CLUSTER HEALTH CHECK

User Manual

Version 1.1.0.0

Revision 1.1

January 3, 2014

TABLE OF CONTENTS

ABOUT....................3

1.0 OVERVIEW....................3

1.1 Requirements....................3
1.2 Installation....................3
1.3 Configuration....................4
1.3.1 Tool configuration files....................5
1.3.2 Group configuration files....................8
1.4 Tool output processing methods....................9

2.0 FRAMEWORK....................10

2.1 The hcrun command....................11
2.2 The config_check command....................21
2.3 Using config_check and hcrun to run health checks....................23

3.0 GUIDELINES FOR ADDING NEW TOOLS....................28

4.0 REFERENCE....................29

About

To determine the health of a cluster, it is necessary to get the current state of all the components which build up the cluster. In most installations these are:

- Compute nodes
- Ethernet network
- InfiniBand network
- Storage

In this document, we describe the usage of the Cluster Health Check (CHC) framework which is used to perform checks on these components. By working with results of these checks, we will be able to state if the cluster is healthy or not.

1.0 Overview

The Cluster Health Check framework provides an environment for integrating and running a subset of individual checks which can be performed on a single node or a group of nodes. The tests are categorized into different tool types and groups, so that only checks for a specific part of the cluster can be performed. Because the framework and toolset is extensible. users can also create their own groups and tools for checking different components of the cluster.

The main wrapper tool is **`hcrun`**, which provides access to all tools and passes them the environment.

1.1 Requirements

The cluster health check framework with toolkit should be installed on the xCAT server because many of the individual checks and framework use xCAT commands. In addition some of the health checks are MPI parallel applications which run using POE runtime environment.

Required software packages

- xCAT
- POE environment
- Python 2.6

1.2 Installation

The cluster health check framework along with some tools is delivered as an rpm, and is installed as user root with the rpm command:

```
# rpm -ihv cluster-healthcheck-1.1.0.0-0.noarch.rpm
```

The rpm command installs framework which consists few commands and health check tools to */opt/ibmchc*.

To run the health checks directly from command prompt, there are two possibilities.

Either edit *~/.profile* and add the following line:

```
# export PATH=$PATH:/opt/ibmchc/bin
```

Or create a symlink to */opt/ibmchc/bin/hcrun* with:

```
# ln -s /opt/ibmchc/bin/hcrun /usr/bin/hcrun
```

1.3 Configuration

The cluster heath check and verification tools are developed to check the health of individual cluster components or set of cluster components. The health check tools can be grouped into different groups of tools based on cluster components checked by the tools. There are three different types of configuration files (global, tool, group) to configure framework, individual tools and tool groups. These configuration files help to integrate new health check tools seamlessly with the framework. The general format of the configuration file is keyword value pairs defined under sections.

The Master or global configuration file `chc.conf` is created in the directory `/etc/opt/ibmchc/` The settings can be changed in Master configuration file */etc/opt/ibmchc/chc.conf* as shown in *Example 1-1*.

Example 1-1 */etc/opt/ibmchc/chc.conf*

```
[MasterConfig]

# The list of high level directories under
# which configuration of different health
# checking tools and groups are defined.
configpath =
/opt/ibmchc/tools:/opt/ibmchc/ppc64/tools:/opt/ibmchc/x86_64/tools:/opt/ibmchc/conf

# stdout results of health checking tools are saved in the
# file specified below.
toolsoutput = /var/opt/ibmchc/log/toolsoutput

# The summary of health checking tools
# are logged in the file defined below.
summaryfile = /var/opt/ibmchc/log/hcrun_sum.log

# The hcrun logs are stored in the file defined below.
hcrunlog = /var/opt/ibmchc/log/hcrun.log

# The user id which is used to run MPI jobs.
# Tools can access user name using environment variable $HC_USER.
# No default user is used.
hcuser =

# The comma separated UFM server host names or ip
# addresses. No default values.
# Tools can access subnet Managers using environment
# variable $HC_SUBNET_MANAGERS
subnet_managers =
# The comma separated IB Switch nodes or node groups.
# Tools can access IB Switch groups using environment variable $HC_IB_SWITCHES
# No default values.
ib_switches =
```

The defaults values defined are fine for most cases; however the following parameters need to be updated to match each unique cluster's environment:

- hcuser

  Some of the tests run with the PE environment. These tests must not be submitted as root, thus a dedicated non-root user must be configured. The value is exported through environment variable *HC_USER. The user must have authority to run applications in the PE environment.*

- subnet_managers

  If an InfiniBand subnet manager is available, its hostname can be specified in the configuration file . Multiple servers can also be specified as a comma separated list. The value is exported through environment variable *HC_SUBNET_MANAGERS. Tools that require access to subnet manager servers will use this environment variable.*

- ib_switches

  Specify the xCAT node groups for the InfiniBand switches as a comma separated list. Note that this requires the switches to be in the xCAT database and have ssh access enabled. Different groups for different model of switches should be created in the xCAT database. The value is exported through environment variable *HC_IB_SWITCHES.Tools that require access to InfiniBand switches will use this environment variable.*

1.3.1 Tool configuration files

Each of the available tools has its own configuration file. The default configuration file should work for most cases. You may wish to customize to your own needs, or if you want to add a tool to the health check framework, you will need to create a configuration file for it. The configuration keywords of the tools are shown in Table 1-1. Each tool configuration file needs to be somewhere in or below the path which is specified by the `configpath`parameter in the global configuration file. All tool configuration files must have the '*.conf'* extension. The following table shows all available parameters or keywords and their value types for such a tool configuration file.

Table 1-1 Health Check Tool specific configuration keywords

Parameter	**Required**	**Value type**	**Remark**
[HealthCheckTool]	Yes	n/a	Section header for the configuration file
name	Yes	String	Name of the Health check Tool
description	Yes	Single line Text	Short description of the tool
Type	Yes	String	Tool type to which this tool belongs. For display purposes only.
executable	No	Path to executable script or binary	Specifies the program to be run. **Default if not set**: Tool configuration path without '*.conf*' extension
Copy	No	yes | no	Specifies if the tool needs to be copied to the node or node group (uses **xdsh -e**) **Default if not set**: 'yes'
starttool	No	all | local	Specifies if the tool needs to be run on all nodes (provided in a nodelist to hcrun) or just on the node where hcrun is running. **Default if not set**: all
arguments	No	String or Strings separated by space	Default command line arguments for the executable. It is possible to specify non default arguments in the hcrun command line after the tool name argument. If any arguments have multiple words separated with a space then they need to be enclosed within double quotes.
environment	No	Environment variables with their values.	Environment variables exported for the tool (Declaration of multiple variables with ';'). Each environment variable is defined as variable name and value pairs separated by equal (=) sign. If **hcrun -n <NODERANGE>** is specified then the noderange is exported as *HC_NODERANGE.*
precheck	No	Tool name(s)	Optional comma separated list of tool names which will run before this tool

Parameter	Required	Value type	Remark
predatacollectiontool	No	Tool name(s)	Optional comma separated list of tool names which will run just before this tool to collect data. Optional command line arguments also can be specified which over-writes the default command line arguments defined for pre-data collection tool.
postdatacollectiontool	No	Tool name(s)	Optional comma separated list of tool names which will run after this tool to collect data. Optional command line arguments also can be specified which over-writes the default command line arguments defined for post-data collection tool.
processmethod	No	xcoll | compare | xcoll,compare	Optional comma separated list of supported tool output process methods. The process is shown in Error: Reference source not found1.4 Tool output processing methods" on page 10.
passexitcode	No	Numeric	Optional comma separated list of exit codes of the tool, which indicate the test as "passed". The exit code "0" is considered default pass exit code.
failexitcode	No	Numeric	Optional comma separated list of exit codes of the tool which indicate the test as "failed". The non-zero exit codes considered default fail exit provided no pass exit codes are defined. If either pass exit codes, warning exit code, or error exit codes are defined then exit codes which are not members of those exit codes are considered Fail exit codes.
warningexitcode	No	Numeric	Optional comma separated list of exit codes of the tool which indicate the test passed with "warning". No default exit codes.
errorexitcode	No	Numeric	Optional comma separated list of exit codes which indicate the test has hit "errors" to start. No default exit codes.

Example 1-2 shows configuration file for `fs_usage` tool. The executable is copied to all the specified nodes, runs on all of the nodes, and that only when it exits with an exit code zero (0) it is considered to be a successful check. The `fs_usage` check is part of the tool type node. The `fs_usage` tool supports both tool output process methods (xcoll and compare) as described in "1.4 Tool output processing methods" on page 10.

Example 1-2 /opt/ibmchc/tools/node/fs_usage.conf

```
[HealthCheckTool]
name=fs_usage
description=Checks the usage of the provided filesystems
type=node
executable=/opt/ibmchc/tools/node/fs_usage
arguments=/tmp /var/tmp
copy=yes
starttool=all
processmethod=xcoll,compare
passexitcode=0
errorexitcode=128
```

The configuration file makes it easy to implement or integrate new checks into the cluster health check framework. Even the scripting language can be chosen easily, for example Bash, Python or Perl script, as long as the interpreter is available on the target node. There are mainly two types of scripts or checks. One type is simple query scripts, which query values and output them. The other type, check scripts, which have logic implemented and also provide exit codes for pass, fail, or warning. The following are the simple steps followed for adding new script:

1. Create the health check script
2. Copy your script to the appropriate arch specific folder (*/opt/ibmchc/tools* or */opt/ibmchc/ppc64/tools* or */opt/imbchc/x86_64/tools*)
3. Create the configuration file for the script in the same folder with extension '*.conf*'
4. Make the script executable

1.3.2 Group configuration files

The group configuration file defines multiple tools which should run together in a specific order. There are some pre-defined groups which come with the rpm and users also can define their own groups depending on their needs. Predefined groups are located in */opt/ibmchc/conf/groups/chc.* These configuration files may not require editing. The User-defined groups are located in */opt/ibmchc/conf/groups/user.*

Example 1-3 shows a group configuration file example. All tools will be executed in the given order. The Section header is called *[HealthCheckToolsGroup]*. The name, and description are mandatory keywords and keyword environment is optional. All these keywords take Strings as their values. The keyword *tools* is a mandatory keyword whose value is a comma separated list of tool names in a specific order.

Example 1-3 /opt/ibmchc/conf/groups/chc/node_check.conf

```
[HealthCheckToolsGroup]
```

```
#The name of the group
name=node_check

#The short description of the tool group.
description=This group has node check tools

#The environment variables and their values. These environment variables
# are exported while running the member tools as part of the group.
Environment=

#The member tools of the group
tools=cpu, memory, gpfs_state, leds, temp, nfs_mounts, firmware, fs_usage, os
```

The environment variables defined in the group configuration file are exported for every tool of the group before starting the tool. In the case of a conflict, these environment variables over-write the tools environment variablest. To overwrite tool keywords of the tool configuration files (like arguments or environment), add the following keyword to the group configuration file:

<toolname>.<keywordname>=<new value>

as shown in Example 1-4.

Example 1-4 same as Example 1-3 above but with overwritten fs_usage arguments

```
[HealthCheckToolsGroup]
#The name of the group
name=node_check

#The short description of the tool group.
description=This group has node check tools

#The environment variables and their values. These environment variables
# are exported while running the member tools as part of the group.
Environment=

#The member tools of the group
tools=cpu, memory, gpfs_state, leds, temp, nfs_mounts, firmware, fs_usage, os

#The fs_usage arguments is redefined in the group.
fs_usage.arguments=/tmp
```

The tool keyword values redefined in the group for a tool is used while running that tool under that group. Otherwise the default values specified for that keyword in tool configuration file is used. The group environment variables and redefined keyword values of tools can be used to change the behavior of the tool while running as part of that group.

Creating a group configuration file and integrating with the framework is simple one step process. That is;

1. Create the configuration file with extension '*.conf*' in */opt/ibmchc/conf/groups/user* using your favorite text editor.

The new group automatically gets integrated with framework.

1.4 Tool output processing methods

The cluster health check provides different methods to process the output of a tool.
The current version provides the following modes:

- plain (no post processing)
- xcoll
- compare

Some tools can't support compare and xcoll because the output of those tools may not be in the format suitable to use xcoll and compare methods. Hence, the tools must define the methods they support as shown in "1.3.1 Tool configuration files" on page 5. The processing method plain is the default method. The processing methods can be used for health check groups as well. If any member tool of a group does not support specified methods then that tool is run in default (plain) mode.

1. **Plain**
   The plain mode is the default mode for all the tools. In plain mode tools will be executed and the output or results are shown without any post processing. This is useful to check values of individual nodes.

2. **xcoll**
   In xcoll mode the tool will be executed and the output or results are piped (forwarded) to the xCAT command xcoll The xcoll command summarizes the output of different nodes and nodegroups, whose output matchies, to one "output group". More details about the xCAT command xcoll can be found at

   http://xcat.sourceforge.net/man1/xcoll.1.html.

   This method is generally used for the health check tools which query the attributes of the cluster components without checking results. This method is good for checking consistency within a nodegroup.

3. **compare**
   The compare process method uses */opt/ibmchc/tools/config/config_check* utility tool to compare the output of the tools with a given baseline, or expected output. The config_check is a front end tool to the xCAT **sinv** command which collects the output and compares the results with given templates. For more information on **sinv**, refer to:

   http://xcat.sourceforge.net/man1/sinv.1.html.

   The baseline can be created by the cluster health check tool by initially running health check tool against a given seed node. This will create a baseline template with the output of seed node. Typically, the user will verify the template when it is first created. On further invocations of the health check, the results of the different nodes are checked against the baseline. The baseline template can be regenerated or replaced by running health check tool using different or same seed node again. Template regeneration is typically done after maintenance is applied that will change the tool's results.

Note: It is extremely important that the seed node has the expected results so that subsequent invocations only reveal mismatches to a known good result.

2.0 Framework

The Cluster Health Check framework consists of mainly the **hcrun** command and the utility command **config_check**. In this chapter the framework commands `hcrun` and `config_check` are described.

2.1 The hcrun command

NAME

`hcrun` - Runs individual health checks or group of health checks and processes their results.

PURPOSE

Runs individual health checks or group of health checks on range of nodes or node groups. The results of the health checks are processed using different process methods.

SYNTAX

```
hcrun [-h] [-v] [-c] { -l | [[-f ] -n noderange ] [[–s seed_node]  –m
process_method ]{ [-p] group[,group,...] |-t tool [,tool,... |
cmd_args ]}}
```

DESCRIPTION

There are various health checking and verification tools available for checking the cluster health during cluster bring-up, maintenance and problem debug time. These tools are developed to check the health of individual cluster components or group of cluster components. At any time, these individual tools or a group of tools can be executed to check or verify the health of the cluster components. The health check tools are grouped into different groups based on health check functionality or components tested by them. There are separate individual configuration files for each tool customizing the tool to support different cluster environments.

The hcrun command is used to run cluster health checking tools interactively. The xCAT command xdsh, is used to run health checking tools on the specified set (node range) of nodes. If nodes are not specified then the health checking tools are executed on the local node (Management Node) based on configuration of the tool. If tool is configured to run on multiple nodes but did not provide the nodes to run then it would be in error. The hcrun command accepts either a single health check tool with its command line arguments or a list of health check tools separated by commas. If a list of health check tools are specified then the command line arguments specified in configuration files of individual tools are used while executing respective tools and the tools also get executed in the order they are specified.

The hcrun also accepts a list of health checking tool group names separated by commas. If group names are specified all the tools of the individual groups are executed in an order specified in the group configuration file until either the first health check tool fails or all the tools are executed successfully. Even though the tools of a group are executed in the order the tools are specified in the group configuration file, the pre-check tools of individual tools are always executed before the individual tool is executed. If –c (continue) option is specified then all the health check tools of the groups are executed even if one or more health check tools fail. By default, the execution of health check tools of a group are stopped once a tool fails. If multiple groups are specified then the tools of the groups are executed in the order the groups are specified on the hcrun command line.

The cluster health check tools of the groups are executed on the same nodes specified on hcrun command line. The tools are executed using xdsh command. If any tool of a group should not be started using the xdsh command then the configuration keyword "StartTool" of that tool should be set to "local", in which case that particular tool is started on local node (Management Node) only. For any reason, if a particular tool wants to know the set of nodes (node range), it can access the environment variable **HC_NODERANGE** which is always set by hcrun command to the node range specified to hcrun command.

The hcrun command saves the cluster health checking tools results in the file specified by keyword "toolsoutput" in the master configuration file. These results are used for historical analysis. The hcrun command also saves the summary of the cluster health checking tools in a summary file specified by the keyword "summaryfile". The summary file has brief details of the cluster health checking tools like start and end times of the tools along with final status (Fail/Pass) of the tools. In addition to storing the tools results in the file specified by the keyword "toolsoutput", it also displays the health checking tools results to the stdout.

The output of the tools can also be processed using different utility tools like the xcoll command. By default the health checking tools are run in plain mode where the results of the tools are displayed. The hcrun command also supports two other methods “xcoll” and “compare”. When xcoll is used the tools output is piped through the xcoll command of xCAT before display, where the xcoll analyzes the tools output and segregate the nodes which have common output. This would help to identify the set of nodes on which the health check is passed and on which health check is failed, or to reveal unexpected inconsistencies. When “compare” method is used the output of health check tools from each node is compared against a template (baseline output). If output from one or more nodes does not match with the baseline template, alternative templates along with the nodes matching the alternative templates are displayed. This method can be used to find out the nodes which are deviating from the known baseline. The health check is considered to have passed if the tool output from all the given nodes are matched with the template, else the health check is considered to have failed.

OPTIONS

-h The small usage information about the hcrun command is displayed.

-v The hcrun command and cluster health checking tools are executed in verbose mode. The environment variable HC_VERBOSE is set to 1 and exported to health checking tools if hcrun is run with option (-v). By default the HC_VERBOSE is set to 0. To support verbose mode each cluster health checking tool should verify the HC_VERBOSE environment variable. If HC_VERBOSE=1, then the command should run in verbose mode.

-c The execution of health checking tools of a group are to continue running, even if one or more cluster health checking tools of the group fails. This does not apply to any “pre-checks”, which are considered to be required to pass.

-l The command lists all the configured cluster health checking tools according to their types. The defined tool groups also listed along with their member tools and other attributes. If –l option along with one or more tool names are specified then all attributes defined for the tools are listed. If –l option along with one or more groups are specified then the tools of the specified groups and other attributes of the groups are listed.

-n <NODERANGE>
The cluster health checking tool specified or the cluster health checking tools of the specified group are executed on the nodes specified by NODERANGE. The syntax of the node range is the same as supported by xCAT. The detailed explanation of NODERANGE syntax can be found at xCAT document available at http://xCAT.sourceforge.net/man3/noderange.3.html . An

environment variable HC_NODERANGE is set with node range specified and exported to health checking tools.

-fGenerally (by default) the execution of the health checking is stopped if one or more nodes in the node range specified are not reachable. If –f (force) option is specified then the execution of health checking is continued even if one or more nodes are not reachable.

group[,group,…]

One or more group names separated by comma, whose cluster health checking tools are executed. The tools of the groups are executed in the order the groups specified. The tools within a group are also executed in the order the tools defined in that group.

-*p* If groups are specified on command line, the tools of the individual groups are executed in the order the tools defined in respective groups. If –p (preview) option along with groups is specified then the health checking tools are listed in the order they would get executed but the tools actually would not be executed.

-t tool[,tool,…]

The tools specified are executed. If either only tool name is specified without any command line arguments required for that tool or list of tools are specified then the default command line arguments configured in configuration files of the tools are used while executing the respective tools. If tool name along with command line arguments are specified then the tool is executed using the command line arguments specified by overriding default command line arguments configured in configuration file.

<cmd_args>

The command line arguments passed to the tool when single health check tool name is specified.

-m <process_method>

The methods used to post process the output of the tools. The supported methods are “xcoll” and “compare”. When “xcoll” method is used the tools output is piped through the xcoll command of xCAT before display, where the xcoll command analyzes the tools output and segregates the nodes which have common output. If “compare” method is used the output of health check tools form each node is compared against a template (baseline output). If output from one or more nodes does not match with the baseline template, alternative templates along with the nodes matching the alternative templates are displayed. The hcrun uses the utility command config_check (See section **2.2 The config_check command**) to compare the tools results with baseline.

-s <seed_node>

A seed node along with processing method "compare" is used, first the baseline template is created by running tool on the seed node. Then the output of health check tools form each node is compared against template (baseline output) generated. If output from one or more nodes does not match with baseline template, alternative templates along with the nodes matching the alternative templates are displayed. The hcrun uses the utility command config_check (See section **2.2 The config_check command**) to compare the tools results with baseline.

EXAMPLES

1. To list all the health check tools available run the hcrun command without passing any arguments as shown below.

```
e119f3ems2:~ # hcrun
config_check            : Util tool used to compare the query attributes of nodes against a seed node
switch_module           : Query the switch modules installed
run_ppping              : Runs ppping interconnect test
ibdiagcounters_clear    : Util tool to clean IB diag counters
fs_usage                : Checks the usage of the provided filesystems
run_nsdperf             : Runs nsdperf interconnect test
switch_health           : Checks the switch health report
syslogs_clear           : Util tool to clean the syslogs
firmware                : Checks the BMC and UEFI firmware of the node
run_jlink               : Runs jlink interconnect test
hca_basic               : Checks basic IB fabric parameters
file_rm                 : Util tool to remove files copied to the execution node
nfs_mounts              : Checks nfs mounts
memory                  : Checks the Total memory on the node
jlinksuspects           : Correlates sub-par jlink BW with BER calculations per Subnet Manager(s)
jlinkpairs              : Gets all jlink pairs whose BW is below threshold
hostfile_copy           : Util tool to copy hostfile to the execution node
lastclear               : Returns timestamp from last clearerrors on the Subnet Manager(s)
clearerrors             : Clears the IB error counters on the Subnet Manager(s)
switch_inv              : Query the switch inventory
switch_clk              : Checks the switch clock consistency
run_daxpy               : Runs daxpy test on node(s)
run_dgemm               : Runs degemm test on the node(s)
leds                    : Checks the leds of the node
ibtools_install         : Installs IB Tools on the Subnet Manager(s)
temp                    : Checks all temperatures
gpfs_state              : Checks the gpfs state
ibtoolslog_clean        : Archives the ibtools logs on the Managment Node and Subnet Manager(s)
switch_code             : Checks to see if the switch code is consistent
ipoib                   : Query IPoIB settings
hostfile_rm             : Util tool to remove hostfile to the execution node
ibqber                  : Runs an IB query and BER calculation on the Subnet Manager(s)
switch_ntp              : Checks to see if the switch ntp is enabled
os                      : Checks the OS and kernel version
cpu                     : Checks for the CPU status
berlinks                : Gets the links from a BER calculation on the Subnet Manager(s)
file_copy               : Util tool to copy required files to the remote node
e119f3ems2:~ #
```

DESCRIPTION

Each tool name, followed by a colon, and followed by a short description of the tools is displayed.

2. To list the tools as per their type and the tool groups in which they are members, run the following command.

```
e119f3ems2:~ # hcrun -l
The Tool Type    : The Tools              : Tool Description
=================================================================
node           : firmware               : Checks the BMC and UEFI firmware of the node
               temp                  : Checks all temperatures
               run_daxpy              : Runs daxpy test on node(s)
               nfs_mounts             : Checks nfs mounts
               fs_usage               : Checks the usage of the provided filesystems
               leds                  : Checks the leds of the node
               os                    : Checks the OS and kernel version
               run_dgemm              : Runs degemm test on the node(s)
               gpfs_state             : Checks the gpfs state
               memory                 : Checks the Total memory on the node
               cpu                   : Checks for the CPU status

tools_util      : config_check           : Util tool used to compare the query attributes of nodes against a seed
node
               file_rm                : Util tool to remove files copied to the execution node
               syslogs_clear           : Util tool to clean the syslogs
               file_copy              : Util tool to copy required files to the remote node
               hostfile_copy           : Util tool to copy hostfile to the execution node
               hostfile_rm            : Util tool to remove hostfile to the execution node
               ibdiagcounters_clear    : Util tool to clean IB diag counters

ib             : switch_ntp             : Checks to see if the switch ntp is enabled
               ipoib                 : Query IPoIB settings
               switch_health           : Checks the switch health report
               switch_module           : Query the switch modules installed
               hca_basic              : Checks basic IB fabric parameters
               switch_inv             : Query the switch inventory
               switch_code             : Checks to see if the switch code is consistent
               switch_clk             : Checks the switch clock consistency

interconnect     : run_ppping              : Runs ppping interconnect test
               run_jlink              : Runs jlink interconnect test
               run_nsdperf             : Runs nsdperf interconnect test

tools_ib        : jlinkpairs             : Gets all jlink pairs whose BW is below threshold
               ibtools_install         : Installs IB Tools on the Subnet Manager(s)
               clearerrors            : Clears the IB error counters on the Subnet Manager(s)
               lastclear              : Returns timestamp from last clearerrors on the Subnet Manager(s)
               ibtoolslog_clean        : Archives the ibtools logs on the Managment Node and Subnet Manager(s
)
               jlinksuspects           : Correlates sub-par jlink BW with BER calculations per Subnet Manager(s)
               berlinks               : Gets the links from a BER calculation on the Subnet Manager(s)
               ibqber                 : Runs an IB query and BER calculation on the Subnet Manager(s)

=================================================================

The Group Name  : Group Attributes                 : Attributes Values
=================================================================
==========
node_test       : Description                      : These are intrusive node health testing tools.
              Environment                      :
              Tools                          : ['run_dgemm', 'run_daxpy']

node_check      : Description                      : This group has node check tools
              Environment                      :
              Tools                          : ['cpu', 'memory', 'gpfs_state', 'leds', 'temp', 'nfs_mounts', 'firmware',
'fs_usage', 'os']

ib_check        : Description                      : This group has ib check tools
              Environment                      :
              Tools                          : ['ipoib', 'hca_basic', 'switch_code', 'switch_module', 'switch_inv',
'switch_clk', 'switch_health', 'switch_ntp']
```

```
interconnect_test: Description                    : These are intrusive interconnect health  testing tools
              Environment                  :
              Tools                        : ['run_ppping', 'run_jlink', 'run_nsdperf']

run_ibtools     : Description                     : Runs jlink with a tools_ib pipeline
              Environment                  :
              Tools                        : ['clearerrors', 'lastclear', 'run_jlink', 'ibqber', 'berlinks', 'jlinkpairs',
'jlinksuspects', 'ibtoolslog_clean']

e119f3ems2:~ #
```

DESCRIPTION

First the tools are listed as per their type. The configured types are node, tools_util, ib, interconnect and tools_ib. Then the configured groups node_test, node_check, ib_check, interconnect_test, and run_ibtools are displayed along with their member tools.

3. The attributes of a tool can be displayed using hcrun command as shown below.

```
e119f3ems2:~ # hcrun -l -t cpu
Tool Name    : Tool Attributes          : Attributes Values
=====================================================
===========
cpu          : Executable               : /opt/ibmchc/x86_64/tools/node/cpu
             Arguments                 :
             Environment               :
             Description               : Checks for the CPU status
             Precheck Test             : []
             Predata Collection Test  : []
             Post Data Collection Test: []
             Data Processing Method   : ['xcoll', 'compare']
             Copy                      : yes
             Start Tool                : all
             Groups                    : ['node_check']
             Type                      :   node
             Pass Exit Codes           : ['0']
             Fail  Exit  Codes         :  []
             Warning Exit Codes        : []
             Error Exit Codes          : ['128']

e119f3ems2:~ #
```

DESCRIPTION

The tool “cpu” along with its various attributes is displayed.

4. The attributes of configured tool group can be listed as shown below using hcrun command.

```
e119f3ems2:~ # hcrun -l node_test
The Group Name  : Group Attributes            : Attributes Values
=====================================================
==========
node_test       : Description                 : These are intrusive node health testing tools.
```

```
            Environment              :
            Tools                    : ['run_dgemm', 'run_daxpy']

e119f3ems2:~  #
```

DESCRIPTION

The tool group "`node_test`" along with its various attributes is displayed.

5. The health check tool "cpu", which query different attributes of "`cpu`" and display is run using `hcrun` command as shown below.

```
e119f3ems2:~ # hcrun -n e119f4m1n0[4-5] -v -t cpu

e119f4m1n04: CPU Model:              Intel(R) Xeon(R) CPU           5160  @ 3.00GHz
e119f4m1n04: Turbo HW:         Off
e119f4m1n04: Turbo Engaged:           No
e119f4m1n04: HyperThreading HW:        Disable
e119f4m1n04: Socket(s):               Core(s) per socket:     2
e119f4m1n04: Active Cores:            4
e119f4m1n04: scaling_governor:  ondemand
e119f4m1n04: scaling_max_freq:  2997000
e119f4m1n04: scaling_min_freq:  1998000
e119f4m1n05: CPU Model:              Intel(R) Xeon(R) CPU           5160  @ 3.00GHz
e119f4m1n05: Turbo HW:         Off
e119f4m1n05: Turbo Engaged:           No
e119f4m1n05: HyperThreading HW:        Disable
e119f4m1n05: Socket(s):               Core(s) per socket:     2
e119f4m1n05: Active Cores:            4
e119f4m1n05: scaling_governor:  ondemand
e119f4m1n05: scaling_max_freq:  2997000
e119f4m1n05: scaling_min_freq:  1998000
The health check tool  cpu                              [ PASS ]
e119f3ems2:~ #
```

DESCRIPTION

As shown from health check tool "cpu" output, various attributes from nodes `e119f4m1n04`, and `e119f4m1n05` are displayed.

6. To process the health check tools output using method "xcoll" run the hcrun command as shown below.

```
e119f3ems2:~ # hcrun -n e119f4m1n0[3-7] -m xcoll  -t cpu
====================================
e119f4m1n06,e119f4m1n03,e119f4m1n05,e119f4m1n04,e119f4m1n07
====================================
CPU Model:            Intel(R) Xeon(R) CPU           5160  @ 3.00GHz
Turbo HW:             Off
Turbo Engaged:          No
HyperThreading HW:      Disable
Socket(s):            Core(s) per socket:     2
Active Cores:         4
scaling_governor:       ondemand
scaling_max_freq:       2997000
scaling_min_freq:       1998000

The health check tool  cpu                              [ PASS ]
```

```
e119f3ems2:~ #
```

DESCRIPTION

The results from health check tool "`cpu`" is processed by xcoll method and summarized. Here all the nodes, `e119f4m1n06, e119f4m1n03, e119f4m1n05, e119f4m1n04, and e119f4m1n07` have same results returned from "`cpu`" health check tool.

7. To over-write the default arguments defined for a health check tool run the hcrun command by specifying the command line arguments to the tool as shown below.

```
e119f3ems2:~ # hcrun -n e119f4m1n0[3-7] -c  -m xcoll  -t  fs_usage
====================================
e119f4m1n06,e119f4m1n03,e119f4m1n05,e119f4m1n04,e119f4m1n07
====================================
/tmp: 7%
/var/tmp: 7%

The health check tool  fs_usage                    [ PASS ]
e119f3ems2:~ # hcrun -n e119f4m1n0[3-7] -c  -m xcoll  -t  fs_usage /tmp
====================================
e119f4m1n06,e119f4m1n03,e119f4m1n05,e119f4m1n04,e119f4m1n07
====================================
/tmp: 7%

The health check tool  fs_usage                    [ PASS ]
e119f3ems2:~ #
```

DESCRIPTION

As shown above the second execution of health check tool "`fs_usage`" uses the argument (/tmp) passed on the command line. The first execution uses the default arguments defined in the configuration file.

8. To process the health check tool group results using method "`xcoll`" run the hcrun command as shown below.

```
[root@c933mnx01 node]# hcrun -n c933f01x31,c933f01x33 -c -m xcoll node_check
====================================
c933f01x33
====================================
CPU Model:           Intel(R) Xeon(R) CPU         X5560  @ 2.80GHz
Turbo HW:            Off
Turbo Engaged:         No
HyperThreading HW:      Enable
Socket(s):           Core(s) per socket:     4
Active Cores:         16
scaling_governor:      performance
scaling_max_freq:      2793000
scaling_min_freq:      1596000

====================================
c933f01x31
====================================
CPU Model:           Intel(R) Xeon(R) CPU         X5560  @ 2.80GHz
Turbo HW:            Off
Turbo Engaged:         No
HyperThreading HW:      Enable
```

```
Socket(s):            Core(s) per socket:     4
Active Cores:          16
scaling_governor:       ondemand
scaling_max_freq:       2793000
scaling_min_freq:       1596000

The health check tool  cpu                          [ PASS ]
=======================================
c933f01x33,c933f01x31
=======================================
Memory 32097 MB

"2048 MB","DIMM01","BANK01","800 MHz"
"2048 MB","DIMM02","BANK02","800 MHz"
"2048 MB","DIMM03","BANK03","800 MHz"
"2048 MB","DIMM04","BANK04","800 MHz"
"2048 MB","DIMM05","BANK05","800 MHz"
"2048 MB","DIMM06","BANK06","800 MHz"
"2048 MB","DIMM07","BANK07","800 MHz"
"2048 MB","DIMM08","BANK08","800 MHz"
"2048 MB","DIMM09","BANK09","800 MHz"
"2048 MB","DIMM10","BANK10","800 MHz"
"2048 MB","DIMM11","BANK11","800 MHz"
"2048 MB","DIMM12","BANK12","800 MHz"
"2048 MB","DIMM13","BANK13","800 MHz"
"2048 MB","DIMM14","BANK14","800 MHz"
"2048 MB","DIMM15","BANK15","800 MHz"
"2048 MB","DIMM16","BANK16","800 MHz"

The health check tool  memory                          [ PASS ]
=======================================
c933f01x33,c933f01x31
=======================================
fsB : OK
verbsRDMA : OK

The health check tool  gpfs_state                     [ PASS ]

c933f01x33: LED 0x0000 (Fault) active to indicate system error condition.
c933f01x33: LED 0012 (PS) active to indicate Sensor 0x71 (Power Supply 2) error.
rvitals  FAIL!
The health check tool  leds                           [ FAILED ]
=======================================
c933f01x33
=======================================
Ambient Temp: 31 C (88 F)

=======================================
c933f01x31
=======================================
Ambient Temp: 30 C (86 F)

The health check tool  temp                          [ PASS ]
=======================================
c933f01x33,c933f01x31
=======================================
ugpfs01-ug:/gpfs/cnfs01/data/u
ugpfs01-ug:/gpfs/cnfs01/data/admin

The health check tool  nfs_mounts                     [ PASS ]
=======================================
c933f01x33,c933f01x31
=======================================
BMC Firmware: 1.35 (YUOOE9B 2012/11/29 09:14:45)
UEFI Version: 1.16 (D6E158A 2012/11/26)

The health check tool  firmware                       [ PASS ]
=======================================
c933f01x33,c933f01x31
```

```
========================================
/tmp: 2%
/var/tmp: 2%

The health check tool  fs_usage                         [ PASS ]
========================================
c933f01x33,c933f01x31
========================================
Description:    Red Hat Enterprise Linux Server release 6.3 (Santiago)
Release:        6.3
Codename:       Santiago
Kernel:         2.6.32-279.el6.x86_64

The health check tool  os                               [ PASS ]
[root@c933mnx01 node]#
```

2.2 The config_check command

NAME

config_check - Runs health checks and compares the output against a baseline

PURPOSE

Runs the commands from the command file on a single node or range of nodes, and node groups. The output from each node is compared with a baseline template or output collected from a seed node. Differences are highlighted. It is intended as a utility or wrapper for health check commands/scripts.

SYNTAX

```
config_check [-l] | [-h] | [<nodegroup>

  [-i <seed node>] -n <name> [-L] -f <command-file>

  [-t max number of templates] [-k] [-y] [-v]

  ]
```

DESCRIPTION

The config_check utility command is used to run commands from a given command file or script on the specified node(s) or node groups. The output generated from commands of command file is compared using the xCAT command sinv against the template, whose name is specified. While comparing the results with the template, if the output from one or more nodes does not match the template, then alternate templates are generated and contains those varying results. The alternative templates generated can be saved in the directory specified by environment variable HC_TEMPLATESDIR, which can be used for examination or analysis. The alternative templates are named by appending a sequence number, starting from 1, to the base template name. By default, the alternate templates are removed as soon as the test is completed unless requested to keep them. The output log describes the differences from the baseline template.

The config_check command can also be used to generate new templates by running the commands from command file on the seed node specified and to compare the output from all the nodes or node groups specified with the new template created. The config_check command uses the directory specified by environment variable HC_TEMPLATESDIR to look for the template, and to save the new template under the name that is specified.

The commands specified in the command file are run on nodes or node groups using the xCAT command `xdsh`. If the commands specified in command file are remote execution commands like `rinv, rvitals` then the commands of the command file have to be run locally on Management node. By default, the `Config_check` command display pass/fail health check by comparing the health check output with template. It can be run in verbose mode as well to display the details of the heath check result. The details or logs of the health checks also saved in a log file in directory specified by environment variable HC_LOGDIR. The `config_check` utility tool can be run by exporting environment variables HC_TEMPLATESDIR, HC_LOGDIR, and by specifying template name.

OPTIONS

<nodegroup>

The range of nodes or node groups on which the given commands of command file are executed. The syntax of the node group is same as supported by `xCAT`.

-l The command lists the templates available. It list the templates available in HC_TEMPLATESDIR directory.

-n <name>

The name short name (without extension `template.config_check`) of the template used to compare the results with. If used with -i option it is name of template to be generated.

-f <command-file>

The commands file name, whose commands are run on the given node range or node group.

-L The given commands from command file are run on local management node.

-i <seed_node>

The seed node on which the given commands from command file are executed to generate the templete. The generated template is stored in HC_TEMPLATESDIR directory with the given name. It also executes the specified commands from command file on nodes or node group and compared against generated template using **xCAT** command `sinv`.

-t <max_number_of_templates>

The maximum number of alternative templates allowed being generated if results are not matched with given template. The default value is 10.

-k Keeps alternative template generated. These can be helpful for examining the results later.

-y Do not copy the given commands from command file to node group. The commands of command file already there on the nodes or node groups.

-v Run the command in verbose mode. By default the command just displays pass or fail depending on whether results matched with template or not. In verbose more it lists which nodes match which template including alternative templates.

-h Display the command usage statement to standard output.

EXAMPLES

1. To run the `cpu` health check tool using `config_check` explicitly by exporting environment variables HC_TEMPLATESDIR , HC_LOGDIR, run the following commands.

```
e119f3ems2:~ # export  HC_TEMPLATESDIR=/opt/ibmchc/conf/templates
e119f3ems2:~ # export HC_LOGDIR=/var/opt/ibmchc/log
e119f3ems2:~ # /opt/ibmchc/tools/config/config_check e119f4m1n0[3-7] -n cpu -f
/opt/ibmchc/x86_64/tools/node/cpu -t 10 -v
The following nodes match /opt/ibmchc/conf/templates/cpu.template.config_check:
e119f4m1n06,e119f4m1n03,e119f4m1n05,e119f4m1n04,e119f4m1n07

Info:Results can be found in file:
/var/opt/ibmchc/log/cpu.template/config_check.log.20131217_143054
e119f3ems2:~ #
```

The above results indicate that all nodes run against match the template. A mmismatch would have idisplayed a list of nodes with an alternate template name.

2.3 Using `config_check` and `hcrun` to run health checks

The `config_check` utility tool is developed to run under `hcrun` command, which exports the HC_TEMPLATESDIR and HC_LOGDIR specified in the i master configuration file of `hcrun`. The template name is automatically generated from the health check tool name and health check group name, when run under `hcrun` command. The template name is the same as tool name extension with `template.config_check`, if run without any group. If a tool is run as part of a group then template name is tool_name, followed by dot (.), followed by group name extension with `template.config_check`.

The following steps have to be followed to run health check using config_check under hcrun command.

- Create a script that has a consistent output on a given set of nodes. Sometimes this can be a simple wrapper for a command, and may involve using grep -v to remove unique parameters, such as IP addressing.
- Create the hcrun configuration file for the health check tool and define the `processmethod` of the health check tool as "`compare`" as shown in the Example 3-1 below:

Example 3-1 /opt/ibmchc/tools/node/fs_usage.conf

```
[HealthCheckTool]
name=fs_usage
description=Checks the usage of the provided filesystems
type=node
executable=/opt/ibmchc/tools/node/fs_usage
arguments=/tmp /var/tmp
copy=yes
starttool=all
processmethod=xcoll,compare
passexitcode=0
errorexitcode=128
```

- The first time you run `config_check` against a node group via `hcrun`, a seed node has to be specified to create a template based on the results from the seed node. On the first run, you should use verbose mode in hcrun (-v). You should check the template that is output (the name will be indicated in the STDOUT), and be sure that the results are as expected. If you don't do this, you will at least know if all of the nodes in the node group get consistent results relative to each other.
- In subsequent runs, you may use verbose or not. If there is a failure, you can find the latest output file and templates from the output.
- If all nodes mismatch the template, and you agree with the new values, you will want to re-run with the seed node, so that a new baseline template is generated. This will be used for comparison on all new runs.

EXAMPLES

1. To run the health check tool "cpu" using `config_check` under hcrun command, run the following command:

Note: The hcrun calls the config_check utility when method "compare" is specified.

```
e119f3ems2:~ # /opt/ibmchc/bin/hcrun -n e119f4m1n0[3-7] -m compare -t cpu
The health check tool cpu                              [ PASS ]
e119f3ems2:~ #
```

2. To run the health check tool “cpu” using `config_check` under hcrun command in verbose mode, run the following command:

   Note: The hcrun calls the config_check utility when method “compare” is specified.

```
e119f3ems2:~ # /opt/ibmchc/bin/hcrun -n e119f4m1n0[3-7] -m compare -v -t cpu
   The following nodes match /opt/ibmchc/conf/templates/cpu.template.config_check:
   e119f4m1n06,e119f4m1n03,e119f4m1n05,e119f4m1n04,e119f4m1n07

   Info:Results can be found in file:
   /var/opt/ibmchc/log/cpu.template/config_check.log.20131217_144504
   The health check tool  cpu                          [ PASS ]
   e119f3ems2:~ #
```

DESCRIPTION

The config_check tool will indicate whether nodes match or do not match the baseline template, and the output file that contains the details.

This output line indicates that a set of nodes match the baseline template:

```
The following nodes match /opt/ibmchc/conf/templates/cpu.template.config_check:
e119f4m1n06,e119f4m1n03,e119f4m1n05,e119f4m1n04,e119f4m1n07
```

This output line indicates where the detailed output file is to be found:

```
Info:Results can be found in file:
/var/opt/ibmchc/log/cpu.template/config_check.log.20131217_143054
```

When run under hcrun, the above details are only output in verbose mode. In either case, an indication of pass/fail is output:

```
The health check tool  cpu                          [ PASS ]
e119f3ems2:~ #
```

3. To run the health check tool “temp” using `config_check` under hcrun command in verbose mode, run the following command:

```
e119f3ems2:/opt/ibmchc/bin #./hcrun -n e119f4m1n1[3-7] -m compare -v –t temp
 Template path:/opt/ibmchc/conf/templates/temp.template.config_check.
 Template cnt:10.
 --
 The following nodes match /opt/ibmchc/conf/templates/temp.template.config_check:
 e119f4m1n14,e119f4m1n17,e119f4m1n16
 The following nodes match /opt/ibmchc/conf/templates/temp.template.config_check_1:
 e119f3ems2
 The following nodes match /opt/ibmchc/conf/templates/temp.template.config_check_2:
 e119f4m1n15,e119f4m1n13
```

When running a script on the local node to check remote nodes, there will be a template mismatch with the local node (e119f3ems2). It is to be ignored.

```
Info:Details can be found in file:
/var/opt/ibmchc/log/temp.template/config_check.log.20131220_080425

The health check tool  temp                         [ FAILED ]
e119f3ems2:/opt/ibmchc/bin #
```

DESCRIPTION

In this example, the config_check tool will indicate that nodes do not match the baseline template, and the output file that contains the details.

This output line indicates that a set of nodes match the baseline template:

```
The following nodes match
/opt/ibmchc/conf/templates/temp.template.config_check:
e119f4m1n14,e119f4m1n17,e119f4m1n16
```

This output line indicates the local node matches an alternate template:

```
The following nodes match
/opt/ibmchc/conf/templates/temp.template.config_check_1:
e119f3ems2
```

This output line indicates some nodes match a different alternate template:

```
The following nodes match
/opt/ibmchc/conf/templates/temp.template.config_check_2:
e119f4m1n15,e119f4m1n13
```

Because of a quirk with how sinv works, this output indicates that when a tool runs on local node the local node mismatch should be ignored:

When running a script on the local node to check remote nodes, there will be a template mismatch with the local node (e119f3ems2). It is to be ignored.

This output line indicates where the detailed output file is to be found:

```
Info:Details can be found in file:
/var/opt/ibmchc/log/temp.template/config_check.log.20131220_080425
```

When run under hcrun, the above details are only output in verbose mode. In either case, an indication of pass/fail is output:

```
The health check tool  temp                         [ FAILED ]
e119f3ems2:/opt/ibmchc/bin #
```

4. To run the health check tool group "`node_check`" using `config_check` under `hcrun` command, run the following command:

```
e119f3ems2:~ # hcrun -n e119f4m1n0[3-7] -c  -m compare node_check
The health check tool  cpu                          [ PASS ]
The following nodes match /opt/ibmchc/conf/templates/memory.node_check.template.config_check:
e119f4m1n03,e119f4m1n05,e119f4m1n04,e119f4m1n07
The following nodes match /opt/ibmchc/conf/templates/memory.node_check.template.config_check_1:
e119f4m1n06


Info:Details can be found in file:
/var/opt/ibmchc/log/memory.node_check.template/config_check.log.20131222_072841

The health check tool  memory                    [ FAILED ]
The health check tool  gpfs_state                [ PASS ]

e119f4m1n06: LED 0013 (LED 0x13) active to indicate Sensor 0x60 (Drive 1 Status) error.
e119f4m1n06: LED 0x0000 (LED 0x0) active to indicate system error condition.
rvitals  FAIL!
The health check tool  leds                      [ FAILED ]
Template path:/opt/ibmchc/conf/templates/temp.node_check.template.config_check.
Template cnt:10.
--
The following nodes match /opt/ibmchc/conf/templates/temp.node_check.template.config_check:
e119f4m1n07
The following nodes match /opt/ibmchc/conf/templates/temp.node_check.template.config_check_1:
e119f3ems2
The following nodes match /opt/ibmchc/conf/templates/temp.node_check.template.config_check_2:
e119f4m1n06,e119f4m1n03
The following nodes match /opt/ibmchc/conf/templates/temp.node_check.template.config_check_3:
e119f4m1n05,e119f4m1n04

When running a script on the local node to check remote nodes, there will be a
template mismatch with the local node (e119f3ems2). It is to be ignored.


Info:Details can be found in file:
/var/opt/ibmchc/log/temp.node_check.template/config_check.log.20131222_072854

The health check tool  temp                      [ FAILED ]
The health check tool  nfs_mounts                [ PASS ]
The health check tool  firmware                  [ PASS ]
The health check tool  fs_usage                  [ PASS ]
The health check tool  os                        [ PASS ]
e119f3ems2:~ #
```

DESCRIPTION

The health check tools, `memory`, `leds` and `temp` have failed and all other tools of the group "`node_check`" have passed. For the failed checks `memory` and `temp` the alternative templates are created for the nodes which deviate from the baseline template. The health check tool `leds` does not support processing method "`compare`", hence it is run in `plain` mode.

3.0 Guidelines for adding new tools

There are few simple guide lines that need to be followed while developing the Cluster Health Checking Tools which will be run using hcrun command. They are:

1. Each cluster health checking tool should have its own configuration file specifying different attributes of the tool.
2. In general, the tools should exit with zero on successful and non-zero on failure. If not, then the exit codes of the tool need to be configured in the configuration file of the tool with exact exit codes for success, success with warning and failure.
3. If cluster health checking tools exit with a different exit code for different conditions like success, failure, and warning, then those exit codes need to be configured in the configuration file.
4. The cluster health checking tools should check the environment variable HC_VERBOSE to run health checking tool in verbose mode. If HC_VERBOSE=1, then cluster health checking tool should run in verbose mode.
5. In case of errors or warnings, the messages should be printed to "stderr" which hcrun command grabs and logs in the summary file along with exit codes. The hcrun also saves the stderr output along with stdout in the file defined by master configuration keyword "toolsoutput".
6. If the tool is developed to start execution only on lthe ocal node then the keyword "startTool" should be set to "local" in the configuration.
7. If the tool's binary/script is available on all compute nodes at a specified location, then it is not required to copy the binary to compute nodes before starting the execution. The configuration keyword "copy" could be set to "no" so that hcrun would not copy the binary/script to the compute nodes before starting execution.
8. The cluster health checking tools can access HC_NODERANGE environment variable to know the set of nodes user specified to hcrun command.
9. The cluster health checking tools can access HC_HOSTFILE environment variable to know the host file generated from the node range. There will be one hostname per line in the host file generated from node range.
10. The cluster health checking tools can also use HC_USER environment variable to access the user id specified in Master configuration file. The HC_USER is generally used for specifying non-root user id. The tools which want to use non-root user id during execution can use the user id defined by HC_USER. This is necessary for tools that use PE runtime.
11. The cluster health checking tools which want to use the hosts where UFM servers (or other subnet manager servers) are running can access the environment variable HC_SUBNET_MANAGERS. The environment variable HC_SUBNET_MANAGERS is set with subnet managers specified in Master Configuration file.
12. The cluster health checking tools which wanted to use the IB switches xCAT node group can access the environment variable HC_IB_SWITCHES. The environment variable HC_IB_SWITCHES is set with ib_switches specified in Master Configuration file.

4.0 Reference

1. The details about the xdsh command can be found at http://xCAT.sourceforge.net/man1/xdsh.1.html
2. The details about the xcoll command can be found at http://xCAT.sourceforge.net/man1/xcoll.1.html
3. The details about the xCAT `sinv`command can be found at http://xcat.sourceforge.net/man1/sinv.1.html