xCAT 2.0 Beta Release Cookbook

03/11/2008

1.1 Release Description

xCAT 2.0 is a complete rewrite of xCAT 1.2/1.3 implementing a new architecture (see description at end of this document). All commands are client/server, authenticated, logged and policy driven. The clients can be run on any OS with Perl, including Windows. The code has been completely rewritten in Perl, and table data is now stored in a relational database. For the beta, we are including SQLite with the xCAT OSS rpm. To use the new Service Node feature, you must be using Redhat 5 or Fedora 8 and install and setup the PostgreSQL Database. See instructions below in Chapter 1.10 "xCAT Hierarchy using Service Nodes chapter".

The code is being released as RPMs and SRPMs . For the beta release, there is support for x86_64 hardware (IPMI and Blades) and ppc64 hardware (js and qs blades) . The OS must be RedHat 5, CentOS5, Fedora 8 or SLES 10.

The beta code should not be used for production work

1.1.1 Function supported:

- Tools to manipulate the database tables: tabdump,tabrestore,tabedit, chtab , nodels, nodech, nodeadd, noderm, chdef, mkdef, lsdef, rmdef,
- Cluster setup commands: makehosts, makedhcp ,makeconservercf
- Notification commands (infrastructure allowing users to register for xCAT database table changes): regnotif, unregnotif
- Monitoring commands (monitoring plug-in infrastructure allowing plug-in third party monitoring software to the xCAT cluster): startmon, stopmon, lsmon.
- Hardware control commands : lsslp, rscan, rpower, reventlog, rinv, getmacs, rvitals
- Install commands : rnetboot
- Parallel remote and remote copy commands : xdsh, xdcp, xdshbak, psh. xdsh/xdcp is now packaged with xCAT.
- Node discovery and diskfull and diskless deployment of CentOS5 and RHELS5, Fedora 8 on the supported hardware (see Prerequisites).
- Hierarchical install (diskfull/diskless) using Service Nodes on Redhat 5 or Fedora 8.
- For a list of all 2.0 xCAT commands run ***rpm –ql xcat-client***.
- manpages
- Diskless/Stateless install
- Data abstraction commands to make creating node and other database definitions easier. See Chapter 1.5.3, “Using xCAT Object Definition Commands”.

1.1.2 Function not supported

- No imaging
- No flash
- pSeries hardware control using HMC, IVM, FSP for Power5 and Power6 hardware
- Web GUI interface

1.1.3 Prerequisites:

- Hardware requirements:
  - x3455, x3550, x3650, x3455, LS21, HS21, LS41, x336, x346, ppc64
  - no SOL for x386 or x486
  - Must be IPMI based, rack mounted unit.
  - Blades
  - Ethernet switch must be SNMP enabled for node discovery.
- Software supported
  - RedHat5, CentOS5, Fedora 8, SLES 10

1.1.4 Licensing

xCAT 2.0 is OSS with a EPL license. For license information visit

http://www.opensource.org/licenses/eclipse-1.0.php

1.2 Installing xCAT 2.0 Software

Install your xCAT management node with RedHat5, CentOS5, Fedora 8, SLES 10 making sure to install **all** packages available with the distribution to reduce the number of dependency RPMs you need to track down.

1. If you installed the xCAT2.0 alpha code, you should remove it from the system.
2. You can easily use YUM to install xCAT 2.0 on your management node.
   a. If your management node is connected to the internet, YUM can be pointed directly to the download site. Down load the following two files to the management node and place in the /etc/yum.repos.d directory.
      http://xcat.sourceforge.net/yum/xcat-core/xCAT-core.repo
      http://xcat.sourceforge.net/yum/xcat-oss/xCAT-oss.repo
   b. If your management node is not connected to the internet, download the following tar files to a system that is and then copy them to your management node:

      http://xcat.sourceforge.net/yum/core-repo.tar.bz2

      http://xcat.sourceforge.net/yum/oss-repo.tar.bz2

Un-tar the files. Each tar file has a mklocalrepo.sh script that you will need to run to modify the included copy of the xCAT-*.repo file to point to your local copies of the rpms.

3. Make sure that the management node hostname resolves to the ip address set in /etc/hosts. The output of “hostname –d” should print the correct domain name. If /etc/hosts is set with the long and short hostname, this should work.

For example: 7.113.47.250 rh5.clusters.com rh5 line in /etc/hosts results in

[root@rh5 ~]# hostname -d

clusters.com

4. If OpenIPMI-tools is installed on your system, remove it “rpm –e OpenIPMI-tools”. The version shipped with Redhat 5/CentOS5 is back-level and has many bugs. xCAT will be installing a newer version from the xCAT-oss.repo.
5. Now run “yum install xCAT” to install the xCAT and dependent OSS rpms. Postscripts in the rpms will set up default xCAT configurations and start the xcatd daemons.
6. If you are reinstalling xCAT 2.0, run “yum update” to update the xCAT packages.
7. Check to make sure the xcatd daemons started:

```
[root@rh5 xCAT-core]# ps -ef | grep -i xcat
root      3471     1  0 14:07 ?        00:00:00 xcatd: SSL listener
root      3472  3471  0 14:07 ?        00:00:00 xcatd: UDP listener
root      3473  3471  0 14:07 ?        00:00:00 xcatd: install monitor
```

   If not, start it manually:

```
[root@rh5 xCAT-core]# service xcatd start
Starting xCATd
```

8. Run “chkconfig tftp on” run “service xinetd restart” to enable the TFTP server.
9. Add xCAT manpages /opt/xcat/share/man to your MANPATH . You may need to export LC_ALL=en_US for the manpages to display correctly.

1.3 Syslog setup

The install of xCAT will automatically setup syslog.conf with the following entries to log errors to the /var/log/localmessages file. Severe errors from commands and errors from the xCAT will be logged in syslog. You can tailor the configuration but be sure and include the facility local4 which is used by xCAT.

```
/etc/syslog.conf
*.debug          /var/log/messages
*.crit           /var/log/messages
```

1.4 xCAT 2.0 Commands

Note: use '<xCAT command> -h' for a usage message from each command. MAN pages are not available at this time.

XCAT COMMAND	DESCRIPTION
chtab (Note: will be renamed to **tabch** in beta)	To add or update rows in a table. Allows you to add nodes, create groups, add attributes to the xCAT tables. chtab node=devnode01 nodelist.group=all,compute will add a new node devnode01 to the nodelist table and assign to the all and compute groups. chtab key=rsh site.value=/usr/bin/ssh will assign the site table rsh attribute to /usr/bin/ssh chtab –d node=devnode01 will delete the previously create node from the nodelist table.
copycds	Copies Linux distributions and service levels to install directories.
chdef	Change xCAT data object definitions.
chvm	Changes HMC- and IVM-managed partition profiles. (not available for use)
getmacs	Collect node MAX addresses
lsdef	Use this command to list xCAT data object definitions.
lsslp	Queries selected networked services configuration information.
lsmon	Lists monitoring plug-in module names, status and description.
lsvm	Lists partition profile information for HMC- and IVM-managed nodes. (not available for use)
makeconservercf	Make Conserver Configuration
mkdef	Use this command to create xCAT data object definitions.
makedhcp	Sets up the DHCP server.
makehosts	Creates entries in /etc/hosts for nodes. Node nodenames and ip addresses must be setup in the hosts table.
makenetworks	Builds the networks table
mkvm	Creates HMC- and IVM-managed partitions. (not available for use yet)
nodeadd	Add a node to the cluster

	For example: nodeadd <noderange> [table.column=value] [table.column=value]…. nodeadd blade1-blade7 nodelist.groups=all,compute nodeadd also supports some short cut tags: groups is equivalent to table.column = nodelist.groups • nodeadd blade1-blade8 groups=all,compute mgt is equivalent to table.column = nodehm.mgt • nodeadd blade7 mgt=blade switch is equivalent to table.colum= switch.switch • nodeadd blade8 switch=switch1
nodech	Change node information
nodels	Display information about a node or range of nodes or all nodes
noderm	Remove Node
nodeset	Installs, boots the nodes uses pxe.
psh	Runs a command across a list of nodes or nodegroups in parallel
rbeacon	Turns beacon on/off/blink or gives status of a node or a range of nodes.
rbootseq	For boot of Bladecenter node range. Change each node boot order.
regnotif	Register a Perl module or a command that will get called when changes occur in desired xCAT database tables. See Using xCAT Notification
reventlog	Retrieves or clears remote hardware event logs
rinv	Retrieves hardware configuration information for a single or range of nodes
rmdef	Use this command to remove xCAT data object definitions.
rmvm	Removes HMC- and IVM-managed partitions. (not supported yet)
rnetboot	Will force an unattended network install for a range of nodes (diskless) .
rpower	Boots, resets, powers on and off and queries nodes Note: "boot" option not implemented yet. Use either "on" or "reset" options as appropriate.
rscan	Collects node information from hardware control point

rsetboot	rsetboot (IPMI) is a way to specify the singular device to try to boot only for the next power cycle
rspreset	Used to reset service processors out-of-band
rvitals	Retrieves hardware vital information from the on-board Service Processor for a range of nodes
startmon	starts a monitoring plug-in module to monitor the xCAT cluster.
stopmon	stops a monitoring plug-in module to monitoring the xCAT cluster
tabdump	Display Database table information for table requested. tabdump with no input will display a list of all valid table names. tabdump –d <tablename> will list the fields of the table and their definitions
tabedit	Edit a table . Must export EDITOR to define your editor.
tabrestore	Restore a table from the table.csv template or from a tabdump output file.
unregnotif	Unregistered a Perl module or a command that was watching for changes of desired xCAT database tables.
xdcp	Concurrently copies files to/from multiple nodes. See xdsh and xdcp man page for more information
xdsh	Concurrently runs remote commands on multiple nodes. All dsh code is now shipped with xCAT 2.0. If dsh rpms were obtained from the following website and installed for the alpha release, you should erase the csm.dsh* rpm. http://www14.software.ibm.com/webapp/set2/sas/f/csm/download/home.html. See xdsh man page for more information.
xdshbak	Presents formatted output from the xdsh command

1.5 xCAT Tables

Note: The Database Table Schema can be viewed in the /usr/lib/perl5/site_perl/5.8.3/xCAT/Schema.pm file or by running the tabdump command.

TABLE NAME	DESCRIPTION
chain	Lists action that occur during node install, node boot . Used by nodeset.
deps	Dependency node table
hosts	List of hosts, alias hostname, ip addresses. Used to update /etc/hosts with makehosts
ipmi	Lists information on the nodes IPMI interface – bmc, username, password

iscsi	List information for setting up iSCSI
mac	Lists mac address for each node.
monitoring	Lists the monitoring plug-in module names that are monitoring the xCAT cluster.
monsetting	List the monitoring plug-in specific settings.
mp	This is the management processor network. Whereas the mpa.tab lists the adapter, this table lists devices that are networked off that adapter via daisy chained networks, or in the case of Blade Center, an internal network..
mpa	Lists the MPA, username and password for the nodes.
networks	Defines masks, gateways and DNS servers. Build my makenetworks command.
nodegroup	Lists information on all nodegroups defined
nodehm	Defines the hardware management method for each node.
nodelist	Defines all nodes and groups.
nodepos	Node physical location
noderes	Installation resources for the node.
nodetype	Node install type (osversion, arch, type)
notification	Lists the Perl modules and commands that will get called for changes in certain xCAT database tables.
osimage	Contains information that describes a unique operating system image that may be deployed on a cluster node
passwd	user names and passwords used by xCAT scripts
policy	Table controls the policy for the execution of the xcat commands.
postscripts	Comma separated list of scripts that should be run on this node after installation or diskless boot. (TBD)
ppc	Store Series p hardware components – HMC, IVM, BPA, FSP, LPAR
ppcdirect	Contains direct-attached FSP hardware information
ppchcp	Contains HMC and IVM hardware information
site	Main xCat configuration file. Holds global information for the cluster.
switch	Lists switch interface(s) for the node.

vpd	Vital product data table. Holds machine serial number and model type.

1.5.1 Table edit commands

To manage these tables directly, xCAT provides the **chtab, tabdump, tabrestore**, and **tabedit** commands.

The following are some basic examples of how to use the database table commands.

1. To see what tables exist in the xCAT database:

   ```
   tabdump
   ```

2. To display the definition of the attributes of the nodelist table:

   ```
   tabdump –d nodelist
   ```

3. To display the contents of the site table

   ```
   tabdump site
   ```

4. To back up all the xCAT tables.

   ```
   mkdir -p /tmp/xcatdb.backup
   ```

   ```
   for i in `tabdump`;do echo "Dumping $i..."; tabdump $i
   >/tmp/xcatdb.backup/$i.csv; done
   ```

5. Add a new node “devnode01” to the “nodelist” table and assign it to the “all” and “compute” groups.

   ```
   chtab node=devnode01 nodelist.group=all,compute
   ```

6. Assign the “site” table “rsh” attribute to “/usr/bin/ssh”.

   ```
   chtab  key=rsh  site.value=/usr/bin/ssh
   ```

7. Delete the previously created node from the “nodelist” table.

i) chtab –d node=devnode01 nodelist

8. To restore database tables that were dumped with tabdump:

   ```
   cd /tmp/xcatdb.backup
   ```

   ```
   for i in *.csv;do echo "Restoring $i..."; tabrestore $i; done
   ```

1.5.2 Using the node* commands

These are a set of commands for adding (nodeadd), changing (nodech), listing (nodels) and removing (noderm) from the database.

The following are some basic examples of how to use the node* commands:

1. To add a node to the nodelist table with groups all:
```
nodeadd sn1 nodelist.groups=all
```
2. To change the node sn1 os definition in the nodetype table:
```
nodech sn1 nodetype.os=rhel5
```
3. To remove node sn1 from all database tables:
```
noderm sn1
```
4. To list all the nodes in the input noderange:
```
nodels sn1-sn10
```

1.5.3 Using xCAT object definition commands

In addition to managing the database tables directly xCAT also supports the concept of data object definitions. Data objects are abstractions of the data that is stored in the xCAT database. This support provides a conceptually simpler implementation for managing cluster data. It is also more consistent with other IBM systems management products such as Director, CSM, and AIX/NIM etc. The attributes and values defined in the data object definitions will still be stored in the database tables defined for xCAT 2.0. These data object definitions should not limit experienced xCAT customers from managing the specific tables directly, if they so desire. A new set of commands is provided to support the object definitions. These commands will automatically handle the storage in and retrieval from the correct tables.

The following data object types are currently supported.

- **site** - Cluster-wide information. All the data is stored in the *site* table.
- **node** - Information for a specific cluster node. The data for a node is stored in multiple tables in the database. The commands that are provided to manage these definitions automatically figure out which attributes are stored in which table. It is therefore not necessary to keep track of a large number of table names and attribute locations.
- **network** - A description of a unique network. This data is stored in the *networks* table.
- **monitoring** - A description of a monitoring plugin. This data is stored in the *monitoring* table.
- **notification** - Defines the Perl modules and commands that will get called for changes in certain xCAT database tables. The data is stored in the *notification* table.

- **group** – Defines a set of nodes. A group definition can be used as the target set of nodes for a specific xCAT operation.   It can also be used to define node attributes that are applied to all group members. The group data is stored in multiple tables in the database.
- **policy** – Define the policies used when executing xCAT commands. The data is stored in the *policy* table.

There are four xCAT commands that may be used to manage any of the data object definitions.

- **mkdef** – Make data object definitions.
- **chdef** - Change data object definitions.
- **lsdef** - List data object definitions.
- **rmdef** - Remove data object definitions.

The following are some basic examples of how to use the database object definition commands. For more information on using these commands refer to the MAN pages.

1) To view the list of supported object definition types you can issue any of the commands with the "-h" option. Along with the usage you will also see a list of supported object types.
   lsdef –h

2) To get a description of the attributes that can be defined for each object type you can issue the **lsdef** command with the "-t <object type>" option.
   lsdef –h –t node

3) To get a list of all the objects currently defined.
   lsdef –a

4) To get the details of a specific node definition.
   lsdef -t node –l –o node01

5) To create a very simple node definition.
   mkdef –t node –o node02 groups="all,aix"

6) To create a node group containing all nodes that have a "nodetype" attribute set to "compute".
   mkdef -t group -o computenodes -w nodetype= compute

7) To change the site definition.
   chdef -t site -o clustersite rsh=/bin/rsh rcp=/bin/rcp installdir=/xcatinstall

8) To remove all node and group definitions.
   rmdef –t node,group

9) To remove the group called hmcnodes.

```
rmdef -t group -o hmcnodes
```

In addition to the standard command line input and output the **mkdef**, **chdef**, and **lsdef** commands support the use of a stanza file format for the input and output of information. Input to a command can be read from a stanza file and the output of a command can be written to a stanza file. A stanza file contains one or more stanzas that provide information for individual object definitions. For example:

5. To create a set of definitions using information contained in a stanza file.
   ```
   cat mystanzafile  | mkdef  -z
   ```

6. To write all node definitions to a stanza file.
   ```
   lsdef –t node -l -z > nodestanzafile
   ```

The stanza file support also provides an easy way to backup and restore the cluster data.

For more information on the use of stanza files see the **xcatstanzafile** MAN page.

Note: In some cases the object definition commands may not be able to recognize changes that were made by updating the database tables directly by using the table commands. Generally speaking, the intermixing of the use of the two sets of commands is not recommended.

1.6 Using xCAT hardware commands

1.6.1 Hardware discovery

The following commands can be used to gather information about cluster hardware. See the MAN pages for additional details.

1. **rinv** - Retrieves hardware configuration information for a single or range of nodes and groups.

   For example:

   ```
   rinv node5 all

       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 Controller 0 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: 2
node5: Processor Speed: 866 MHz
node5: Total Memory: 512 MB
node5: Memory DIMM locations: Slot(s) 3 4
```

2. **rvitals** - Retrieves hardware vital information for a single or range of nodes and groups.

   For example:

   ```
   rvitals node5 all

       node5: Frame Voltage (Vab): 201V
       node5: Frame Voltage (Vbc): 203V
       node5: Frame Voltage (Vca): 202V
       node5: Frame Current  (Ia): 19A
       node5: Frame Current  (Ib): 19A
       node5: Frame Current  (Ic): 20A
       node5: System Temperature: 33 C (91.4 F)
       node5: Running
   ```

3. **lsslp** - Queries selected networked services information within the same subnet. If the HMC/IVM that you are interested in discovering is on the same subnet as your Management Node, you can run the **lsslp** to discover and add his hardware to the xCAT database.

   Note that the dependent programs **slp_query** and **libslp_client.so** are compiled modules required to perform SLP broadcasts. These modules can be obtained by posting a request to the xCAT mailing list (please specify the target O/S in the request).

   For example:

   ```
   lsslp -s HMC

       device type-model serial-number          ip-addresses           hostname

       HMC    7310CR2    103F55A       1.1.1.115 2.2.2.164 3.3.3.102  hmc01
   ```

```
HMC    7310CR2    105369A       3.3.3.103 2.2.2.103 1.1.1.163   hmc02
HMC    7310CR3    KPHHK24       3.3.3.154 2.2.2.110 1.1.1.154  hmc03
```

4. **rscan** - Collects node information from one or more hardware control points.

   For example:

   ```
   rscan hmc01

      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
   ```

5. **getmacs** – Gathers adapter MAC information from cluster nodes.
   For example:

   ```
   getmacs node01

        lpar4:

        #Type  Location Code   MAC Address  Full Path Name Ping Result

        ent U9133.55A.10B7D1G-V12-C4-T1 8ee2245cf004 /vdevice/l-
        lan@30000004  virtual
   ```

1.6.2 Hardware Control

The following commands can be used to control cluster hardware. See the MAN pages for additional details.

7. **rnetboot** – Initiate a network boot request on one or more cluster nodes.

   For example, to initiate a network boot of the node “node01”, enter:
   rnetboot node01

8. **rpower** – Boots, resets, powers on and off, and queries node hardware, and devices.
   For example, to power on a node, enter:
   rpower -n clsn04 on

1.7 Adding and Installing Nodes

1) Check the default required site table attributes:

```
[root@rh5 xCAT-core]# tabdump site
#key,value,comments,disable
"xcatdport","3001",,
"xcatiport","3002",,
"master","9.114.47.251",,
"domain","ppd.pok.ibm.com",,
"installdir","/install",,
"timezone","America/New_York",,
"nameservers","176.60.50.209",,
```

   To change any of these values, use chtab or tabedit.
   chtab:
   a) chtab key=domain site.value=<your domain name>
      For example: chtab key=domain site.value=clusters.com
   b) chtab key=master site.value=<ip address on the cluster network of Master node>
      For example: chtab key=master site.value=8.777.43.5
   c) chtab key=dhcpinterfaces site.value=<comma delimited list of nics to run dhcp>
      For example: chtab key=dhcpinterfaces site.value=eth1

   tabedit:
   a) export EDITOR=vim (or your favorite editor)
   b) tabedit site
   c) make your changes, and use the editor command to save the file and quit. Your changes will automatically be imported into the xCAT database.

2) Change the xCAT passwd table to set the default root password when the node is installed:
      chtab key=system passwd.username=root passwd.password=cluster
3) Check the 1350 default database template files in /usr/share/xcat/template/e1350 directory to see if they apply to your environment. These templates, or templates you create from them, can be used to load the database xCAT tables using the ***tabrestore <path to template>*** command. The README, in the directory, explains how to use these files.
4) The ***tabdump <tablename>*** will dump current contents of the database table. This can be used to dump the contents of a table and, if you redirect the output to a file, you can later reload the data using ***tabrestore***.
5) Use ***tabedit <tablename>*** to make any needed changes to the tables. Check the previous released xCAT tables for definitions. The 2.0 tables contain a header with the format of the fields in comments.
6) Define the nodes in your cluster by using the nodeadd command. Ensure that all nodes, bmcs or management modules, and switches have hosts definitions, or the dhcp configuration will not update, and the bmcsetup will not receive meaningful data. (see nodeadd command in the xCAT Tables).

7) If you want ***makehosts*** to update the /etc/hosts file for the defined nodes, bmcs/mms, and switches, use tabedit to update the ***hosts*** table with the hostnames and ip addresses to be added to /etc/hosts. Then run ***makehosts .***
8) ***makenetworks*** runs during the xCAT install and updates the networks table. You should ***tabdump networks*** to ensure the setting are correct. If any need changing, ***tabedit networks*** table. Ensure the networks to be managed have the "dynamicrange" set to a hyphenated range of IP addresses to serve as staging for nodes being brought up**.** If any new networks are added, the ***makenetworks*** should be run again.
9) Run ***makedhcp –n***. Review the /etc/dhcp.conf file created to ensure all your network definitions are correct. Note that the node host definitions will no longer appear here, but rather will appear in the leases file (/var/lib/dhcpd/dhcpd.leases) after the initial DHCP request from the node. xCAT 2.0 sets up dhcp to use the OMAPI command shell to setup, query and change the dhcp configuration. See **man omshell**, and http://linux.die.net/man/3/omapi for more information.
10) Run "service dhcpd start" to load the initial omapi dhcp configuration.
11) For blades, make sure your bladecenter management module is configured for the SNMP protocol:
    a) Telnet into you management module. Once in, do the following (assumes "mm[1]" is the current active mm and "PASSW0RD" is your mm password).
    b) env -T mm[1]
    c) users -1 -ap sha -pp des -at set -ppw PASSW0RD
    d) Log off the management module and test the connection with a query command such as ***rpower <noderange> stat*** or ***rinv <noderange> all***.

    Note: This was only tested with the latest release level firmware BPET32D. Older firmware may not properly support SNMP.
12) Set up conserver.
    a) Update the nodehm table (***tabedit nodehm***) to set fields for **cons**, **termport**, and **termserver** for your nodes. Currently, supported values for **cons** are "blade" and "ipmi".
    b) Run makeconservercf to generate a conserver 8 configuration file. Review /etc/conserver.cf. Make sure you have valid "trusted" entries in the "access{}" stanza for any host starting a console (most likely your management node).
    c) Start the conserver daemon: service ***conserver start***
    d) Try opening a console: ***console -M <management node> <node>***
13) xCAT 2.0 will discover your hardware:
    a) Create the initrd:
        (1) rm /tftpboot/pxelinux.cfg/default
        (2) ***mknb x86_64*** (creates the netboot image and writes out the master parameter to the /tftpboot/pxelinux.cfg/default file).
    b) Make sure your boot sequence is set to boot from network before harddrive: ***rbootseq <noderange> list***

        If not, change it: ***rbootseq <noderange> f,c,n,h***
    c) Power up the system using ***rpower <noderange> on.***
    d) Within a few seconds of booting to the network, any BMCs should be configured and be setup to allow ssh. All nodes will be network booted (you can watch /var/log/messages for DHCP and TFTP traffic).

e) ***nodels <noderange> vpd.serial vpd.mtm mac.mac*** should show interesting data after discovery.

14) Run ***copycds*** with full path to the ISO images
15) Run ***nodech*** (or ***tabedit***) to change nodetype OS and setup node profile :
    nodech <noderange> nodetype.os=<os> nodetype.profile=compute
    (for now only, the **compute** template file has been provided. See /usr/share/xcat/install/). Current possible values for **os**: rhels5, rhelc5,centos5
    If using 64 bit distro, the **nodetype.arch** should have been populated with "x86_64" at discovery time. If not, set this value, too. This is the only architecture supported for now.
16) Run ***nodech*** (or ***tabedit***) to set noderes nfsserver :
    nodech <noderange> noderes.nfsserver=<server>
    (Note: may need to use your management server IP address instead of the hostname for the nfsserver for now)
17) Also check the following fields to make sure they are set correctly and update as necessary:
    noderes.installnic -- the Ethernet adapter on the node used for installation
    noderes.serialport -- standard SOL for Blades "1", for IPMI nodes "0"
    nodehm.serialspeed -- standard SOL for Blades "19200"
    nodehm.serialflow -- standard SOL for Blades "hard"
18) Postscripts that will be run during node install are identified in **/etc/xcat/postscripts.rules** and located in **/install/postscripts**. Not all of the postscripts have been ported to xCAT 2.0 yet, so you may get some "script not found" messages during the postscript processing. Also, the ***postage*** and ***postrules*** commands have not been ported yet, so debug may take a little more effort.
19) Run makedhcp <noderange> to setup dhcp for the install.
20) Run ***nodeset <noderange> install***, to setup for installing the OS.
21) Run ***rpower <noderange> on or rpower <noderange> reset***, to boot the systems and start the network install process.
    (note, ***rpower <noderange> boot*** is not working yet)
    - ❖ The kexec to installers doesn't have the client scripts written yet, necessitating the reboot, if wanting to try kexec for now, you have to manually transfer the kernel, initrd, and run kexec -f with the right arguments to the xCAT nbfs environment)

1.8 Using xCAT Notification Infrastructure

With xCAT 2.0, you can monitor xCAT database for changes such as nodes entering/leaving the cluster, hardware updates, node liveness (to be added later) etc. In fact anything stored in the xCAT database tables can be monitored through the xCAT notification infrastructure. To start getting notified for changes, simply register your Perl module or command as the following:

regnotif ***filename tablename -o actions***

where

filename is the full path name of your Perl module or command.

tablenames is a comma separated list of table names that you are interested in.

actions is a comma separated list of data table actions. 'a' for row addition, 'd' for row deletion and 'u' for row update.

Example**:**

regnotif /opt/xcat/lib/perl/xCAT_monitoring/mycode.pm nodelist,nodhm -o a,d

regnotif /usr/bin/mycmd switch,noderes -o u

Use the following command to view all the modules and commands registered.

tabdump notification

To un-register, just do the following:

unregnotif ***filename***

Example:

unregnotif /opt/xcat/lib/perl/xCAT_monitoring/mycode.pm

unregnotif /usr/bin/mycmd

If the *filename* specifies a Perl module, the package name must be **xCAT_monitoring::xxx**. It must implement the following subroutine which will get called when database table change occurs:

processTableChanges*(tableop, table_name, old_data, new_data)*

where:

tableop Table operation. It can be 'a' for row addition, 'd' for row deletion and 'u' for row update.

tablename The name of the database table whose data has been changed.

old_data An array reference of the old row data that has been changed. The first element is an array reference that contains the column names. The rest of the elements are array references each contains attribute values of a row. It is set when the action is u or d.

new_data A hash reference of the new row data; only changed values are in the hash. It is keyed by column names. It is set when the action is u or a.

If the file name specifies a command (written by any programming languages or scripts), when the interested database table changes, the info will be fed to the command through the standard input. The format of the data in the STDIN is as following:

action(a, u or d)

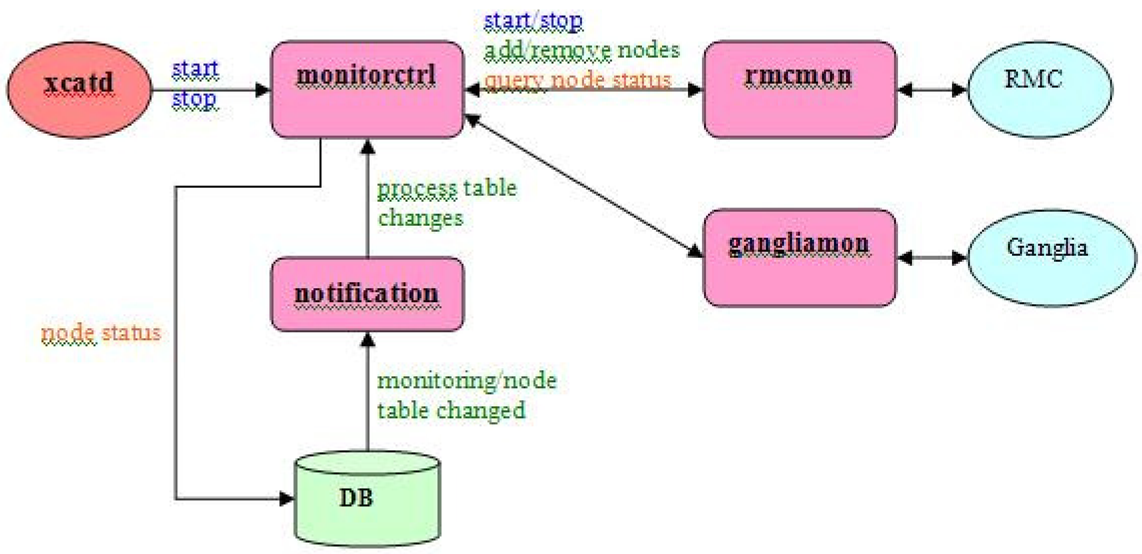
tablename

```
[old value]
col1_name,col2_name...
col1_val,col2_val,...
col1_val,col2_val,....
...
[new value]
col1_name,col2_name,...
col1_value,col2_value,...
...
```

The sample code can be found under **/opt/xcat/lib/perl/xCAT_monitoring/samples/mycode.pm** on a installed system.

1.9 Using xCAT Monitoring Plug-in Infrastructure:

With xCAT 2.0, you can integrate 3rd party monitoring software into your xCAT cluster. The idea is to use monitoring plug-in modules that act as bridges to connect xCAT and the 3rd party software. The functions of a monitoring plug-in module include initializing the 3rd party software, informing it with the changes of the xCAT node list, setting it up to feed node status back to xCAT etc. The following figure depicts the relationship and data flow among xcatd, plug-in modules and 3rd party monitoring software.



To use this infrastructure, first create a monitoring plug-in module and put it under **/opt/xcat/lib/perl/xCAT_monitoring/** directory. If the file name is xxx.pm then the package name will be **xCAT_monitoring::xxx**. The following is a list of subroutines that a plug-in module must implement:

start

stop

supportNodeStatusMon

startNodeStatusMon

stopNodeStatusMon

addNodes

removeNodes

processSettingChanges

getDiscription

Please refer to */opt/xcat/lib/perl/xCAT_monitoring/samples/tmplatemon.pm* for the detailed description of the functions.

Second, register the module in xCAT ***monitoring*** table using the following command:

startmon *name [-n|--nodestatmon] [-s|--settings settings]*

where

name is the monitoring plug-in module short file name without the extension. In this case xxx. Use ***lsmon –a*** command to list all the possible plug-in module names.

-n or --nodestatmon indicates it can help feeding the node status to xCAT. The node status is stored in the ***status*** column of the ***nodelist*** table.

-s or –settings specifies the plug-in specific settings. These setting will be used by the plug-in to customize certain entities for the plug-in or the third party monitoring software. The format of the setting string is: [key=value],[key=value]… Please note that the square brackets are needed. e.g. [mon_interval=10],[toggle=1]

Example**:**

startmon *xxx –n* (with feeding the node status to xCAT table)

or

startmon *xxx* (not feeding the node status to xCAT table)

Once it is registered, xCAT will automatically, through the plug-in module, start the 3rd party software for monitoring. To unregister the monitoring plug-in and stop the monitoring use this command:

stopmon *name*

Example**:**

stopmon *xxx*

Though you can write your own monitoring plug-in modules, over the time, xCAT will supply a list of built-in plug-in modules for the most common monitoring software. They are:

- xCAT (xcatmon.pm)   (released in this beta)
- RMC (rmcmon.pm)
- Ganglia (gangliamon.pm)
- Nagios (nagiosmon.pm)
- SNMP (snmpmon.pm)

xcatmon.pm is included in this release. It provides node liveness monitoring using fping. This can be used if no other 3rd party software is used for node status monitoring. The ***status*** column of the ***nodelist*** table will be updated periodically with the latest node liveness status by this plug-in. To activate, use the startmon command:

startmon xcatmon –n –s [ping-interval=2]

where 2 means that the nodes are pinged for status every 2 minutes**.**

1.10 xCAT Hierarchy using Service nodes

In large clusters it is desirable to have more than one node (the Management Server) handle the installation of the compute nodes.   We call these nodes service nodes. You can have one or more service nodes setup to install groups of compute nodes.

The service nodes need to communicate with the xCAT2.0 database on the Management Server and run xCAT command to install the nodes.   The service node will be installed with the xCAT code and we required the PostgreSQL Database be setup instead of SQLite. PostgreSQL allows a client to be setup on the service node such that the service node can access the database on the Management Server (Masternode).

1.10.1 Setting up PostgreSQL

The following rpms should be installed from the RHELS5.1/Fedora8 media on the Management Server (and service node when installed).
perl-DBD-Pg-1.49-1.fc6
postgresql-server-8.1.9-1.el5
postgresql-8.1.9-1.el5

To setup the database on the Management Server follow this steps.
This example assumes:
192.168.0.1: ip of master
xcatdb: database name
xcatadmin: database role (aka user)
cluster: database password
192.168.0.10 & 192.168.0.11 (service nodes)
Substitute your address and desired userid and database name as appropriate

1. service postgresql initdb
   Initializing database: [OK]

2. /etc/init.d/postgresql start
    Starting postgresql service: [OK]
3. su – postgres
4. -bash-3.1$ createuser -P xcatadmin
   Enter password for new role: cluster
   Enter it again: cluster
   Shall the new role be a superuser? (y/n) n
   Shall the new role be allowed to create databases? (y/n) n
   Shall the new role be allowed to create more new roles? (y/n) n
   CREATE ROLE
5. $ createdb -O xcatadmin xcatdb
   CREATE DATABASE
6. $ exit
   logout
7. cd /var/lib/pgsql/data/
8. vi pg_hba.conf
   #lines should look like:
   local all all ident sameuser
9. IPv4 local connections:
   host all all 127.0.0.1/32 md5
   host all all 192.168.0.1/32 md5
   host all all 192.168.0.10/32 md5
   host all all 192.168.0.11/32 md5 where 192.168.0.10 and 11 are service nodes.
10. vi postgresql.conf
11. set listen_addresses to '*':
    listen_addresses = '*'   This allows remote access
12. service postgresql restart
13. (if trying to save an existing configuration/migrate)
    #mkdir -p ~/xcat-dbback
14. cd /etc/xcat
    for i in *sqlite; do
    tabdump ${i%%.sqlite}> ~/xcat-dbback/$i.csv   done
15. /etc/sysconfig/xcat should contain this, substitute your cluster facing address for 192.168.0.1, and user and password are xcatadmin cluster in this instance
    XCATCFG='Pg:dbname=xcatdb;host=192.168.0.1|xcatadmin|cluster'
    export XCATCFG
    XCATROOT=/opt/xcat
    export XCATROOT
16. copy /etc/sysconfig/xcat /install/postscripts/sysconfig/xcat
17. chmod 700 /etc/sysconfig/xcat #only root should be able to read due to passwd
18. . /etc/sysconfig/xcat   #read the text into the current shell
19. You . /etc/sysconfig/xcat to a setup shell script in /etc/profile.d, so the XCATROOT and XCATCFG environment variables are setup when you login.
20. Add this line to /etc/profile.d/xcat.sh : export XCATCFG="Pg:dbname=xcatdb;host=9.114.47.227|xcatadmin|cluster" with your database name, admin and password substituted.

21. Now initialize the database
    - chtab key=xcatdport site.value=3001
      chtab key=xcatiport site.value=3002
      chtab key=master site.value=$(getent hosts `hostname`|awk '{print $1}')
      chtab key=tftpdir site.value=/tftpboot
      chtab key=domain site.value=$(hostname -d)
      chtab key=installdir site.value=/install
      chtab key=timezone site.value=`grep ^ZONE /etc/sysconfig/clock|cut -d= -f 2|sed -e 's/"//g'`
      chtab priority=1 policy.name=root policy.rule=allow
      chtab priority=2 policy.commands=getbmcconfig policy.rule=allow
      chtab priority=3 policy.commands=nextdestiny policy.rule=allow
      chtab priority=4 policy.commands=getdestiny policy.rule=allow
      Setup your xCAT servers in the site.tab table
      chtab key=xcatservers site.value=sn1,sn2,.......
22. service xcatd restart
23. Restore your saved database: cd ~/xcat-dbback
     for i in *csv; do
    tabrestore $i
    done
24. chkconfig postgresql on
25. Need to update the policy table: Run this command to get correct Master node name known by ssl:
    - openssl x509 -text -in /etc/xcat/cert/server-cert.pem -noout|grep Subject

```
Subject: CN=mgt.cluster
Subject Public Key Info:
X509v3 Subject Key Identifier:
```

26. Update the policy table with mgt.cluster output from the command:
    chtab priority=5 policy.name=<mgt.cluster> policy.rule=allow
27. service postgresql restart

1.10.2 Defining your service nodes

The noderes table defines which service node will service the nodes in your cluster.
For each of the nodes in the cluster, change the service node attribute in the noderes table to point to the name or ip address of it's service node. So for nodes node1-node25, setup the service node sn1. Then assign the service that you would like run on the service node.

Define nodes:

- nodeadd node1-node25 nodelist.groups=compute,all

Define service nodes and as nodes and in the site table

- nodeadd sn1-sn2 nodelist.groups=service,all
- chtab key=xcatservers site.value=sn1,sn2

Assign service node to the node group

- nodech node1-node25 noderes.servicenode=sn1

Define services to run on the servicenode for the node group.

- nodech node1-node25 noderes.tftpserver=sn1
- chtab netname=extnet networks.dhcpserver=78.44.66.1 (ip address of sn1)

Note: if in the noderes table you have an assigned servicenode for a node, and the field for the service (e.g nfsserver) is left blank, it is assumed that you want that service running on the service node. So you can either explicitly assign a service node to a node for any given service, or you can leave the fields blank and the service node assigned to the node will run all services for that node.

For example: [root@xcat20mn bin]# tabdump noderes

```
#node,servicenode,netboot,tftpserver,nfsserver,monserver,kernel,initrd,kcmdline,nfsdir,serialport,installnic,primarynic,xcatmaster,current_osimage,next_osimage,comments,disable

"mynode","sn1","pxe","sn1",,,,,,,,,,,,,
```

Here sn1 is the servicenode for mynode. sn1 is the tftpserver for mynode because it was explicitly set. sn1 is also the nfsserver, because no other service node was put in that field. The settings for the services in the database will determine which services are setup on the service node. These services are setup when the xcatd daemon is started on the service node.

The services that are setup by xCAT on the service node are as follows:

- nfs (always setup)
- dns
- conserver
- tftp
- http (automatically installed)
- dhcp
- syslog (always setup)

1.10.3 Installing xCAT Service Nodes (diskfull)

Before installing make sure that at least one node in the database has the service node you are going to installed defined as its service node.

Follow the normal steps for an OS install, see 1.7 " Adding and Installing Nodes".

In addition we need to install the xCAT rpms and dependencies on the Service Node:

- Create a directory /install/postscripts/xcat/RPMS/noarch
- Create a directory /install/postscripts/xcat/RPMS/x86_64
- The following rpms should be in /install/postscripts/xcat/RPMS/noarch
    - perl-Expect-1.20-1.noarch.rpm
    - perl-xCAT-2.0-*.rpm
    - xCAT-client-2.0-*.rpm
    - xCAT-nbkernel-x86_64-2.6.18_8-*.noarch.rpm
    - xCAT-nbroot-core-x86_64-2.0-*.noarch.rpm
    - xCAT-nbroot-oss-x86_64-2.0-*.noarch.rpm
    - xCAT-server-2.0-*.noarch.rpm
- The following rpms should be in /install/postscripts/xcat/RPMS/x86_64
    - atftp-0.7-1.x86_64.rpm
    - atftp-client-0.7-1.x86_64.rpm
    - atftp-debuginfo-0.7-1.x86_64.rpm
    - conserver-8.1.16-2.x86_64.rpm
    - conserver-debuginfo-8.1.16-2.x86_64.rpm
    - fping-2.4b2_to-2.x86_64.rpm
    - ipmitool-1.8.9-2.x86_64.rpm
    - ipmitool-debuginfo-1.8.9-2.x86_64.rpm
    - perl-IO-Tty-1.07-1.x86_64.rpm
    - xCATsn-2.0-*.x86_64.rpm

Run nodeset <service nodename>. Check /install/postscripts/<service nodename> to see if the servicenode is one of the postinstall scripts that will be run. If not, make sure you have a node defined in the database with this service node designated as its service node in the noderes table.

rpower <service nodename> off

rpower <service nodename> stat - do this until you see status is off

rpower <service nodename> on

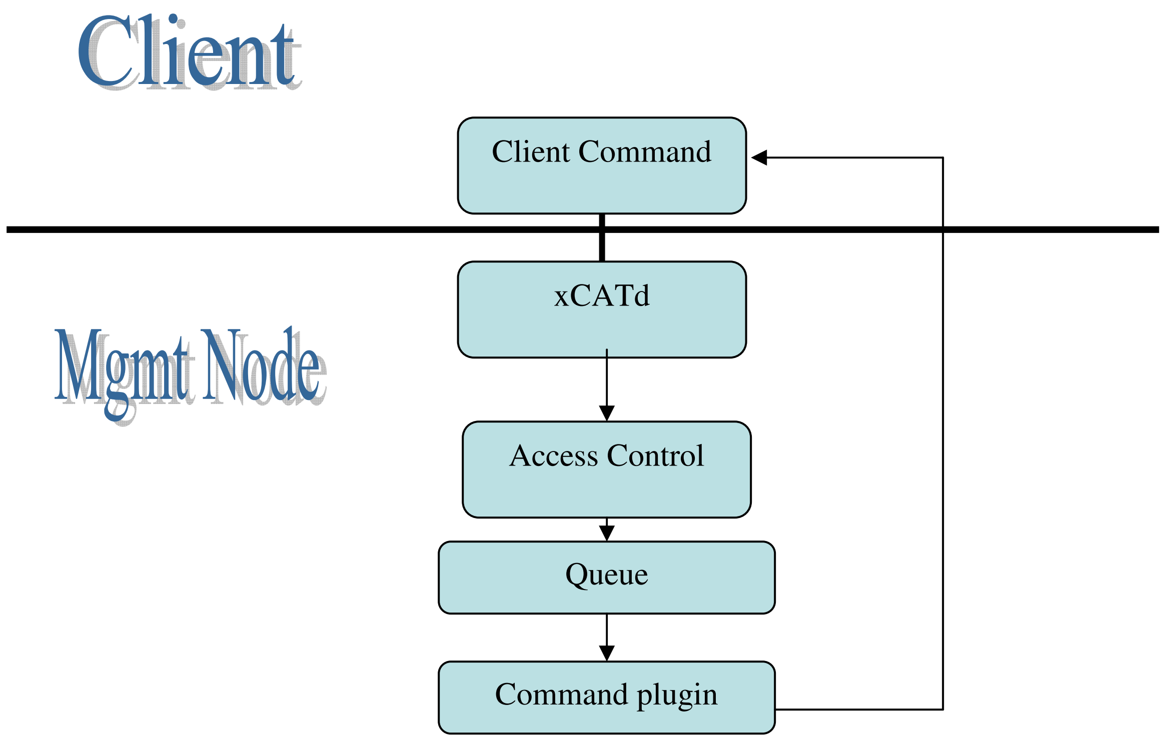
You can monitor progress with nodestat < service nodename> and tailing /var/log/messages

When the service node comes up, it should have services (tftp, dhcp, dns, nfs, conserver) started, you should be able to run run database command like nodels or tabdump on the service node.

1.11 xCAT Architecture

General/Overall Concepts

The heart of the xCAT 2.0 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 information from the nodes



1.11.1 Client/Server

1.11.2 Flow

- User invokes an xcat cmd on the client
- The cmds can either be a sym link to xcatclient or a thin wrapper that calls xcatclient.
- Some cmds will implement their own xcatclient function, if they have more processing than the generic xcatclient function supports. (e.g. xdsh/xdcp).
- The xcatclient function packages the info into xml and passes it to xcatd
- xcatd receives the request and forks to process the request
- The ACL/Role Policy Engine determines whether this person is allowed to execute this request. It evaluates the following info:

    - The cmd name and args
    - Who executed the cmd on the client machine
    - The hostname/IP address of the client machine
    - The node range passed to the cmd
- If the ACL check is approved, the cmd is passed to the Queue:
    - The queue can run the action in either of 2 modes. The client cmd wrapper decides which mode to use (although it can give the user a flag to specify):
        - Keep the socket connection with the client open for the life of the action and continue to send back the output of the action as it is produced.
        - Initiate the action, pass the action ID back to the client, and close the connection. At any subsequent time, the client can use the action ID to request the status and output of the action. This is intended long running cmds.
    - The Queue logs every action performed, including date/time, cmd name, arguments, who, etc.
    - In phase 2, the Queue will support locking (semaphores) to serialize actions that should not be run simultaneously.
- To invoke the action, the xml is passed to the process_request() function of the appropriate plugin pm which contains the code for the function being run.
    - With the request examined per policy table, and noderange expanded to nodes, the request is passed in its entirety (including tags otherwise ignored) to a plugin's process_request function, which will receive two arguments, the first the aforementioned hash reference, the second a reference to a callback function to call per response message to send back.
    - The appropriate pm is chosen by loading all of the plugins from /usr/lib/xcat/plugins and invoking handled_commands to see which cmds each pm handles.
    - Data is returned from the command plugin back to the client command handle_response routine.