use RSA\_EXPORT ciphers for ssl communication by default. However, xCAT does allow user to choose the ciphers from the site.xcatsslciphers attribute.

Make sure you do not put RSA\_EXPORT related ciphers in this attribute.

It is recommended that you upgrade openssl to 1.0.1L and upper version for the fix of this problem. Go to the os distribution to get the latest openssl package.