# CSM Diagnostic

# Version 1.5.0

# User Guide

March 2019

Author:
Alda Ohmacht, Sandy Kaur

## Change log

| Version | Author | Date | Description |
|---|---|---|---|
| 1.4.0 | Alda Ohmacht | 10/31/18 | daxpy, dgemm, dgemm-gpu and jlink testes changed to reflect the Spectrum MPI healtcheck new directory structure. The POWER[8\|9] subdirectory was added. Directory is: /opt/ibm/spectrum_mpi_/healthcheck/POWER[8\|9]/ |
| | | 11/02/18 | daxpy-per-socket and dgemm-per-socket tests were modified to use the implementation from Spectrum MPI healthcheck. |
| 1.5.0 | Alda Ohmacht | 1/15/19 | nvvs test: handle WARN returned by nvidia nvvs program |
| | | 1/31/19 | Tested with HTX version 506 |
| | | 2/26/19 | chk-nfs-mount test: nfs protocol added as parameter in addition to the filesystem mount point |

## Table of Contents

List of Figures



List of Tables

# 1. Introduction

## *1.1 Overview*

The Diagnostics framework provides mechanism to test hardware and software components and then, organize the test results. This is done by running a suite of tests across multiple nodes simultaneously and analyzing the resulting errors and data. The results are stored in the CSM DB for future reference and ease of use, and the detailed results are stored in the Big Data Store (BDS).

Results of the Diagnostic run can also be used as the criteria to make the node available/unavailable to the users. Examples:

- A pre-defined set of tests (bucket) can run as Node mode, as part of a xCAT post-boot script. Diagnostic should run as the last script of the post-boot process, to make sure processes and network are all up. If Diagnostic returns 0, the node can be made available to the users.
- A pre-defined set of tests (bucket) can run as part of the job prolog and epilog (Node mode), making sure the job can proceed and/or the node is left in an good condition for the next job .
- A pre-defined set of tests (bucket) can run periodically to make sure nodes are still in the same state or not. Nodes that were marked unavailable can become available and vice-versa.

The framework is driven by configuration files that can be easily customized and extended.

Three configuration files are installed in the Diagnostic configuration directory: a configuration file providing the overall configuration of the diagnostic subsystem, a configuration file for the tests, groups and buckets, and a configuration file that describes the cluster and its nodes.

The exact syntax of the configuration files is described in the appendix *Configuration files*.

The framework is developed in Python 2.7.5, uses CSM command line APIs to interact with CSM. CSM APIs are being developed in C. The tests integrated into the Diagnostic framework is written in bash script and perl.

Before describing the details of the Diagnostics framework, we present the following definitions:

> ***Test:*** A single executable running on a single or multiple node that exercises one portion of the system.
>
> ***Test Group***: An abstract collection of test cases based on specific hardware coverage, for example memory, processor, GPU, and network.

***Test Bucket*:** A collection of tests grouped according to some principle. For example, a bucket might be based on node health. Alternatively, it might be based on the thoroughness of the health check or on the network stress it provides. A test bucket will typically contain multiple test groups.

Diagnostics runs in two modes, based on the type of nodes on which they run:

***Management mode:*** In this mode, Diagnostics run on the Management node, using the xCAT environment. It can target multiple nodes, run tests, and consolidate the output.

***Node mode***: In this mode, Diagnostics run on a node other than the Management node, with no xCAT environment. All the tests are executed on the node.

The Diagnostics application can be invoked by the Workload Manager, or from allocation prolog/epilog or standalone.

The Diagnostics are also responsible for running quick, non-intrusive tests that can run concurrently with user jobs, reference here as health check tests.

## *1.2 Terms and Concepts*

Diagnostic tests in this document includes hardware diagnostic tests (CPU, nest, I/O, GPUs, Infiniband), performance tests and health check status tests.

Target in this document refers to a single node or a set of nodes. It includes all types of nodes in the system: compute node, login node, management node, workload manager node, launch node.

## *1.3 Design Requirements*

- Facilitate the execution of diagnostic tests, gather and interpret results, and store the conclusions and failure data. Diagnostic tests will run during bringup phase, in production after any cluster event, software upgrade, hardware change, facilities change, cluster or nodes reboot and to detect hardware degradation
- Maximize diagnostics test coverage and minimize diagnostics run time.
- Help with the cluster administration priorities, constraints, and necessities.
- Provide results that are as human-readable as possible, allowing a customer to run diagnostics and determine FRU replacement without the need of a hardware expert.
- Provide the ability to run tests, that run concurrently with user jobs, “health check” support.
- Provide the ability to run as pre and post job.
- Support run diagnostics on nodes other than compute nodes, i.e., login node, launch node, etc.

- Integrate Diagnostics into Coral Big Data Solution.
- Easy to integrate new tests.

# 2. Architecture

In most of the cases, the Diagnostic framework runs as a privileged user.

Diagnostics can also run in a non-CSM environment. Without CSM, the diagnostics run, and results are not stored in the CSM database, hcdiag_query command line is not applicable, there is no RAS integration and there is no concept of node allocation. A file *"result_sumary"* is created with the summary of the run and its content is displayed. Example: Running Diagnostic without CSMRunning Diagnostic without CSM.

Figure 1 shows the high-level overview of the Diagnostics and its environment. The modules in blue and gray shades are part of the Diagnostic Subsystem.

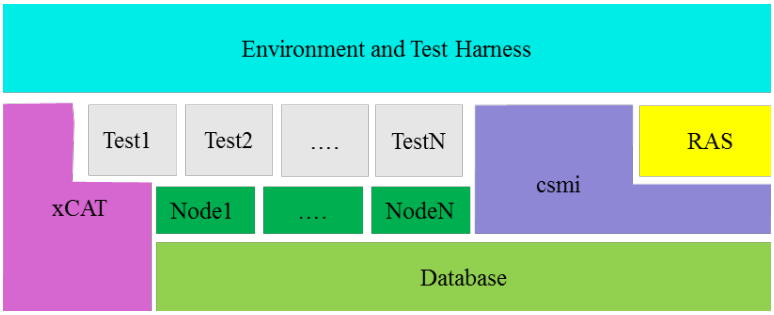


Figure 1: High level Overview

The diagnostic environment and harness module are responsible for determining and establishing the diagnostics environment. It manages abnormal termination, log directory structures, file locations, the scheduling of individual tests within a run, etc. It is responsible for parsing all command line parameters and configuration files. Also, it writes all output files not written by the tests themselves.
The diagnostic environment is also were testcases are encapsulated/unified for pass/fail determination.

The master and tests configuration files, and information retrieved from the database such as hardware components and their status, are used as input to the framework.

When running in Management mode, the harness checks if the test is present in the target nodes, or if it is local to the management node. If a test is local to the management node, the Harness will copy the test to the target nodes before execution.

For tests that are local to the management node, and requires more than one file, for example: a script, a configuration and a binary file, it is required that all files are in a subdirectory called the test name (see Test configuration file ).

The log output from the harness is organized under directory, either a default directory or a user specified directory. Inside this directory will be a main log file consisting of output from the harness. All output from the harness is found here. The file name has the run ID (a unique identifier of a Diagnostic run) embedded in it to clarify which run it came from. Subdirectories are created for each run and for each test that is run on the node. Each test subdirectory contains output logs specific to each test.

The harness log is sent to Big Data Solution.

The test modules are responsible for working with the various other objects to run tests and analyze the results. They parse the output, correlate it with other forms of data, determine if the test passed or failed, etc.

The CSM APIs module provides a well-defined interface to the functionality provided by various CSM components. CSM APIs also provides the interface to the database. CSM uses the PostgreSQL database

The CSMP APIs use the CSM infrastructure, component responsible for providing system services across the cluster network connecting all nodes. The service includes the database, RAS services, Big Data Store, etc.

The RAS module provides the interface for the RAS subsystem. Diagnostics play the "Publisher" role and generate RAS events for test failures. RAS events initiated by diagnostic programs will have the top-level string "hcdiag." The RAS event ID format is "toplevel.sublevel1.sublevel2".

## *2.1 Communication Protocol*

When running in Management mode, Diagnostic use the xCAT (Extreme Cloud Administration Toolkit) framework to communicate with remote nodes. xCAT is open-source distributed computing management software developed by IBM and used for the deployment and administration of Linux clusters.

The protocol used to talk to xCAT is a simple string-based command protocol. Replies are received as stdout and stderr strings (see also Security)

Dianostic uses the xCAT commands:

- xdsh: runs remote commands in parallel on remote nodes and/or Management.

- xdcp: concurrently copies files to or from remote target nodes.
- nodestat: displays the running status of each node in a noderange. It is used to validate the target passed as argument, as well to to check the status of the nodes.
- lsdef: list xCAT data object definitions

## *2.2 Security*

The xCAT framework supports SSL to secure the communication messages between Login Nodes, the Management Node, Compute Nodes and Service Nodes. It also supports using SSH as the secure remote shell to execute commands on remote nodes.

By default, only "*root*" on the management node has xCAT privileges and can run diagnostics.

A non-root user can be configured to run diagnostics, by following "Granting users xCAT privileges" page at https://sourceforge.net/p/xCAT/wiki/Granting_Users_xCAT_privileges/ -setup-sudo-for-xCAT-user

We do recommend the creation of a special user to run Diagnostic to facilitate to run all the tests: tests that require sudo privileges and tests that are not recommended to run as *root*. We also recommend the creation of a new group and the Diagnostic user be part of this group. This group will have CSM api privileges.

In this document, we assume *diagadmin* the user authorized to run Diagnositc, and the group will be referenced as *csmadmin*.

These are the main recommended steps to setup a new user to run Diagnostic.

- Creates the *csmadmin* group
- Creates a Linux user, *diagadmin,* in all nodes of the system and add it to the *csmadmin* group.
- Grant xCAT privileges to *diagadmin*
- Grant sudo privileges to *diagadmin*.
- Add *csmadmin* group to the *csm_api.acl* file (it is located in the **"/etc/ibm/csm/csm_api.acl"**) in the "*privileged_group_id*" list.
- If pam module is used, add *diagadmin* to the whitelist.

Example of a *csm_api.acl* file and *diagadmin* setting.

```
# cat csm_api.acl
{
  "privileged_user_id": "root",

  "privileged_group_id": "csmadmin",

  "private":
  ["csm_allocation_query_details",
   "csm_allocation_delete",
   "csm_bb_cmd",
   "csm_allocation_step_query_details"],

  "public":
  ["csm_allocation_step_cgroup_create",
   "csm_allocation_step_cgroup_delete",
   "csm_allocation_query",
   "csm_allocation_query_active_all",
   "csm_allocation_resources_query",
   "csm_allocation_step_begin",
   "csm_allocation_step_end",
   "csm_allocation_step_query",
   "csm_allocation_step_query_active_all",
   "csm_diag_run_query",
   "csm_node_attributes_query",
   "csm_node_attributes_query_history",
   "csm_node_resources_query",
   "csm_node_resources_query_all"]
}


#  id
uid=642490(diagadmin)
gid=1043351(diagadmin)groups=1043351(diagadmin),1043350(csmadmin)
```

- Include *diagadmin* to the xCAT policy table, to allow *user* to run xCAT commands (see the “Granting users xCAT privileges” document.
  The user has to be able to issue the following commands: xdcp, nodestat, xdsh. This is specified in the police table. An example of a user that can issue all xcat commands in the police table is:

```
"7.0","diagadmin",,,,,,"allow",,
```

- Create SSL certificates for user *diagadmin.*

- Setup sudo privileges for user *diagadmin*, by creating the file diagadmin in /etc/sudoers.d directory with the following contents:

```
diagadmin     ALL=(ALL)       NOPASSWD: ALL
```

Or adding the following line in /etc/sudoers file:

```
%diagadmin     ALL=(ALL)      NOPASSWD: ALL
```

Make sure sudo package is installed on all nodes.

- The management node must be defined in the xCAT database in order to run tests on the Management node, tests with "*targetType = Management*" in the

```
xcatconfig -m
```

See the *chk-sys-firmare, ppping, chk-csm-health* as examples in /opt/ibm/csm/hcdiag/etc/test.properties file.

## *2.3 Invocation*

There are three methods to initiating the diagnostics system.

1. The first, standalone mode, with or without the work load manager as shown in **Error! Reference source not found.**Figure 2: Diagnostics flow.

2. The second method is starting diagnostic routine as part of a pre-job or post-job execution that can be job dependent. The prolog and epilog scripts are specified at job allocation time as shown in Figure 3: Diagnostics in job prolog / epilog flow.These scripts run with privileged credentials and have full access to the CSM database.

3. RAS event handling is the third method of invoking diagnostics (also shown in Figure 2: Diagnostics flow)

### *2.3.1 Standalone*

In standalone mode, the Diagnostic is initiated by a command line. This command line might be for a single test, a list of tests, or a test bucket and can run on a single node or set of nodes, if running in Management mode. As shown Figure 2, the diagnostics subsystem will parse the run request, call the CSM APIs to initiate a diagnostic run allocation and start the diagnostic tests. The diagnostic status will be stored in the CSM database and the log file will be stored in the Big Data Store. The output of the individual tests will be stored on disk and is referenced by the Big Data Store as well as the database.

A diagnostic failure might result in a RAS event. It should also be noted that some diagnostic programs can run directly under the control of the Workload Manager. This allows, for example, a thorough test bucket to be run on a node that is merely deemed as suspicious while the node is still an active part of the cluster.

Running diagnostics without the Workload Manager allows diagnostic tests to run on compute nodes and non-compute nodes.
The "create allocation" request with type "diagnostics", CSM API *csm_allocation_create*, allows allocation requests that include nodes that are not available to the Workload Manager: nodes that are not Compute Nodes or Compute Nodes that are not been used by the Workload Manager.

Diagnostics runs outside the Workload Manager are recommended for highly intrusive tests. Simple check tests that do not require an allocation.

Although not shown, diagnostics can run without the CSM infrastructure.

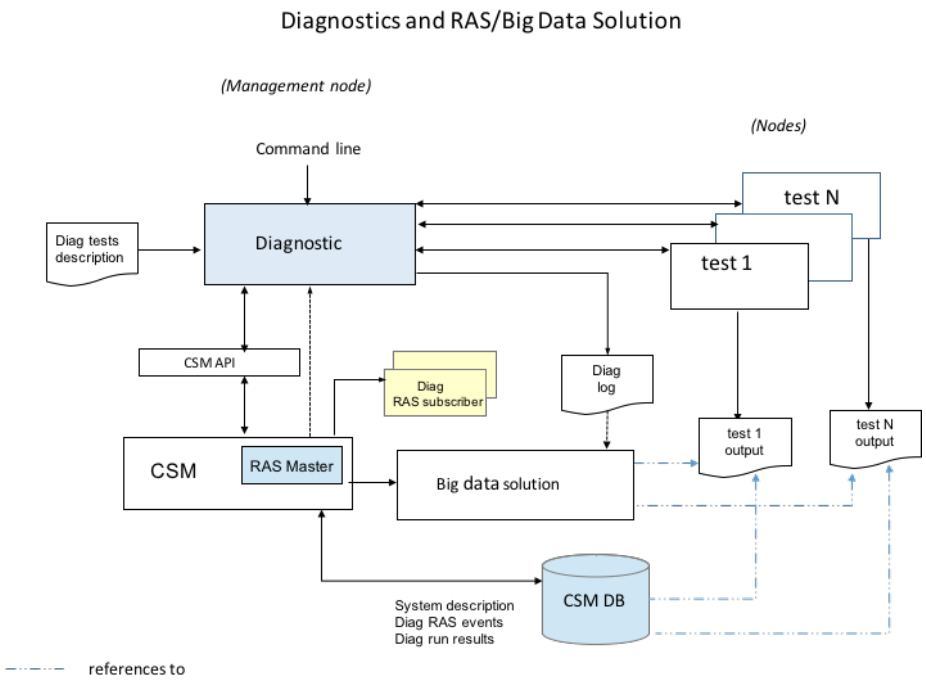


Figure 2: Diagnostics flow

### *2.3.2 Job prolog / epilog*

Diagnostics can be run as part of allocation prolog and epilog. The prolog and epilog flags are specified at job allocation time. Users can specify user flags to the prolog and epilog script, at job submission time. The system administrator can define system flags at the scheduler queue level. These user and system flags are passed to the CSM allocation APIs.

Users specify *-alloc_flags* to the LSF submission command *bsub*:

*bsub –alloc_flags*

*-alloc_flags* will be introduced to LSF to allow users to specify prolog / epilog flags.

A System administrator may specify system flags for any jobs submitted to a particular queue by using the following keyword for that queue in lsb.queues:

*CSM_REQ=[jsm=y|n][:step_cgroup=y|n][:core_isolation=y|n][:cn_mem=<mem>][:alloc_flags=<string>]*

For example:

*CSM_REQ=jsm=y:step_cgroup=n;alloc_flags=node_health*

*CSM_REQ* will be introduced by LSF to allow CSM-specific functions to be specified at the queue level.

Demonstration of running Diagnostics as part of job/prolog is shown in Running Diagnostic via job prolog in the Appendix.

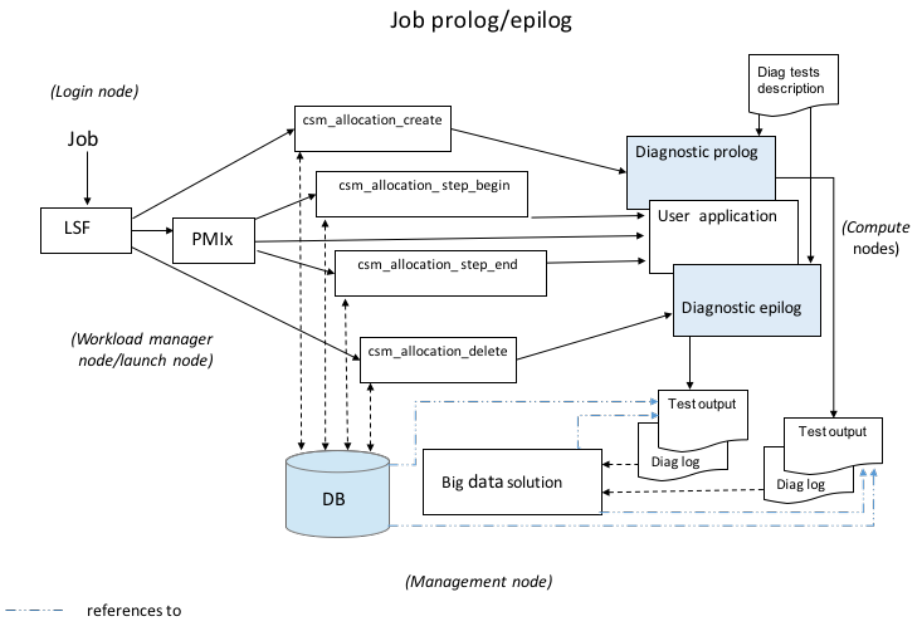


**Figure 3: Diagnostics in job prolog / epilog flow**

### *2.3.3 RAS event handling*

Diagnostic can be triggered by an RAS event. This is done by invoking Diagnostic from a RAS control action script.
The flow for this scenario is also shown in Figure 2: Diagnostics flow.
Diagnostic also creates. For the complete list of events created by Diagnostic, see

## 2.4 Database

Figure 4: Diagnostic database tables shows the Entity-Relationship (ER) model for the tables specific to the diagnostic subsystem. Diagnostic queries many other tables, using CSM APIs, to retrieve information about the system such as csm_node, csm_ras_type, csm_switch, etc. and updates the RAS event table. A complete list of database tables was provided in the SW33 milestone report.

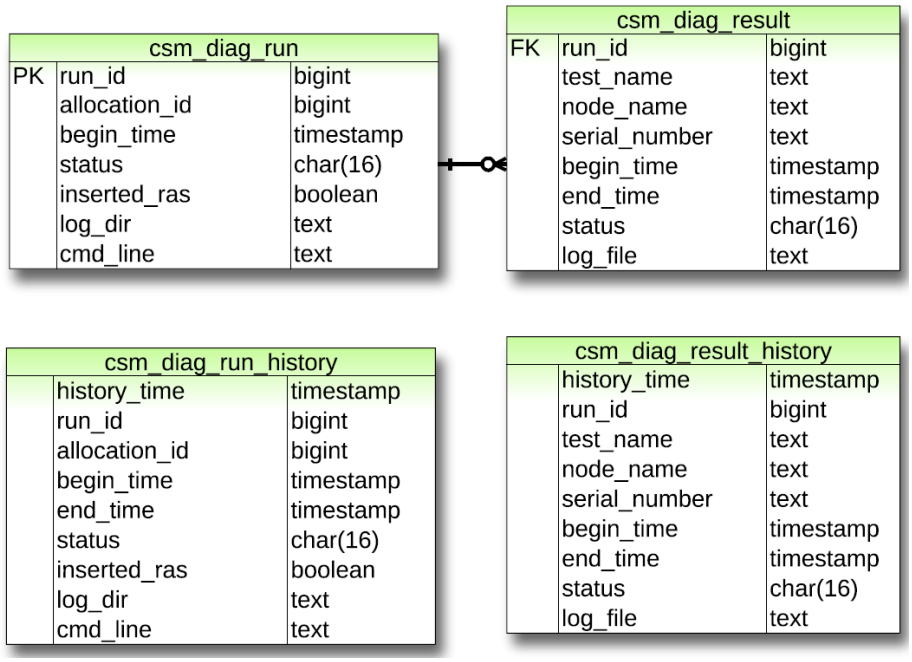


**Figure 4: Diagnostic database tables**

### *2.4.1 csm_diag_run*

| Column name | Comments |
|---|---|
| run_id | Unique identifier of a Diagnostics run. The run_id column format is: **"ddddddttttttttttt"**, where d is the date in compact form, t is the timestamp up to microseconds (6 digits)<br><br>PostgreSQL's max bigint is: 9223372036854775807. |

| | |
|---|---|
| allocation_id | The unique identifier of an allocation from the csm_allocation table or 0 indicating that Diagnostic is started with no allocation option. |
| begin_time | Date and time of the start of the run. |
| status | The termination status of the run. Possible values are: "*RUNNING*", "*COMPLETED*", "COMPLETED_FAIL", "*FAILED*" and "*CANCELED*". |
| inserted_ras | Indicates if a ras_message was created in this run. |
| log_dir | Full path of the root directory for the logs and output. The format is:<br><br><logdir>/<run_id><br><br>logdir is defined in hcdiag.properties file and it can be overwritten by command line with the argument –logdir. |
| cmd_line | The command issued to start Diagnostic issued by the user and its parameters. |

**Table 1: csm_diag_run**

### *2.4.2 csm_diag_result*

| Name | Comments |
|---|---|
| run_id | The unique identifier of a run from csm_diag_run table. |
| test_case | The name of the Diagnostic test case. Possible values are listed in the /opt/ibm/csm/hcdiag/etc/test.properties file. |
| node_name | The node name where the test ran. |
| serial_number | Serial number of the node name. |
| begin_time | Date and time of the start of the test run. |
| end_time | Date and time of the end of the test run. |
| status | The result of the test on this node: PASS or FAIL. |
| log_file | Full path and name of the output produced by the test. Format is: <log_dir>/<test_case>/<node_name>-<yyyy-mm-dd-hh:mi:ss>.output |

**Table 2: csm_diag_run_result**

## 2.5 CSM APIs

Table 3: Diagnostics CSM API, describes the APIs created specifically for the Diagnostic framework.

The Diagnostic framework uses CSM APIs not described here, for example: csm_ras_event_create, csm_node_atributes_query, etc. For complete list of CSM APIs, please refer to SW33 milestone report.

| API Function Name | Comments |
|---|---|
| csm_diag_result_create | Used by an application to update CSM DB with the result of a diagnostic test execution. |
| csm_diag_run_begin | Used by an application to record that a diagnostic run has started. |
| csm_diag_run_end | Used by an application to record in the “csm_diag_run” table of the CSM DB that a diagnostic run finished. |
| csm_diag_run_query | Used by an application to retrieve information about a Diagnostics run. |
| csm_diag_run_query_details | Used by an application to retrieve information about a Diagnostics run and a diagnostics result. |

**Table 3: Diagnostics CSM API**

# 3. Tests

This section describes the diagnostic tests IBM delivers as part of the Coral Diagnostic.

Few definitions prior to describe the tests:

**Test type:**

- Non-intrusive tests are status check programs that ensure that all of the hardware and software is configured as expected, do not have an unexpectedly large number of recovered errors, and is not missing any critical components. These nonintrusive tests should be run at system install, upgrade and periodically to ensure that there are no unexpected system changes. These tests can run in parallel with user jobs. We will refer these tests as health checks.
- Intrusive tests need to be run on a dedicated node(s). This type of test cannot run in parallel with user jobs. It is strongly recommended that these tests be run at initial system installation and whenever there is a system change, such as a software or hardware upgrade. Tests that might affect a user's job performance are also considered "intrusive".

**Test category**:

- 1st pass: Used to find suspicious hardware or configuration. Provides a wide coverage to detect hardware degradation. In general, the first-pass tests have short durations, simple configuration and light resource usage.
- 2nd pass: Used for in-depth analysis. The second pass tests are not intended to be used on a regular basis because they may take longer than the first-pass tests with more complex configuration.

Coral Diagnostic application integrates the following tests. They are categorized according to the overall area of coverage: Power, Network, GPU, Filesystems, etc.

NOTES:

- The tests scripts might need to be customized. We assume default directories for tests that invoke scripts/binaries installed by third party rpms, that might not match the user installation.
- Some configuration variables need to be modified according to the customer requirements.

- For Spectrum MPI tests: daxpy, dgemm, dgemm_gpu and jlink, please make the appropriate changes as instructed by the README file in /opt/ibm/spectrum_mpi/healthcheck/POWER[8|9]/<test>/README.<test>.
- Currently, running Diagnostic's dgemm, dgemm-gpu and jlink tests via LSF are not supported.

### *3.1 System*

Table 4: Base system tests lists the tests for diagnosing CPU, cache, and DRAM Memory, which are POWER-based components.

This is a list of diagnostic tests that are normally run on a single Power node. These tests require the htx package to be installed (See HTX package), and htxd daemon running. To avoid having an extra daemon running on the nodes all the time, the Diagnostic starts htxd daemon (if is not running already) and stops it (if it was started by Diagnostic).

Successfully completing these tests implies that the node is healthy for non-GPU, non-parallel workloads. Unless noted, the diagnostics in this section are *intrusive*. It should be noted that all of the hxe* tests are IBM written and may overlap with vendor diagnostic tests.

**Table 4: Base system tests**

| *Name* | *Coverage* | *1st pass* | *2nd pass* | *Description* |
|---|---|---|---|---|
| **hxecache** | Cache | Y | N | Stress tests L1/L2/L3 caches<br>- Has L2/L3 rollover tests and L2 cache line bouncing test.<br>- Can run various pre-fetch irritators alongside cache tests.<br>- Supports basic, transient, N stride, Partial pre-fetch types.<br>- Rule file tunable to avoid using specified chiplet(s) to indirectly test dynamic ECO.<br>- Pre-fetch threads randomize DSCR fields. |
| **hxecpu** | Generic processor core features | Y | Y | Test Power processor core by generating pseudo random sequence of instructions<br>- Supports testing of most user mode instructions<br>- Supports consistency checking<br>- Supports CPU thread specific instructions biasing and FP data corners biasing (number line)<br>- Capable of generating test instructions throughput of few MIPS |

| | | | | |
|---|---|---|---|---|
| | | | | - Seed based test case recreation ability |
| **hxediag** | Network | Y | N | Diagnostic exerciser for Broadcom NIC, Mellanox RoCE and IB adapters. |
| **hxeewm** | Thermal | N | Y | Stress test thermal aspects of Power server.<br>- Includes various workloads to stress processor core and memory<br>- Provides processor thread level workload control.<br>- Perform TDP, RDP, MSRP type of tests. |
| **hxefabricbus** | Fabric bus stress | N | Y | Supports inter node and intra node bus testing<br>- Intra node links test<br>- Inter node links test<br>- Supports various data patterns to be sent over links<br>- Allows to hand code processor/memory mappings to customize stress on links |
| **hxefpu64** | Floating point Unit | Y | Y | Test VSU (Vector Scalar Unit) of Power processors by generating pseudo random sequence of instructions.<br>- Supports testing of BFP, DFP, VMX, VSX instructions.<br>- Supports both correctness and consistency checking.<br>- Supports CPU thread specific instructions biasing and FP data corners biasing (number line)<br>- Capable of generating test instructions throughput of few MIPS.<br>- Seed based test case recreation ability<br>- Supports creation and execution of a macro to focus on hand coded list of instructions<br>- In case of miscompare, exerciser generates detailed miscompare report<br>- At the end of test execution, provides instructions coverage report |
| **hxemem64** | Memory subsystem | Y | Y | Stress test memory subsystem of Power. Aims at max local memory accesses. Tlbie buster can be added in the 2nd pass<br>- Can test large amount of system real memory by allocating it through OS shared memory and performing WRC ops on available memory.<br>- Supports variety of patterns (data pattern through file, immediate pattern, special patterns)<br>- Can test all available and configured physical page sizes and segment sizes<br>- Supports various test operations on memory segments (sequential access, cache line hoping, any size hoping)<br>- Support multiple threads |

| | | | | |
|---|---|---|---|---|
| | | | | - Support testing L4 cache. |
| **hxerng** | Random number generator engine | N | Y | Stress random number generator engine. |
| **hxesctu** | Cache | N | Y | Test processor cache coherency<br>- Hits at false shared cache line from up to 8 cpu threads to stress cache coherency logic<br>- Supports CPU thread specific instructions biasing.<br>- Capable of generating testcase using most user mode power processor instructions<br>- Seed based test case recreation ability |
| **hxestorage** | Storage subsystem | Y | Y | Tests storage subsystem by using end devices as target<br>- Performs raw mode storage ops with data integrity checks on variety of storage devices ( LV, RAID, JBOD, SAN, Flash/HDD, Dynamic Tiers, DVDRAM)<br>- Multiple threads with capability to control write/read ratio tunables for transfer length, random/seq access ratio, buffer alignment<br>- Metadata for each LBA to isolate data integrity issues<br>- Separate performance and validation coverage modes |
| **hxenvidia** | GPU | Y | Y | (intrusive)<br>Exerciser for NVIDIA GPU in IBM Power System. |

### *3.2 Network*

This section lists network interface card (InfiniBand) diagnostics, which are components provided by Mellanox. These tests ensure the soundness of the networking infrastructure. As can be seen from the table below, some of these tests are very intrusive. However, there is a wide scope of error conditions that are diagnosed on both the HCAs and switches by these tests.

| ***Name*** | ***1st pass*** | ***2nd pass*** | ***Description*** |
|---|---|---|---|
| **chk-capi** | Y | | Checks if the node is CAPI enabled. |
| **chk_hca_attributes** | Y | N | (non-intrusive)<br>Verify HCA board id and firmware version.<br>(ibv_ devinfo) |
| **chk-ib-node** | Y | N | (intrusive) |

| | | | |
|---|---|---|---|
| | | | Checks nodeIBbandwidth |
| **chk-ib-pcispeed** | Y | N | (intrusive)<br>Checks IB adapters vendor, speed and width |
| **chk-mlnx-pci** | Y | | (intrusive)<br>Checks Mellanox PCI configuration. |
| **chk-mlxlink-pci** | Y | | (non-intrusive)<br>Check Mellanox link errors |
| **compdiag** | Y | | (intrusive)<br>Comprehensive fabric diagnostic test using multiple ibdiagnet options. Superset of the atomic ibdiagnet tests. |
| **hcatest** | Y | | (intrusive)<br>Run verbs latency test to verify operating HCA.<br>(ib send lat) |
| **ibcredit** | Y | | (non-intrusive)<br>Check for credit loops.<br>(ibdiagnet) |
| **ipoib** | Y | | (non-intrusive)<br>Shows the state, maximum transmission unit (MTU), and mode of the configured IP over InfiniBand (IPoIB) devices on the compute Node. |
| **ipverbs** | Y | | (intrusive)<br>verify verbs point to point communication between nodes |
| **lnklwls** | Y | | (non-intrusive)<br>Verify link width and speed.<br>Use skip option to remove unnecessary checks (fabric discovery, cable, duplicate lid checks, etc.).<br>(ibdiagnet) |
| **lnkqual** | Y | | (non-intrusive)<br>Check for marginal link error [ber thresh (10e14)].<br>(ibdiagnet)<br>(non-intrusive) |
| **jlink** | Y | | (intrusive)<br>Measures the bandwidth between compute nodes and finds bad or low-performing links. The test has a wide variety of options, including an option to verify the validity of data sent across the links. |
| **ppping** | Y | N | (non-intrusive)<br>Determines whether the configured IPoIB devices are configured correctly and reply to local ping requests. All given nodes ping each other, showing that the nodes are reachable. |
| **swhealth** | Y | | (non-intrusive)<br>Switch health report: power supply, fan, etc. |

Table 5: Network tests

### *3.3 GPUs*

Table 6: GPU related diagnostic tests lists the tests available for the GPU.
See also hxenvidia test in Table 4: Base system tests.

The NVIDIA Data Center GPU Manager (DCGM) is the framework within which diagnostics can be run, including the existing NVVS test, new online HW diagnostics and new background health check.

DCGM provides an exhaustive suite of GPU tests for diagnostics, health check, behavior monitoring, configuration management, policy oversight, and statistics. These tests are an integral part of the management infrastructure for an enterprise data center with hybrid compute nodes.
There are three main groups of NVIDIA diagnostics, the results are available as PASS, FAIL (and in some cases) WARN.

***Background health checks***

An active health monitoring & analysis is a set of non-intrusive tests that asses the health of the GPU subsystems, such as PCIe, SM, MCU, PMU, Inforom, Power and thermals system. It can run during fixed polling intervals as part of the standard job execution environment. These tests will check for GPU hang, data corruption, performance issues, changes in clock frequencies and other unexpected changes to the environment. CSM will classify the results as errors or attention messages. All messages will be logged and visible to the user, while errors may potentially terminate the job.

***Online diagnostics***

These diagnostic tests are intrusive, requiring exclusive access to the target GPUs. They validate device sub-components, interlink bandwidth, memory/ECC state and deployment is software integrity. Depending on the depth of coverage, these diagnostics are designed to complete within a few seconds to an hour (short, medium and long duration) and can be performed at job prologue/epilogue, when a job fails, or as part of the regular diagnostic run schedule.

Unencrypted summary data with time series statistics will be stored in the log analysis system. For HW diagnostics an additional encrypted log file is saved for NVIDIA analysis.

***Offline diagnostics***

The offline diagnostics(the NVIDIA Field Diag) are a highly intrusive programs that requires a special software environment. The software environment may require a reboot to set up and require that the node unavailable for scheduling by the workload manager. The duration of these diagnostics may be up to 8 hours. The results are available as PASS or FAIL. A simple unencrypted description of the failing test is available, while an

additional encrypted log file will be saved for NVIDIA analysis. NVIDIA will provide a full list of items these diagnostics cover, the data collected, and actions taken.

The Field Diag includes HW coverage for NVLINK, including the GPU – CPU path in NVLINK 2.0.

**Table 6: GPU related diagnostic tests**

| *Name* | *1st pass* | *2nd pass* | *Description* |
|---|---|---|---|
| **chk-gpu-ats** | Y | | (non-intrusive)<br>Checks if the GPU ATS is enabled or not by looking at the content of /sys/module/nvidia_uvm/parameters/nvm8_ats_mode file. |
| **chk-nvidia-clocks** | Y | | (non-intrusive)<br>Checks NVIDIA GPU Application Clocks (Graphics and Memory) |
| **chk-nvidia-smi** | Y | | (non-intrusive)<br>Checks possible stuck nvidia-smi |
| **chk-nvidia-vbios** | Y | | (non-intrusive)<br>Checks the GPUs vbios. |
| **chk-nvidia-speed** | Y | | (non-intrusive)<br>Checks the nvlink status and speed |
| **dcgm- diag** | Y | | *(intrusive)*<br>Online Diagnostic. *DCGM diag* detects and troubleshoots common problems affecting GPUs in a high-performance computing environment, without requiring the node to be taken offline. Failure of the HW diagnostics components of this test are not grounds for RMA – the Field Diag must subsequently run for final confirmation. |
| **dcgm-diag-double** | | Y | *(intrusive)*<br>Online Diagnostic. *DCGM diag* double precision errors. |
| **dcgm-diag-single** | | Y | *(intrusive)*<br>Online Diagnostic. *DCGM diag* single precision errors. |
| **dcgm-health** | Y | | *(non*-intrusive)<br>DCGM provides a series of health checks for monitoring the state of the GPU HW and driver stack during running jobs. |

| | | | |
|---|---|---|---|
| **dgemm-gpu** | Y | | *(intrusive)*<br>Check GPU performance |
| **fieldiag** | | Y | *(intrusive)*<br>Offline diagnostic. Tests in the package provide confirmation of the numerical processing engines in the GPU, integrity of data transfers to and from the GPU, and test coverage of the full onboard memory address space available to CUDA programs. This test requires several hours for full coverage.<br>NVIDIA ships this test in a tar file. The tar file contains the fieldiag binary and the source code for the kernel module. |
| **gpudirect** | | Y | (intrusive)<br>(Exercise GPUDirect RDMA and Peer to Peer data movement and basic Async functions. |
| **gpu-health** | Y | | (intrusive)<br>Check gpu memory bandwidth, gpu dgemm flops and nvlink transfer speeds per socket.<br><br>NOTE: The source code for this test is shipped in hcdiag/samples and should be compiled by the customer. |
| **nvvs** | Y | | *(intrusive)*<br>Online Diagnostic*. D*etects and troubleshoots common problems affecting GPUs in a high-performance computing environment, without requiring the node to be taken offline. Failure of the HW diagnostics components of this test are not grounds for RMA – the Field Diag must subsequently run for final confirmation.<br><br>NOTE: This test will be discontinued. |
| **p2pBandwidthLatencyTest** | Y | | (intrusive)<br>This test demonstrates the CUDA Peer-To-Peer (P2P) data transfers between pairs of GPUs and computes latency and bandwidth. Tests on GPU pairs using P2P and without P2P are tested.<br><br>NOTE: The source code for this test is shipped as part of the CUDA samples and should be compiled by the customer. |

### *3.4 Storage and Filesystems*

See also hxestorage in Table 4: Base system tests.

Table 7: Storage and Filesystems tests

| *Name* | *Coverage* | *1st pass* | *2nd pass* | *Description* |
|---|---|---|---|---|
| **chk-gpfs-mount** | Filesystem | Y | | Checks if the GPFS Filesystem is mounted correctly. |
| **chk-nfs-mount** | Filesystem | Y | | Checks if the NFS Filesystem is mounted correctly. |
| **chk-nvme** | Storage | Y | | Checks NVMe device vendor and firmware |
| **chk-nvme-mount** | Storage | Y | | Checks if NVMe device is mounted correctly. |
| **gpfsperf** | gpfs performance | | Y | Measures the real I/O performance on the disk in conjunction with the network and the GPFS file system. This program is delivered with the GPFS product. |
| **mmhealth** | gpfs | Y | | Verify status of the gpfs in the node |
| **nsdperf** | Network performance | | Y | Tests the network performance, simulating GPFS NSD protocol traffic. Tests the network IO without involving disk I/O. nsdperf program is delivered with the GPFS product. |

## *3.5 Miscellaneous*

Table 8 in this section provides a list of tests that are not related to GPU, or Power or Network.

System stability and performance is often reduced by unexpected configuration changes or intermittent recoverable failures. These status checks can periodically be executed on a running system to verify the configuration and status of the system with no impact on performance. The health checks read information from necessary devices in a short amount of time and return the results to the user. Those checks are intended to monitor the status of hardware and software.

Table 8: Miscellaneous tests

| *Name* | *Coverage* | *1st pass* | *2nd pass* | *Description* |
|---|---|---|---|---|
| **chk-aslr** | Node configuration | Y | | Checks if node ASLR is enable/disabled |
| **chk-boot-time** | Node | Y | | Checks if the nodes were boot successfully |
| **chk-cpu** | CPU | Y | | Collects CPU and governor information. |
| **chk-cpu-count** | Node | Y | | Checks the number of CPUs on the node |
| **chk-csm-health** | CSM infra structure | Y | | CSM infrastructure health check |
| **chk-free-memory** | Node | Y | | Checks the amount of free memory |
| **chk-idle-state** | Node configuration | Y | | Checks node idle state configuration |
| **chk-kworker** | Node | Y | | Checks kworker process consuming more than the threshold. |
| **chk-led** | Node | Y | | Identifies the faulty LEDs on the node |
| **chk-load-average** | Node | Y | | Check if the node load average is below the threshold |
| **chk-load-cpu** | Node | Y | | Checks if the node cpu load is below the threshold |
| **chk-load-mem** | Node | Y | | Checks if the node memory load is below the threshold |
| **chk-memory** | Node | Y | | Checks the node total memory, number of banks and bank size |
| **chk-noping** | Node | Y | | Checks if the nodes that are booted are pinging. |
| **chk-os** | Node | Y | | Checks operating system and kernel version on a node. |
| **chk-power** | Power Management | Y | | Read the current power capping and power shifting values on the node and compare these values with current power use on the node, CPU/GPU. |

| | | | | |
|---|---|---|---|---|
| **chk-process** | Node | Y | | Checks if a list of process is running on the node |
| **chk-smt** | Node Config | Y | | Checks the node SMT setting. |
| **chk-sw-level** | Node | Y | | Checks the node software stack and levels |
| **chk-sys-firmware** | Node | Y | | Checks firmware levels on the node from the management node. |
| **chk-sys-firmware-local** | Node | Y | | Checks machine firmware levels. |
| **chk-temp** | Node | Y | | Checks all temperature sensors on a node |
| **chk-zombies** | Node | Y | | Check possible stuck zombie tasks |
| **daxpy** | Memory performance | Y | | Checks the memory bandwidth on a compute node. |
| **daxpy-per-socket** | Memory performance | | Y | Checks the memory bandwidth per socket on a compute node. |
| **dgemm** | CPU performance | Y | | Checks the CPU performance on a compute node. |
| **dgemm-per-socket** | CPU performance | Y | | Checks the CPU performance per socket on a compute node.<br>It runs dgemm twice, one using socket 0's CPUs, and an second time using socket 1's CPUs. |
| **rvitals** | Node | Y | | Checks if rvitals are within the thresholds. |
| **test-simple** | Node | Y | | Simple hello test |

# 4. Installation

The Diagnostic rpm file, **ibm-csm-hcdiag** contains the framework and most of the tests interface. It needs to be installed on all nodes that Diagnostic tests will run.

It will install the following:

- */opt/ibm/csm/hcdiag/bin*, main default root directory for Diagnostic framework
- */opt/ibm/csm/hcdiag/etc.,* default location of the configuration files
- */opt/ibm/csm/hcdiag/tests,* default location for the Diagnostic test files
- */opt/ibm/csm/hcdiag/samples,* default location for the source code of tests that is pre-requisite dependent and need to be compiled at the customer environment.

### *4.1 Framework*

The Diagnostic framework supports the following machines:

- Firestone, 8335-GTA
- Garrison, 8335-GTB
- Witherspoon: 8335-GTC and 8335-GTW
- Boston: 9006-22C and 5104-22C

### *4.2 Tests*

Tests integrated into the diagnostic framework support POWER9 based machines.

Tests are, by default, installed under: */opt/ibm/csm/hcdiag/tests/<testname>*. Each test should have its own subdirectory.

For example, for test *test-simple:*
*/opt/ibm/csm/hcdiag/tests/test-simple/test-simple.sh* is where the test will be installed and tests.properties file must have the minimal entry:

```
[tests.test-simple]
description = This is a simple test
executable  = /opt/ibm/csm/hcdiag/tests/test-simple/test-simple.sh
```

If the test exists on the target node, the framework just executes it. If the test does not exist on the target, the framework will copy it to the target prior to the execution. If only one file is found under the "*testname*" subdirectory, the framework uses the xCAT facility "copy and execute",

otherwise the framework will copy all the necessary files to the target nodes prior to the execution.

Other packages are required by Diagnostics tests, for example: tests that starts with hxe requires HTX package installed; dgemm, daxpy, dgemm-gpu and jlink comes from IBM Spectrum MPI package, etc.

### *4.2.1 HTX package*

The HTX package is installed by a rpm file, example: htxrhel72le-505-LE.ppc64le.rpm .
It installs the tests described in Table 4: Base system tests, starting with *hxe** .

HTX (Hardware Test eXecutive) is a system level test suite for validating system p server hardware design. HTX is a suite consisting of test programs for POWER processor core, nest and IO. HTX is used by various test labs across IBM like hardware system test, Manufacturing, Characterization, Server integration, bring-up etc. HTX supports both menu driven user interactive interface as well as automation friendly command line interface for controlling the test suite. HTX test suite can run on AIX as well as Guest and NV Linux (Ubuntu, RHEL, SuSE). HTX is also in use by OpenPOWER member companies.

HTX is now open and source is available @ https://github.com/open-power/HTX

Online information about HTX tests are found at https://w3-connections.ibm.com/wikis/home?lang=en - !/wiki/HTX-Users/page/Welcome
and online information about HTX releases and platform supported at
http://ausgsa.ibm.com/projects/h/htx/public_html/htx_release_info.html

Obtain the rpm file from IBM and install on all nodes.
HTX installation creates an entry in the /etc/init.d directory that should be removed, we don't want the htx daemon to start on the node, it will be started on demand.
Also, the installation process starts the htxd daemon automatically, it should be stopped.

The commands bellow assume that you are installing from the management node using xcat utilities.

```
$ xdsh <nodes> rpm –ivh htxrhel72le-506-LE.ppc64le.rpm
$ xdsh <nodes> rm /etc/init.d/htx.d
$ xdsh <nodes> /usr/lpp/htx/etc/scripts/htxd_shutdown
```

### *4.2.2 IBM Spectrum MPI*

The ibm-smpi package installs dgemm, dgemm_gpu, daxpy and jlink tests.
The executables are located under /opt/ibm/spectrum_mpi/healthcheck/POWER[8|9]/<test>.

For more details of each test, please refer to the readme file under /opt/ibm/spectrum_mpi/healthcheck/POWER[8|9]/<test>/README.<test>.

dgemm test requires IBM ESSL (IBM Engineering and Scientific Subroutine Library) and IBM XL Fortran installed in addition to spectrum_mpi.

#### *4.2.3 NVIDIA Datacenter*

The Diagnostic system will support GPU diagnostic via DCGM (NVIDIA Data Center GPU Manager). Obtain the rpm from NVIDIA and install datacenter on all nodes. The commands bellow assume that you are installing from the management node using xcat utilities:

```
$ xdsh <nodes> rpm –ivh datacenter-gpu-manager-1.4.1-1.ppc64le.rpm
```

For more details on the installation and pre-requisites refer to the CSM Installation and Configuration document.

#### *4.2.4 Hardware Lister package*

lshw (Hardware Lister) is a tool to provide detailed information on the hardware configuration of the machine.
Supported version: *lshw-B.02.18-7.el7.ppc64le.rpm*
It is used by chk-memory test.

### *4.3 Dependencies*

The minimal Diagnostic framework is:

- RHEl 7.5
- xCat 2.1.2 or later
- python 2.7.5

### *4.4 Validation*

To validate the basic diagnostic installation, run test-simple test:

```
/opt/ibm/csm/bin/hcdiag_run.py –test test-simple –target <noderange>
```

# 5. Appendix A

## *5.1 User interface*

### *5.1.1 Run Diagnosic*

```
hcdiag_run.py [-h] (--test t [t ...] | --bucket b | --list [item])
                     [--target n[,n ...]] [--nocsm | --usecsm]
                     [--noallocation | --allocation_id allocation_id]
                     [--fanout fanout_value] [--diagproperties filename]
                     [--testproperties filename]
                     [--clusconf filename] [--logdir dir]
                     [--stoponerror action] [--verbose level] [--version]

 arguments:
  -h, --help            show this help message and exit

  --test t [t ...]      test to run

  --bucket b, -b b      bucket to run

  --list [item], -l [item]
                        list available: test, group, bucket. Default is all

  --target n[,n ...], -t n[,n ...]
                        target on which to run health check/diagnostic. When
                        running in Management Mode, xcat noderange syntax is
                        accepted.

  --nocsm               do not use csm

  --usecsm              use csm

  --noallocation        do not allocate the target nodes

  --allocation_id allocation_id
                        allocation_id that reserves the target

  --fanout fanout_value, -f fanout_value
                        maximum number of concurrent remote shell command
                        processes

  --diagproperties filename
                        diag properties file

  --testproperties filename
                        test properties file

  --clustconf      filename
                        cluster configuration file, yaml format.

  --logdir dir          root directory for the log file
```

```
  --stoponerror action  define action if test fail {no: continue, node: stop
                        at node level, system: stop the run}

  --verbose level, -v level
                        stdout verbosity {debug, info, warn, error, critical}

  --version             show program's version number and exit
```

Mandatory arguments:

- *target* (when running in Management mode): node, list of nodes. Same syntax as xCAT noderange. Example: --target c931f021[2,4,6,8]
- *test* or *bucket* : run only the specified diagnostic tests|bucket. Tests are space separated. Example: --test dgemm daxpy

### *5.1.2 Query Diagnostic*

Query diagnostic run and results. Queries the database and returns formatted data to the user. This command is only valid if Diagnostic is integrated with CSM infrastructure.

```
hcdiag_query.py [-h] [--runid id [id ...]] [--allocation id [id ...]]
                      [--date DATE [DATE ...]] [--status STATUS [STATUS ..]]
                      [--ras {yes,no}] [--results] [-o file] [--version]

arguments:
  -h, --help            show this help message and exit

  --runid id [id ...], -r id [id ...]
                        unique udentifier of the diagnostic run. All other
                        arguments will be ignored.

  --allocation id [id ...], -a id [id ...]
                        unique udentifier of the allocation

  --date DATE [DATE ...], -d DATE [DATE ...]
                        date of the diagnostic run, format: yyyy-mm-dd-
                        hh:mm:ss. If a second date is passed, the interval
                        will be used in the query

  --status STATUS [STATUS ...], -s STATUS [STATUS ...]
                        status of the run {running, completed, completed_
                        fail, canceled, failed}

  --ras {yes,no}        ras message inserted for this run {yes, no}}

  --results             include results of the run, tests and status per node

  -o file, --output file
                        store the output into a file
```

```
--version             show program's version number and exit
```

## 5.2 Configuration files

There are three configuration files, the teste , the master, and the cluster configuration file. Default configuration files are provided.

### *5.2.1 Test configuration file*

The tests.properties file describes the tests supported by the diagnostics. This file also defines test buckets, a collection of tests, to make it easier to run a set of tests.

There are two types of section in this file:

- A "*test*" section, enclosed in square brackets, [], and starting with the word "tests.".
- A "*bucket*" section, also enclosed in square brackets, [], and starting with the word "bucket.".

Atrributes of a test section:

***[tests.test1]***
name of the test. Example: "test1"

*description*
short description of the test (documentation only)

*group*
group that the test belongs to (documentation only)

*timeout*
time in seconds, xCAT waits for output from any currently executing remote targets. Default is 10s.

*targetType*
type of the node the test applies to. Valid values are: compute, management

*executable*
full path of the executable. It is a mandatory attribute.

*args*
arguments for the executable, if any.

*clusterTest*
tells the framework if this is a single node test or a cluster test.

*xcat_cmd*
tells the framework that the executable is a xCAT command, and the nodes do not need

to be reachable.

Attributes of a bucket section:

*[bucket.node_check]*
name of the bucket. Example: node_check

*description*
short description of the bucket (documentation only)

*tests*
list of the test that are part of the bucket, comma separated.

### *5.2.2 Master configuration file*

The hcdiag.properties is the Diagnostic framework configuration file. It contains only one section, ***[MasterConfig]***, with the following attributes:

*testproperties*
full path of the tests properties file. If it is not defined in the hcdiag.properties file nor passed as command argument, /opt/csm/hcdiag/etc./tests.properties is assumed.

*clustconf*
full path of the cluster configuration file. If it is not defined in the hcdiag.properties file nor passed as command argument, /opt/csm/hcdiag/etc/clustconf.yaml is assumed.

*allocation*
tells the diagnostic framework if an allocation (node reservation) is necessary prior to run the tests. Valid values: yes,:no. Default: yes

*csm*
set if csm enviroment will be used or not. Valid values: yes|no. Default: yes

*stoponerror*
set the famework behavior if a diagnostic test fails. Continue, stop the tests on a failing node or stop the run. Valid values: no, node, system. Default value: no

*logdir*
directory where the Diagnostic output and test outputs will be written.Default: /tmp.

*console_verbose*
sets the verbose mode for the console . Valid values are: debug, info, warn, error, critical. Default value: info.

*log_verbose*
sets the verbose mode for the hcdiag log file and the syslog. Valid values are: debug, info, warn, error, critical. Default value: debug.

*installdir*
Diagnostic install root directory. Default: /opt/ibm/csm/hcdiag.

*xcat_bindir*
directory that contains the xcat binaries: xdsh, xdcp, nodestat.

*csm_bindir*
directory where csmi binaries is installed. Default: /opt/ibm/csm/bin.

*watch_output_progress*
if watch_output_progress is set to 0, the framework will wait for the remote test execution to complete or to timeout, whatever comes first. if set to non-zero, the framework will watch the remote test execution output in addition to the behaviour above.

*tty_progress_interval*
time in seconds used by the framework to show progress (display '.')
Default:2

*timeout*
xcat timeout, in second: timeout value for remote command execution.
If any remote target does not provide output to either stdout or stderr within the timeout value, xdsh displays an error message and exits.
Default: 10 , same as xCAT .

*xcat_fanout*
maximum number of concurrent remote shell command processes,
Default: 64, same as xCAT.

*tempdir*
directory used by the framework to copy the executables to the remote nodes
Default = /tmp/hcdiag

*common_fs*

when running Management mode, save the output and log in a common file system or let the framework save the output in the management node file system
(target nodes output is sent via stream pipe to the management node)
Default: no

### *5.2.3 Cluster configuration file*

The clustconf.yaml configuration file describes the cluster in YAML format . It defines the cluster's node or set of nodes grouped by its attributes. We have the following sections in the file:

#### node_info section

***node_info*:** # each case statement defines a subset of the system, that has common attributes.

**- *case*:** < node list or a regex expression that defines the node >
# See: case

***rvitals:*** < points to one of the rvitals definition in the rvitals section>
# See **Error! Reference source not found.**

***ncpus*:** < number of CPUs > *# /usr/bin/lscpu | grep ^'CPU(s)'*

***memory**:* # memory attributes . See details in memory attributes.
***total:*** **<**total amount of memory>
***banks:*** **<** number of memory banks (DIMMs>
***bank_size:*** <size of each bank>

***clock**:* # cpu clock attributes. See details in

clock attributes.
***max**:* < maximum cpu clock frequency >
***min*:** <minimum cpu clock frequency>

***firmware*:** # P9 firmware attributes. See details in firmware attributes.
***name:*** **<** identifier of the firmware>
***versions:*** # firmware versions
**-** '<line 1>'
**-** ' <line N>'

***gpu:*** **#** gpu attributes. See details in gpu attributes.
***pciids*:** # list of all nvidia pci devices IDs
**-** '<id 1>'
**-** ' <id N>'
***device*:** <official product name of the GPU>
***vbios:*** <GPU board vbios>
***clocks_applications_gr*:** <frequency of graphics clock>
***clocks_applications_mem*:** <frequency of memory clock>
***persistence_mode:*** <Enabled|Disabled>
***link_speed:*** <speed>
***nlinks:*** <number of links per gpu>

**ib:** # IB attributes. See details in ib attributes.
***slot_rx**:* < list of ib rx adapters>
***board_id*:** <adapter card's PSID (Parameter-Set Identification)>
***firmware*:** < adapter firmware version>

***os:*** # Operating System attribute. See details in os and kernel attributes
***name:*** "Red Hat Enterprise Linux Server
***pretty_name***: "Red Hat Enterprise Linux Server 7.5 (Maipo)"

***kernel:*** # kernel attributes. See details in os and kernel attributes.
***release:*** # kernel release

***software:*** # list of software used in chk-sw-level test. See examples in software attribute list.

***ufm:*** # UFM attributes
***ip_address**:* IP address of UFM
***user:*** username for authentication
***pw:*** password for authentication

***nvme:*** NVMe attributes. See details in nvme attributes.
***vendor**:* <NVMe device vendor name>

***firmware_rev:*** <NVMe device firmware level>

***temp:***

***celsius_high****:* <high threshold for sensor temperature testing, in Celsius >

***celsius_low****:* <low threshold for sensor temperature testing, in Celsius>

## node_info example

```
node_info:
  - case: (c699c0([0-5]+[0-9]))
    rvitals: wspoon_dd2
    ncpus: 176
    memory:
      total: 606
      banks: 16
      bank_size:  32
    clock:
      max: 3.50
      min: 2.0
    firmware:
      name: 1742FxRev10
      versions:
        - 'BMC Firmware Product:   ibm-v2.0-0-r46-0-gbed584c (Active)*'
        - 'HOST Firmware Product:   IBM-witherspoon-ibm-OP9_v1.19_1.179 (Active)*'
        - 'HOST Firmware Product: -- additional info: buildroot-2018.02.1-6-ga8d1126'
        - 'HOST Firmware Product: -- additional info: capp-ucode-p9-dd2-v4'
        - 'HOST Firmware Product: -- additional info: hostboot-4d6dfdd-p2fc8b56'
        - 'HOST Firmware Product: -- additional info: hostboot-binaries-20c1829'
        - 'HOST Firmware Product: -- additional info: linux-4.16.13-openpower1-p3acb43b'
        - 'HOST Firmware Product: -- additional info: machine-xml-94a137f'
        - 'HOST Firmware Product: -- additional info: occ-f796766'
        - 'HOST Firmware Product: -- additional info: op-build-v2.0.3-308-ge0ef930'
        - 'HOST Firmware Product: -- additional info: petitboot-v1.7.1-pc15aad2'
        - 'HOST Firmware Product: -- additional info: sbe-1b143b5'
        - 'HOST Firmware Product: -- additional info: skiboot-v6.0.5'

    gpu:
      pciids:
        - '0004:04:00.0'
        - '0004:05:00.0'
        - '0004:06:00.0'
        - '0035:03:00.0'
        - '0035:04:00.0'
        - '0035:05:00.0'
      device: "Tesla V100-SXM2-16GB"
      vbios: "88.00.13.00.02"
      clocks_applications_gr: 1530
      clocks_applications_mem: 877
      persistence_mode: Enabled
      link_speed : 25

    ib:
      slot_rx: "00(03|33):01:00.(0|1)"
      board_id: "IBM0000000002"
      firmware: "16.22.8036"

    os:
      name: "Red Hat Enterprise Linux Server"
      pretty_name: "Red Hat Enterprise Linux Server 7.5 (Maipo)"

    kernel:
      release: "4.14.0-49.10.1.el7a.ppc64le"

    software:
        - ibm-csm: 1.1.1-163
        - gpfs.base: 5.0.1
        - lm_sensors: 3.4.0-4.20160601gitf9185e5
        - datacenter: 1.4.2-1
        - cuda: 9-2-9.2.148-1
        - ibm_smpi: 10.02.00.04rtm0

    ufm:
      ip_address: "10.7.0.41"
      user: "admin"
```

```
    password: "123456"

  nvme:
    vendor: "Samsung"
    firmware_rev: "MN12MN12"
  temp:
    celsius_high: "35.0"
    celsius_low: "14.0"
```

Table 9 clustconf.yaml: node_info

## case section

Each case describes a subset of the nodes from the cluster. All nodes that belongs to the "case" statement have to have exactly the same characteristics.
Each "case" statement accepts a single node, a list of nodes, or nodes that follows a regular expression (like bash regular expressions).

Examples:

- One specific node
  - *case: node1*

- A list of nodes
  - *case: node(1|2|3)*
  - *case: (node1|node2|node3)*

- A set of nodes that follows the same syntax: the word node, follow by two digits
  - *case: node([0-9]+[0-9])*

- A set of nodes that starts with the string "node"
  - *case: node.**

- Two set of nodes, nodes in the range node50-node59 and node90-node99:
  - *case: (node5[0-9]|node9[0-9])*

### memory attributes

The command bellow shows how to obtain the attributes necessary for the clustconf.yaml, memory's total, banks and bank_size.

```
# sudo /usr/sbin/lshw -class memory -quiet |  grep -A 1000  "*-memory" |
egrep 'bank|size'

      size: 606GiB
     *-bank:0
          size: 32GiB
     *-bank:1
          size: 32GiB
     *-bank:2
          size: 32GiB
     *-bank:3
          size: 32GiB
     *-bank:4
          size: 32GiB
     *-bank:5
          size: 32GiB
     *-bank:6
          size: 32GiB
     *-bank:7
          size: 32GiB
     *-bank:8
          size: 32GiB
     *-bank:9
          size: 32GiB
     *-bank:10
          size: 32GiB
     *-bank:11
          size: 32GiB
     *-bank:12
          size: 32GiB
     *-bank:13
          size: 32GiB
     *-bank:14
          size: 32GiB
     *-bank:15
          size: 32GiB
#
```

### clock attributes

The command bellow shows how to obtain the attributes necessary for the clustconf.yaml, clock's max and min.

```
# /usr/bin/cpupower  frequency-info |grep "current policy"
  current policy: frequency should be within 3.07 GHz and 3.45 GHz.
#
```

### firmware attributes

```
# /opt/xcat/bin/rinv c699c001 firm  | awk  '{$1= ""; printf "-'\''%s'\''\n", substr($0,2)}'
-'BMC Firmware Product: ibm-v2.0-0-r46-0-gbed584c (Active)*'
-'HOST Firmware Product: IBM-witherspoon-ibm-OP9_v1.19_1.185 (Active)*'
-'HOST Firmware Product: -- additional info: buildroot-2018.02.1-6-ga8d1126'
-'HOST Firmware Product: -- additional info: capp-ucode-p9-dd2-v4'
-'HOST Firmware Product: -- additional info: hostboot-8c017b7-pa10387d'
-'HOST Firmware Product: -- additional info: hostboot-binaries-37be536'
-'HOST Firmware Product: -- additional info: linux-4.16.13-openpower1-pd73059d'
-'HOST Firmware Product: -- additional info: machine-xml-94a137f'
-'HOST Firmware Product: -- additional info: occ-f796766'
-'HOST Firmware Product: -- additional info: op-build-v2.0.5-305-g9caa6d0'
-'HOST Firmware Product: -- additional info: petitboot-v1.7.1-p22c6f3f'
-'HOST Firmware Product: -- additional info: sbe-1b143b5'
-'HOST Firmware Product: -- additional info: skiboot-v6.0.6'
#
```

### gpu attributes

Issue the command bellow to get the clustconf.yaml, GPU's pciids, device, vbios, clocks_applications_gr, clocks_applications_mem and persistence_mode.

```
# nvidia-smi --query-gpu=gpu_name,gpu_bus_id,vbios_version,clocks.applications.gr,
clocks.applications.mem,persistence_mode nvlink --format=csv
name, pci.bus_id, vbios_version, clocks.applications.graphics [MHz],
clocks.applications.memory [MHz], persistence_mode
Tesla V100-SXM2-16GB, 00000004:04:00.0, 88.00.13.00.02, 1312 MHz, 877 MHz, Enabled
Tesla V100-SXM2-16GB, 00000004:05:00.0, 88.00.13.00.02, 1312 MHz, 877 MHz, Enabled
Tesla V100-SXM2-16GB, 00000004:06:00.0, 88.00.13.00.02, 1312 MHz, 877 MHz, Enabled
Tesla V100-SXM2-16GB, 00000035:03:00.0, 88.00.13.00.02, 1312 MHz, 877 MHz, Enabled
Tesla V100-SXM2-16GB, 00000035:04:00.0, 88.00.13.00.02, 1312 MHz, 877 MHz, Enabled
Tesla V100-SXM2-16GB, 00000035:05:00.0, 88.00.13.00.02, 1312 MHz, 877 MHz, Enabled
```

The command bellow shows how to get the clustconf.yaml, GPU's link_speed.

```
# usr/bin/nvidia-smi nvlink -s|grep "GB/s" |cut -d ' ' -f 4 |sort -u
25
```

### ib attributes

The command bellow shows how to get all the IB adapters. This is the information needed for clustconf.yaml file, IB's slot_rx attribute.

```
# lspci |grep Mellanox
0003:01:00.0 Infiniband controller: Mellanox Technologies MT28800 Family [ConnectX-5 Ex]
0003:01:00.1 Infiniband controller: Mellanox Technologies MT28800 Family [ConnectX-5 Ex]
0033:01:00.0 Infiniband controller: Mellanox Technologies MT28800 Family [ConnectX-5 Ex]
0033:01:00.1 Infiniband controller: Mellanox Technologies MT28800 Family [ConnectX-5 Ex]
#
```

This is the command to get the clustconf.yaml IB's board_id and firmware attributes.

```
#  ibv_devinfo|egrep 'fw_ver|board' | sort -u
        board_id:                       IBM0000000002
        fw_ver:                         16.22.8038
#
```

### os and kernel attributes

Use the command bellow to get the value for clustconf.yaml file, OS's pretty_name.

```
#  cat /etc/os-release | grep PRETTY_NAME
PRETTY_NAME="Red Hat Enterprise Linux Server 7.5 (Maipo)"
```

And the command bellow to get the value for clustconf.yaml file, kernel's release.

```
#  uname -r
4.14.0-49.10.1.el7a.ppc64le
#
```

### nvme attributes

The output of the command bellow has the attributes for the clustconf.yaml file, NVMe's vendor and firmware_rev.

```
#  sudo /usr/sbin/nvme list -o json
{
  "Devices" : [
    {
      "DevicePath" : "/dev/nvme0n1",
      "Firmware" : "MN12MN12",
      "Index" : 0,
      "ModelNumber" : "PCIe3 1.6TB NVMe Flash Adapter II x8",
      "ProductName" : "Non-Volatile memory controller: Samsung Electronics Co
Ltd VMe SSD Controller 172Xa PCIe3 1.6TB NVMe Flash Adapter II x8",
      "SerialNumber" : "S3RVNA0J703331"
    }
  ]
}
#
```

### software attribute list

Each item of the list must have the format:

- <software> : <version>

The Diaganostic's chk-sw-level will issue the command:
*rpm -qa|grep $software | grep $version* to validate the entry.

Examples:

- ibm-csm: 1.1.1-163
- gpfs.base: 5.0.1
- lm_sensors: 3.4.0-4.20160601gitf9185e5
- datacenter: 1.4.2-1
- cuda: 9-2-9.2.148-1
- ibm_smpi: 10.02.00.04rtm0

### gpfs_mounts section

This section defines all the gpfs Filesystems required on all nodes.

***gpfs_mounts:*** # list of gpfs Filesystems. Format is:

- {mount: '< mount_point>', match: '<filesystem>' }

### gpfs_mounts example

```
gpfs_mounts:
  - {mount: '/gpfs/r92gpfs01',  match: 'r92gpfs01' }
  - {mount: '/gpfs/r92gpfs02',  match: 'r92gpfs02' }
```

Table 10 clustconf.yaml: gpfs_mounts

### rvitals section

This section defines the rvitals parameter, such as temperature, voltage, current, etc. Values defined in this group are checked in the rvitals test. rvitals test issues the command */opt/xcat/bin/rvitals <noderange>* and validates the value returned by the command with this section of the clustconf.file.

***rvitals:***

***<name>:*** # name, an identifier that will be used in the rvitals attribute in the node_i group.

- {id: 'attribute.', regex: <regex>, range: [ <v1>,<v2>.<v3>] }

### rvitals example

```
rvitals:
wspoon_dd2:
    - {id: 'Ambient', regex: (\S+), range: [10,40] }
    - {id: 'Fan1 \d',  regex: (\S+), range: [0,24000] }
    - {id: 'Fan[0-3] \d',  regex: (\S+), range: [2500,14000] }
    - {id: 'P\d Vcs Temp',  regex: (\S+), range: [15,80]}
    - {id: 'P\d Vdd Temp',  regex: (\S+), range: [15,80]}
    - {id: 'P\d Vddr Temp',  regex: (\S+), range: [15,80]}
    - {id: 'P\d Vdn Temp',  regex: (\S+), range: [15,80]}
    - {id: 'Ambient', regex: (\S+), range: [10,40] }
    - {id: 'Dimm\d+ Temp', regex: (\S+), range: [15,75,N/A]}
    - {id: 'P\d Core6 Temp',    regex: (\S+), range: [0,90,N/A]}
    - {id: 'P\d Core7 Temp',    regex: (\S+), range: [0,90,N/A]}
    - {id: 'P\d Core\d Temp',    regex: (\S+), range: [10,90,N/A]}
    - {id: 'P\d GPU Power',  regex: (\S+), range: [10,1800,N/A] }
    - {id: 'P\d Io Power',  regex: (\S+), range: [10,500,N/A] }
    - {id: 'P\d Mem Power',  regex: (\S+), range: [10,500,N/A] }
    - {id: 'P\d Power',  regex: (\S+), range: [1,1800,N/A] }
```

```
    - {id: 'Ps\d Input Power',  regex: (\S+), range: [10,1800,N/A] }
    - {id: 'Ps\d Input Voltage',  regex: (\S+), range: [200,285,N/A] }
    - {id: 'Ps1 Output Current',  regex: (\S+), range: [0,100,N/A] }
    - {id: 'Ps\d Output Voltage',  regex: (\S+), range: [10,400,0] }
    - {id: 'Ps\d Output Current',  regex: (\S+), range: [10,100,N/A] }
    - {id: 'Ps\d Output Voltage',  regex: (\S+), range: [300,400,N/A] }
    - {id: 'Storage A Power',  regex: (\S+), range: [10,500,N/A] }
    - {id: 'Storage B Power',  regex: (\S+), range: [10,500,N/A] }
    - {id: 'Total Power',  regex: (\S+), range: [10,2000,N/A] }
```

**Table 11 clustconf.yaml: rvitals**

## clustconf.yaml example

An complete example is installed by the Diagnostic rpm as /opt/ibm/csm/hcdiag/etc/clustconf.yaml.

```
#
# 6 gpus
#
  - case: (c699c0([0-5]+[0-9]))
    rvitals: wspoon_dd2
    ncpus: 176
    memory:
      total: 606
      banks: 16
      bank_size:  32
    clock:
      max: 3.50
      min: 2.0
    firmware:
      name: 1742FxRev13
      versions:
        - 'BMC Firmware Product:   ibm-v2.0-0-r46-0-gbed584c (Active)*'
        - 'HOST Firmware Product:   IBM-witherspoon-ibm-OP9_v1.19_1.192 (Active)*'
        - 'HOST Firmware Product: -- additional info: buildroot-2018.02.1-6-ga8d1126'
        - 'HOST Firmware Product: -- additional info: capp-ucode-p9-dd2-v4'
        - 'HOST Firmware Product: -- additional info: hostboot-4bd54cb-p8235b67'
        - 'HOST Firmware Product: -- additional info: hostboot-binaries-d01b024'
        - 'HOST Firmware Product: -- additional info: linux-4.16.13-openpower1-p4213cac'
        - 'HOST Firmware Product: -- additional info: machine-xml-94a137f'
        - 'HOST Firmware Product: -- additional info: occ-f796766'
        - 'HOST Firmware Product: -- additional info: petitboot-v1.7.2-p439439e'
        - 'HOST Firmware Product: -- additional info: sbe-defe254'
        - 'HOST Firmware Product: -- additional info: skiboot-v6.0.8'
  gpu:
      pciids:
        - '0004:04:00.0'
        - '0004:05:00.0'
        - '0004:06:00.0'
        - '0035:03:00.0'
        - '0035:04:00.0'
        - '0035:05:00.0'
      device: "Tesla V100-SXM2-16GB"
      vbios: "88.00.13.00.02"
      clocks_applications_gr: 1312
      clocks_applications_mem: 877
      persistence_mode: Enabled
      link_speed : 25
      nlinks     : 6
    ib:
      slot_rx: "00(03|33):01:00.(0|1)"
      board_id: "IBM0000000002"
      firmware: "16.22.8038"
```

```
    link_speed: 16
    link_width: 8

  os:
    pretty_name: "Red Hat Enterprise Linux Server 7.5 (Maipo)"
  kernel:
    release: "4.14.0-49.11.1.el7a.ppc64le"

  ufm:
    ip_address: "10.7.0.41"
    user: "admin"
    password: "123456"
#
# gpfs mounts expected on all nodes.
gpfs_mounts:
 - {mount: '/gpfs/wscgpfs01',  match: 'wscgpfs01' }
 - {mount: '/gpfs/wscgpfs02',  match: 'wscgpfs02' }

rvitals:
 wspoon_dd2:
  - {id: 'Ambient', regex: (\S+), range: [10,40] }
  - {id: 'Fan1 \d',  regex: (\S+), range: [0,24000] }
  - {id: 'Fan[0-3] \d',  regex: (\S+), range: [2500,14000] }
  - {id: 'P\d Vcs Temp',  regex: (\S+), range: [15,80]}
  - {id: 'P\d Vdd Temp',  regex: (\S+), range: [15,80]}
  - {id: 'P\d Vddr Temp',  regex: (\S+), range: [15,80]}
  - {id: 'P\d Vdn Temp',  regex: (\S+), range: [15,80]}
  - {id: 'Ambient', regex: (\S+), range: [10,40] }
  - {id: 'DIMM\d+ Temp', regex: (\S+), range: [15,75,N/A]}
  - {id: 'P\d Core6 Temp',    regex: (\S+), range: [0,90,N/A]}
  - {id: 'P\d Core7 Temp',    regex: (\S+), range: [0,90,N/A]}
  - {id: 'P\d Core\d Temp',    regex: (\S+), range: [10,90,N/A]}
  - {id: 'P\d GPU Power',  regex: (\S+), range: [10,1800,N/A] }
  - {id: 'P\d Io Power',  regex: (\S+), range: [10,500,N/A] }
  - {id: 'P\d Mem Power',  regex: (\S+), range: [10,500,N/A] }
  - {id: 'P\d Power',  regex: (\S+), range: [1,1800,N/A] }
  - {id: 'Ps\d Input Power',  regex: (\S+), range: [10,1800,N/A] }
  - {id: 'Ps\d Input Voltage',  regex: (\S+), range: [200,285,N/A] }
  - {id: 'Ps1 Output Current',  regex: (\S+), range: [0,100,N/A] }
  - {id: 'Ps\d Output Voltage',  regex: (\S+), range: [10,400,0] }
  - {id: 'Ps\d Output Current',  regex: (\S+), range: [10,100,N/A] }
  - {id: 'Ps\d Output Voltage',  regex: (\S+), range: [300,400,N/A] }
  - {id: 'Storage A Power',  regex: (\S+), range: [10,500,N/A] }
  - {id: 'Storage B Power',  regex: (\S+), range: [10,500,N/A] }
  - {id: 'Total Power',  regex: (\S+), range: [10,2000,N/A] }
```

## 5.3 Return code/Exit code

### *5.3.1 Diagnostic test module return code*

Table 12: Test return codes list the return codes that are reserved by the Diagnostic framework.

**Table 12: Test return codes**

| *RC* | *Description* | *Action* |
|---|---|---|
| **-1** | Unknown xCat error | Check the diagnostic test output and/or the diagnostic log file to find the cause of the error. |
| **99** | DB_ERROR Database error | Check the return code of the CSM APIs call.<br>Possible errors:<br>. csm daemons are not running<br>. database error<br>. sw error |
| **255** | Timeout. xCat timeout | Increase the test case timeout value in the tests.properties file. |
| **N** | Any exit code returned by a test | |

### *5.3.2 Diagnostic exit code*

**Table 13: Diagnostic exit codes**

| *Code* | *Description* | *Action* |
|---|---|---|
| **0** | Diagnostic application and all the tests completed successfully.<br>It will be logged in the database as "COMPLETED" | N/A |

| | | |
|---|---|---|
| **100** | Diagnostic application completed successfully but one or more tests failed.<br>It will be logged in the database as "COMPLETED_FAIL" | The log file/ras will have a message like:<br><timestamp> ERROR: <test_name> FAIL on node <node_name> , serial number: <serial_number>, rc= <rc>. (details in <failed_test_output_file).<br>This message also goes to the RAS subsystem and to the csm_diag_result table.<br><br>Look for the <failed_test_output_file> to find out the cause of fail. |
| **N** | Diagnostic application failed.<br>It will be logged in the database as "FAILED" or "CANCELED". | Possible cause:<br>- Diagnostic received SIGTEM, SIGUSR1, SIGUSR2.<br>- test or bucket validation error<br>- target validation error<br>- database error<br><br>Look for the Diagnostic log file to find the cause of fail. |

## 5.4 Outputs

As described previous the outputs are stored in disk. The directory structure is:

<log_dir>/<run_id>/<test_name>, where:

log_dir : defined in the hcdiag.properties file and can be overwritten by command
run_id : unique identifier of the diagnostic run.
test_name : name of the test

Example for log_dir /home/diagadmin/log
***/home/diagadmin/log/****hcdiag_run-1607261336359470.log*
***/home/diagadmin/log/1607261336359470***
***/home/diagadmin/log/1607261336359470/test-simple***
***/home/diagadmin/log/1607261336359470/test-simple/****c931f02p14-2016-07-26-13:38:54.output*
***/home/diagadmin/log/1607261336359470/test-simple/****c931f02p16-2016-07-26-13:38:54.output*
***/home/diagadmin/log/1607261336359470/test-simple/****c931f02p18-2016-07-26-13:38:54.output*
***/home/diagadmin/log/1607261336359470/test-simple/****c931f02p12-2016-07-26-13:38:54.output*
***/home/diagadmin/log/****hcdiag_run-1607261337209630.log*
***/home/diagadmin/log/1607261337209630***
***/home/diagadmin/log/1607261337209630/nvvslong***
***/home/diagadmin/log/1607261337209630/test-simple***

***/home/diagadmin/log/1607261337209630/nvvslong****/c931f02p16-2016-07-26-13:37:21.output*
***/home/diagadmin/log/1607261337209630/test-simple****/c931f02p16-2016-07-26-13:40:03.output*

### *5.4.1 Framework output*

The framework output is located under <log_dir>, all framework logs are in this location. File start with hcdiag_run_<run_id>.log

### *5.4.2 Test output*

Test output is located under <log_dir>/<run_id>/<test_name>, and has the format: <node_name>-<timestamp>.output

# 6. Appendix B

## *6.1 Tests summary table*

The Table 14: Tests summary, lists all tests integrated into Diagnostic, its type, coverage, modes, dependencies and if *sudo* privilege is required.

**Table 14: Tests summary**

| | NAME | COVERAGE | SUDO req | MGMT MODE only | INTRUSIVE | DEPENDENCIES COMMENTS |
|---|---|---|---|---|---|---|
| 1. | *6.1.1 chk-aslr* chk-aslr | Node config | | | | |
| 2. | chk-boot-time | Node | | ✓ | | xCAT |
| 3. | chk-capi | | ✓ | | | |
| 4. | chk-cpu | Node | ✓ | | | It can also run w/o sudo privilege (not optimal) |
| 5. | chk-cpu-count | Node | | | | |
| 6. | chk-csm-health | CSM | | ✓ | | CSM |
| 7. | chk-free-memory | Node | | | | |
| 8. | chk-gpfs-mount | Filesystem | | | | |
| 9. | chk-gpu-ats | Node config | | | | |
| 10. | chk-hca-attributes | Network | | | | |
| 11. | chk-ib-node | Network | | | | |
| 12. | chk-ib-pcispeed | Network | ✓ | | | |
| 13. | chk-idle-state | Node config | | | | |
| 14. | chk-kworker | Node | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 15. | chk-led | Node | | ✓ | | xCAT |
| 16. | chk-load-average | Node | | | | |
| 17. | chk-load-cpu | Node | | | | |
| 18. | chk-load-mem | Node | | | | |
| 19. | chk-memory | Node | ✓ | | | lshw package |
| 20. | chk-mlnx-pci | Network | ✓ | | ✓ | |
| 21. | chk-mlxlink-pci | Network | ✓ | | ✓ | |
| 22. | chk-nfs-mount | Filesystem | | | | |
| 23. | chk-noping | Node | | ✓ | | |
| 24. | chk-nvidia-clocks | GPU | | | | |
| 25. | chk-nvidia-smi | GPU | | | | |
| 26. | chk-nvidia-vbios | GPU | | | | |
| 27. | chk-nvlink-speed | GPU | | | | |
| 28. | chk-nvme | Storage | ✓ | | | nvme-cli package |
| 29. | chk-nvme-mount | Filesystem | | | | |
| 30. | chk-os | Node | | | | |
| 31. | chk-power | Node | | | | lm_sensors package |
| 32. | chk-process | Node | | | | |
| 33. | chk-smt | Node config | | | | |
| 34. | chk-sw-level | Node | | | | |
| 35. | chk-sys-firmware | Node | | ✓ | | xCAT |
| 36. | chk-sys-firmware-local | Node | | | | |
| 37. | chk-temp | Node | | | | lm_sensors package |
| 38. | chk-zombies | Node | | | | |
| 39. | compdiag | Network | ✓ | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 40. | daxpy | Memory | | | ✓ | IBM Spectrum MPI Does not require sudo by default. If clean caches is requested, then sudo is required. |
| 41. | **Error! Reference source not found.** | Memory | | | ✓ | IBM Spectrum MPI |
| 42. | dcgm-diag | GPU | | | ✓ | NVIDIA |
| 43. | dcgm-diag-double | GPU | | | ✓ | NVIDIA |
| 44. | dcgm-diag-single | GPU | | | ✓ | NVIDIA |
| 45. | dcgm-health | GPU | | | | NVIDIA |
| 46. | dgemm | CPU | | | ✓ | IBM Spectrum MPI |
| 47. | dgemm-gpu | GPU | ✓ | | ✓ | IBM Spectrum MPI Does require sudo by default, but can be disabled (no clock setting) |
| 48. | **Error! Reference source not found.** | CPU | | | ✓ | IBM Spectrum MPI |
| 49. | fieldiag | GPU | ✓ | | ✓ | NVIDIA |
| 50. | gpfsperf | Filesystem | | | ✓ | GPFS |
| 51. | gpudirect | GPU | | ✓ | | |
| 52. | gpu-health | GPU | | | ✓ | Cuda |
| 53. | hcatest | Network | | ✓ | | |
| 54. | hxecache | Memory | ✓ | | ✓ | HTX |
| 55. | hxecpu | Processor | ✓ | | ✓ | HTX |
| 56. | hxecpu_pass2 | Processor | ✓ | | ✓ | HTX |
| 57. | hxediag_eth | Network | ✓ | | ✓ | HTX |
| 58. | hxediag_ib | Network | ✓ | | ✓ | HTX |
| 59. | hxeewm_pass2 | Thermal | ✓ | | ✓ | HTX |
| 60. | hxefabricbus_pass2 | Fabric | ✓ | | ✓ | HTX |

| | | | | | | |
|---|---|---|---|---|---|---|
| 61. | hxefpu64 | Processor | ✓ | | ✓ | HTX |
| 62. | hxefpu64_pass2 | Processor | ✓ | | ✓ | HTX |
| 63. | hxemem64 | Memory | ✓ | | ✓ | HTX |
| 64. | hxemem64_pass2 | Memory | ✓ | | ✓ | HTX |
| 65. | hxenvidia | GPU | ✓ | | ✓ | HTX |
| 66. | hxenvidia_pass2 | GPU | ✓ | | ✓ | HTX |
| 67. | hxerng_pass2 | Processor | ✓ | | ✓ | HTX |
| 68. | hxesctu_pass2 | Memory | ✓ | | ✓ | HTX |
| 69. | hxestorage_nvme | Storage | ✓ | | ✓ | HTX |
| 70. | hxestorage_nvme_pass2 | Storage | ✓ | | ✓ | HTX |
| 71. | hxestorage_sd | Storage | ✓ | | ✓ | HTX |
| 72. | hxestorage_sd_pass2 | Storage | ✓ | | ✓ | HTX |
| 73. | ibcredit | Network | ✓ | | | |
| 74. | ibverbs | Network | | ✓ | | |
| 75. | ipoib | Network | | | | |
| 76. | jlink | Network | | | | IBM Spectrum MPI (does not run as root) |
| 77. | lnklwls | Network | ✓ | | | |
| 78. | lnkqual | Network | ✓ | | | |
| 79. | mmhealth | Filesystem | ✓ | | | GPFS |
| 80. | nsdperf | Network | | ✓ | | GPFS |
| 81. | nvvs | GPU | | | ✓ | NVIDIA |
| 82. | p2pBandwidthLatencyTest | GPU | | | | NVIDIA |
| 83. | ppping | Network | | ✓ | | xCAT |
| 84. | rpower | Node | | ✓ | | xCAT |

| | | | | |
|---|---|---|---|---|
| 85. | rvitals | Node | ✓ | xCAT |
| 86. | swhealth | Network | | |
| 87. | test-simple | Diagnostic infrastructure | | |

## *6.2 Tests man pages*

This section will describe all the tests integrated into Diagnostic in man page style.

- INVOCATION section
  In most of the tests, *"—target <noderange>"* is shown as optional. But what it really means is that when running in Node Mode, *"—target <noderange>"* is not required. It is mandatory when running in Management Mode.

- CONFIGURATION FILE section
  all the files used as an input for the test are listed in this section. Some of the files require changes. The command to obtain the input for the modification is added as a comment (#), next to the line, if applicable.

### *6.2.1 chk-aslr*

NAME

chk-aslr - *check if the node Address Space Layout Randomization(ASLR) is enable or disabled*.

SYNOPSIS

chk-aslr.sh [value]

INVOCATION

hcdiag_run.py –test chk-aslr [--target <noderange>]

DESCRIPTION

Checks if ALSR is set to the value passed as the argument. It compares the value with the content of /proc/sys/kernel/randomize_va_space file.

It can run concurrently with user jobs.

ARGUMENTS

The following values are supported:

- 0 – No randomization. Everything is static.
- 1 – Conservative randomization.
- 2 – Full randomization.

Default is 0 (disabled), return PASS if disabled, FAIL if enabled.

RETURNS

Returns PASS if the ASLR is set to the value passed as argument.

CONFIGURATION FILE

/opt/ibm/csm/hcdiag/etc/test.properties

```
[tests.chk-aslr]
description = check if the node aslr is enable/disabled
executable  = /opt/ibm/csm/hcdiag/tests/chk-aslr/chk-aslr.sh
#args= 1
```

### 6.2.2 *chk-boot-time*

```
NAME
    chk-boot-time - check if the nodes were boot successfully


SYNOPSIS
    chk-boot-time.sh [-v] [-h] <noderange>


INVOCATION
    hcdiag_run.py -test chk-boot-time -target <noderange>



DESCRIPTION
     This test checks if the nodes in the noderange were booted
     successfully and the xcat status are in sync.
     It runs on the management node, and uses XCAT commands
     nodestat and lsdef.

     It can run concurrently with user jobs.

     NOTE: Run in Management mode only.
           User running this test has to have xCAT privileges.
           The xCAT-client package is required



ARGUMENTS
     -v         Set the verbose mode.

     -h         Display the help message.

     noderange  One of more nodes, in XCAT noderange format.


RETURNS
     Returns PASS if all nodes do not show inconsistencies
     between nodestat and lsdef status.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
       [tests.chk-boot-time]
       description = checks if the nodes were boot successfully
       executable  = /opt/ibm/csm/hcdiag/tests/chk-boot-time/chk-
boot-time.sh
       targetType  = Management
       xcat_cmd    = yes
       timeout     = 10
```

### *6.2.3 chk-capi*

```
NAME
    chk-capi - check if the CAPI is enable on the node.


SYNOPSIS
    chk-capi.sh -d <device> | -a [-l] [-v] [-V] [-h]


INVOCATION
    hcdiag_run.py –test chk-capi –target <noderange>



DESCRIPTION
     This test checks if CAPI (Coherent Accelerator Processor Interface) is
     enable on the node.

     It can run concurrently with user jobs.

     NOTE: User running this test has to have sudo privilege.



ARGUMENTS
     -a         Run on all Mellanox devices in system
     -d         PCI device id
     -l         List Mellanox adapters in system
     -v         Additional output for debug
     -V         Display script version
     -h         Display the help message.



RETURNS
     Returns PASS if CAPI is enabled.



CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
       [tests.chk-capi]
       description = checks if the CAPI is enabled on the node
       executable  = /opt/ibm/csm/hcdiag/tests/chk-capi/chk-capi.sh
       args        = -a
       timeour     = 10
```

### *6.2.4 chk-cpu*

```
NAME
    chk-cpu – check cpu related information


SYNOPSIS
    chk-cpu.sh [-v] [-h]


INVOCATION
    hcdiag_run.py –test chk-cpu [–target <noderange>]


DESCRIPTION
     This test checks cpu attributes. Some of the steps looks at
     clustconf.yaml file for validation.
        • number of CPUs (lscpu)
        • if all cores present are online (ppc64_cpu)
        • cpu frequency (ppc64_cpu)

     Also, for information it prints the following information:
        • scaling_governor, scaling_max_freq, scaling_min_freq (sysfs)
        • sub cores per-core and dscr (Data Streaming Control Register) -
          ppc64_cpu.
          The Data Stream Control Register of POWER processors is used to
          control the degree of aggressiveness of memory prefetching for load
          and store. Possible values are 0: medium (default) and 1: off.
        • numactrl –hardware command output

      I can run concurrently with user jobs.

      NOTE: if user running this test has sudo privilege, it does not display
      the DSCR information and it checks the CPU clock from /proc/cpuinfo,
      instead of issuing ppc64_cpu –frequency.


ARGUMENTS
     -v   Set the verbose mode.
     -h   Display the help message.

RETURNS
     Returns PASS if the number of cpus and cpu frequency matches
     what is defined in the clustconf.yaml and if the number of cores
     present is the same as the number of cores online.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
       [tests.chk-cpu]
       description = show and check cpu information
       executable  = /opt/ibm/csm/hcdiag/tests/chk-cpu/chk-cpu.sh
       timeout     = 10

    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
         # grep processor /proc/cpuinfo |wc -l
```

```
        ncpus: 176

        # cpupower frequency-info
        clock:
           max: 3.8
           min: 2.0
```

### 6.2.5 *chk-cpu-count*

```
NAME
    chk-cpu-count – check number of cpus.


SYNOPSIS
    chk-cpu-count.sh [-v] [-h]


INVOCATION
    hcdiag_run.py -test chk-cpu-count [-target <noderange> ]


DESCRIPTION
     This test checks if the number of cpus (lscpu) matches the definition
     in the clustconf.yaml file.

     I can run concurrently with user jobs.


ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.

RETURNS
     Returns PASS if the number of cpus obtained via lscpu command matches
     what is defined in the clustconf.yaml.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
       [tests.chk-cpu-count]
       description = check number of cpus on the node
       executable  = /opt/ibm/csm/hcdiag/tests/chk-cpu-count/chk-cpu-count.sh
       timeout     = 10

    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
         # grep processor /proc/cpuinfo |wc -l
         ncpus: 176
```

### 6.2.6 *chk-csm-health*

```
NAME
    chk-csm-health – csm infrastructure health check


SYNOPSIS
    chk-csm-health.sh


INVOCATION
    hcdiag_run.py –test chk-csm-health –target <mgmt|utility|aggregator>



DESCRIPTION
     It contacts the CSM daemons and check its status. It validates
     the topology of the nodes/csm daemons.
     It invokes the /opt/ibm/csm/bin/csm_infrastructure_health_check CSM
command.

     I can run concurrently with user jobs, with caution.

     NOTE: Run in Management mode only.
           This test does contact ALL reachable daemon regardless of running
jobs or status.
           The ibm-csm-api package is required


ARGUMENTS
     None


RETURNS
     Returns PASS if the execution of the command
     /opt/ibm/csm/bin/csm_infrastructure_health_check was successful.
     Unresponsive daemons in the output does not cause the test to FAIL.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-csm-health]
      description = csm infrastructure health check
      executable  = /opt/ibm/csm/hcdiag/tests/chk-csm-health/chk-csm-
      health.sh
      timeout     = 100
      targetType  = Management
```

### *6.2.7 chk-free-memory*

```
NAME
    chk-free-memory - check if the amount of free memory is above the
    threshold.


SYNOPSIS
    chk-free-memory.sh [threshold]


INVOCATION
    hcdiag_run.py -test chk-free-memory [-target <noderange>



DESCRIPTION
     Check if the node total amount of free memory is bellow or equal the
     threshold. The threshold is in percentage relative to the total memory.

     It can run concurrently with user jobs.


ARGUMENTS
     threshold
         minimal amount of free memory, in percentage relative to the
         total  memory.  The default is 10%.


RETURNS
     Returns PASS if the amount of free memory is above the threshold
     of the Node's total amount of memory.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-free-memory]
      description = check if the amount of free memory is below or equal the
      threshold.
      executable  = /opt/ibm/csm/hcdiag/tests/chk-free-memory/chk-free-
      memory.sh
      timeout     = 10
```

### 6.2.8 *chk-gpfs-mount*

```
NAME
    chk-gpfs-mount - check gpfs Filesystem/mount point


SYNOPSIS
    chk-gpfs-mount.sh [-v] [-h]


INVOCATION
    hcdiag_run.py -test chk-gpfs-mount [-target <noderange>]



DESCRIPTION
     Checks if the gpfs Filesystem(s) is (are) mounted on the node.
     It reads the information from the clustconf.yaml file.

     It can run concurrently with user jobs.


ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS the gpfs Filesystem is (are) mounted correctly.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-gpfs-mount]
      description = check gpfs Filesystem/mount point
      executable  = /opt/ibm/csm/hcdiag/tests/chk-gpfs-mount/chk-gpfs-
      mount.sh
      timeout     = 10

     /opt/ibm/csm/hcdiag/etc/clustconf.yaml
       # gpfs mounts expected on all nodes.
        gpfs_mounts:
           - {mount: '/gpfs/r92gpfs01',  match: 'r92gpfs01' }
           - {mount: '/gpfs/r92gpfs02',  match: 'r92gpfs02' }
```

### *6.2.9 chk-gpu-ats*

```
NAME
    chk-gpu-ats - check if GPU Address Translation Service (ATS)
    is enabled or disabled.


SYNOPSIS
    chk-gpu-ats.sh [value]

INVOCATION
    hcdiag_run.py -test chk-gpu-ats [-target <noderange>]


DESCRIPTION
     Checks if the  GPU ATS is enabled or not by looking at the content
     of /sys/module/nvidia_uvm/parameters/nvm8_ats_mode file.
     The GPU ATS supports allow the GPU to access the CPU's page tables
     directly.

     It can run concurrently with user jobs.



ARGUMENTS
     value
         0: ATS disabled
         1: ATS enabled
          Default value is 1, return PASS if enabled, FAIL if disabled


RETURNS
     Returns PASS if the ASLR is set to the value passed as argument.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
       [tests.chk-gpu-ats]
       description = checks if the GPU ATS is enabled or not.
       executable  = /opt/ibm/csm/hcdiag/tests/chk-ats/chk-ats.sh
       #args     = 1
```

### *6.2.10 chk-hca-attributes*

```
NAME
    chk-hca-atributes - check hca board version and firmware


SYNOPSIS
    chk-hca-attributes.sh [-v] [-h]


INVOCATION
    hcdiag_run.py -test chk-hca-attributes [-target <noderange>]



DESCRIPTION
     Uses ibv_devinfo command to find the board_id and firmware of the IB
     adapter and validates with the definition in clustconf.yaml file.

     I can run concurrently with user jobs.

     NOTE: The libibverbs-utils-41mlnx1-OFED package is required.

ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS if board version and firmware of the IB adapter on the node
     matches the clustconf.yaml definition.



CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-hca-attributes]
      description = check hca board version and firmware
      executable  = /opt/ibm/csm/hcdiag/tests/chk-hca-attributes/chk-hca-
                    attributes.sh
      timeout     = 10


    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
       #   ibv_devinfo | egrep 'board_id|fw'
       ib:
            board_id: "IBM0000000002"
            firmware: "16.21.0106"
```

### *6.2.11 chk-ib-node*

```
NAME
    chk-ib-node - check node IB bandwidth


SYNOPSIS
   chk-ib-node.sh [-c0 cpu] [-c1 cpu] [-m0 mlx_port] [-m1 mlx_port] [-b] [-h]


INVOCATION
    hcdiag_run.py -test chk-ib-node [-target <noderange>]


DESCRIPTION
     Checks the IB bandwidth on a node, loop back traffic.
     It compares the bandwidth with the given threshold.

     NOTE: The perftest package is required.


ARGUMENTS
     -c0   cpu for the first process. Default 84
     -c1   cpu for the second process. Default 172
     -m1   Mellanox port for the first process. Default mlx5_0
     -m1   Mellanox port for the second process. Default mlx5_3
     -b    minimal bandwidth expected on the node. Default is 21250
     -h    display help message


RETURNS
     Returns PASS if the measured bandwidth in loop back mode is greater
     than the expected minimal bandwidth.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-ib-node]
      description = check node IB bandwidth
      executable  = /opt/ibm/csm/hcdiag/tests/chk-ib-node/chk-ib-node.sh
      timeout     = 50
      #args       =
```

### *6.2.12 chk-ib-pcispeed*

```
NAME
    chk-ib-pcispeed – check IB adapters vendor, speed and width


SYNOPSIS
    chk-ib-pcispeed.sh [speed [width [vendor]]]


INVOCATION
    hcdiag_run.py –test chk-ib-pcispeed [–target <noderange>]


DESCRIPTION
     Checks the IB adapters vendor, speed and width.

     I can run concurrently with user jobs.

     NOTE: sudo privilege is required.


ARGUMENTS
     speed    adapter speed. Default is 16GT/s.
     width    adapter width. Default is x8.
     vendor   adapter vendor. Default is  Mellanox.


RETURNS
     Returns PASS if node's IB adapter vendor, width and speed matches the
     Arguments.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-ib-pcispeed]
      description = checkIBadapters vendor, speed and width
      executable  = /opt/ibm/csm/hcdiag/tests/chk-ib-pcispeed/chk-ib-
                    pcispeed.sh
      timeout     = 10
      #args       = 16GT/s x8 Mellanox
```

### *6.2.13 chk-idle-state*

```
NAME
    chk-idle-state – check node idle state configuration

SYNOPSIS
    chk-idle-state.sh


INVOCATION
    hcdiag_run.py –test chk-idle-state [–target <noderange>]


DESCRIPTION
     Check if the node idle configuration is set according to the P9
     for HPC recommendation. The effect of this setting is to hide the stop0
     and stop0_lite states. This procedure is done by installing an
     appropriate firmware image and by making a one-time-only nvram
     setting.

     The test ssues the command cpupower idle_info to check idle state
     configuration. We expect exactly three states and none of them DISABLED.
     Expected configuration:

            CPUidle driver: powernv_idle
            CPUidle governor: menu
            analyzing CPU 0:

            Number of idle states: 3
            Available idle states: snooze stop1 stop2
            snooze:
            Flags/Description: snooze
            Latency: 0
            Usage: 364186
            Duration: 16681534
            stop1:
            Flags/Description: stop1
            Latency: 5
            Usage: 86332
            Duration: 328050509
            stop2:
            Flags/Description: stop2
            Latency: 10
            Usage: 636557
            Duration: 94165984643

       It can run concurrently with user jobs.

       NOTE: This test apply only to P9 machines.

ARGUMENTS
     None


RETURNS
     Returns PASS if the result of the command 'cpupower idle_info'
     matches the expected configuration described above.
```

```
CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-idle-state]
      description = check node idle state configuration
      executable  = /opt/ibm/csm/hcdiag/tests/chk-idle-state/chk-idle-
                    state.sh
      timeout     = 10
```

### *6.2.14 chk-kworker*

```
NAME
    chk-kworker - check kworker process CPU usage


SYNOPSIS
    chk-kworker.sh [threshold] [-v] [-h]


INVOCATION
    hcdiag_run.py -test chk-kworker [-target <noderange>]



DESCRIPTION
     Uses the ps command looking for "kworker" process consuming more
     CPU then the threshold.

     It can run concurrently with user jobs.


ARGUMENTS
     threshold
          Value in percentage to be used as the CPU usage threshold.
          Default is  20%.

     -v    Set the verbose mode.

     -h    Display the help message.


RETURNS
     Returns PASS if no kworker process consuming more CPU than the threshold
     specified.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-kworker]
      description = check kworker process CPU usage
      executable  = /opt/ibm/csm/hcdiag/tests/chk-kworker/chk-kworker.sh
      #args        = 20
```

### *6.2.15 chk-led*

```
NAME
    chk-led - check for nodes with faulty LED.


SYNOPSIS
    chk-led.sh [-v] [-h] <noderange>


INVOCATION
    hcdiag_run.py -test chk-led -target <noderange>



DESCRIPTION
     Uses the command XCAT command rvitals leds to get node(s) with faulty
     LED.

     It can run concurrently with user jobs.

      NOTE: Run in Management mode only.
            User running this test has to have xCAT privileges.
            The xCAT-client package is required



ARGUMENTS
      noderange     One of more nodes, in XCAT noderange format.

      -v            Set the verbose mode.

      -h            Display the help message.



RETURNS
     Returns PASS if node does not have a faulty LED on.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-led]
      description = check for fault led in the node range
      executable  = /opt/ibm/csm/hcdiag/tests/chk-led/chk-led.sh
      targetType  = Management
      timeout     = 10
```

### *6.2.16 chk-load-average*

```
NAME
    chk-load – check if the node load average.


SYNOPSIS
    chk-load.sh [unit] [threshold] [total_process]


INVOCATION
    hcdiag_run.py –test chk-load-average [–target <noderange>]


DESCRIPTION
     Uses the command uptime to get the detect average load in the last
     1 minute, and compares it with the threshold parameter.
     Default threshold is 40%

          System load averages is the average number of processes that are either
          in a runnable or uninterruptable state.  A process  in a  runnable  state
          is either using the CPU or waiting to use the CPU.  A process in
          uninterruptable state is waiting for some I/O access, eg waiting for disk.
          The averages are taken over the three time intervals.  Load averages are
          not normalized  for the  number of CPUs in a system, so a load average of 1
          means a single CPU system is loaded all the time while on a 4 CPU system it
          means it was idle 75% of the time.


     I can run concurrently with user jobs.


ARGUMENTS
     threshold  The threshold for the load average in the last 1 min.


RETURNS
     Returns PASS if the load average is below the threshold.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-load-average]
      description = check if the node load average is below the threshold
      executable  = /opt/ibm/csm/hcdiag/tests/chk-load/chk-load.sh
      args        = average 40
      timeout     = 10
```

### *6.2.17 chk-load-cpu*

```
NAME
    chk-load-cpu - check CPU usage of one of more process(es)


SYNOPSIS
    chk-load.sh [unit] [threshold] [#process]


INVOCATION
    hcdiag_run.py -test chk-load-cpu [-target <noderange>]



DESCRIPTION
     Uses the command ps to get the process(es) using the most cpu.

     I can run concurrently with user jobs.


ARGUMENTS
     unit
          cpu is set for this test.

     threshold
          The threshold for the cpu usage. Default is 90.

     #process
           How many process(es) will be accountable for the CPU usage.
           Default is 1.


RETURNS
     Returns PASS if the cpu usage of the process(es) is below
     the threshold.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-load-cpu]
      description = check if the node cpu load below the threshold
      executable  = /opt/ibm/csm/hcdiag/tests/chk-load/chk-load.sh
      timeout     = 10
      #args        = cpu 90 1
```

### *6.2.18 chk-load-mem*

```
NAME
    chk-load-mem - check memory usage of one of more process(es).


SYNOPSIS
    chk-load.sh [unit] [threshold] [#process]


INVOCATION
    hcdiag_run.py -test chk-load-mem [-target <noderange>]



DESCRIPTION
     Uses the command ps to get the process(es) using the most memory.

     I can run concurrently with user jobs.


ARGUMENTS
     unit
         mem is set for this test
     threshold
         The threshold for the memory usage. Default is 90.
     #process
         how many process(es) will be accountable for the memory usage.
         Default is 1.


RETURNS
     Returns PASS if the process(es) memory usage is below the threshold.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-load-mem]
      description = check if the node memory load below the threshold
      executable  = /opt/ibm/csm/hcdiag/tests/chk-load/chk-load.sh
      args        = mem 40
      timeout     = 10
```

### *6.2.19 chk-memory*

```
NAME
    chk-memory - check the node total memory, number of banks and bank size.


SYNOPSIS
    chk-memory.sh [-v] [h]


INVOCATION
    hcdiag_run.py -test chk-memory [-target <noderange>]


DESCRIPTION
     Uses the lshw command to get the information about the total memory and
     DIMMS, and checks against the clustconf.yaml file definition.

     I can run concurrently with user jobs.

     NOTE: sudo privilege is required.
           lshw package is required.


ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS if the total memory and DIMMS information matches the
     definition in the clustconf.yaml file.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-memory]
      description = check node total memory, number of banks and bank size
      executable  = /opt/ibm/csm/hcdiag/tests/chk-memory/chk-memory.sh
      timeout     = 10


    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
      # sudo lshw -quiet -class memory | egrep 'total|size|bank'
      memory:
            total: 572
            banks: 16
            bank_size:  32
```

### *6.2.20 chk-mlnx-pci*

```
NAME
    chk-mlnx-pci - check if the Mellanox adapters are configured correctly.


SYNOPSIS
    chk-mlnx-pci.sh


INVOCATION
    hcdiag_run.py -test chk-mlnx-pci [-target <noderange>]



DESCRIPTION
     Check if all the Mellanox pci adapters are found and they are
     configured correctly.


     NOTE:  sudo privilege is required.
            Run on Witherspoon only.


ARGUMENTS
     None


RETURNS
     Returns PASS if the Mellanox total number of adapters are found,
     if it is GEN4 and pass the pci link check.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-mlnx-pci]
      description = check mellanox pci configuration
      executable  = /opt/ibm/csm/hcdiag/tests/chk-mlnx-pci/chk-mlnx-pci.sh
      timeout     = 10
```

### *6.2.21 chk-mlxlink-pci*

```
NAME
    chk-mlxlink-pci - check Mellanox adapters link errors.


SYNOPSIS
    chk-mlxlink-pci.sh [threshold]


INVOCATION
    hcdiag_run.py -test chk-mlxlink-pci [-target <noderange>]



DESCRIPTION
     Check if the Mellanox link errors bellow are above the threshold.
        • RX Errors
        • TX Errors
        • CRC Error dllp
        • CRC Error tlp


     NOTE: sudo privilege is required.


ARGUMENTS
     threshold    Error threshold. Default 100.


RETURNS
     Returns PASS none of the link exceeds the error threshold.


CONFIGURATION FILE

    /opt/ibm/csm/hcdiag/etc/test.properties
      tests.chk-mlxlink-pci]
      description = check Mellanox link errors
      executable  = /opt/ibm/csm/hcdiag/tests/chk-mlxlink-pci/chk-mlxlink-
      pci.sh
      #args       = 100
```

### *6.2.22 chk-nfs-mount*

```
NAME
    chk-nfs-mount - check if the NFS Filesystem is mounted correctly.


SYNOPSIS
    chk-nfs-mount.sh <filesystem type [filesystem type] >


INVOCATION
    hcdiag_run.py -test chk-nfs-mount [ -target <noderange> ]



DESCRIPTION
     Check if the NFS Filesystem is mounted correct and responding to ls.



ARGUMENTS
     filesystem    mount point of the filesystem
     type          type of the filesystem. Example: nfs, nfs4


RETURNS
     Returns PASS all the Filesystem is mounted and responding to ls.


CONFIGURATION FILE

    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-nfs-mount]
      description = check if the Filesystem is mounted correctly
      executable  = /opt/ibm/csm/hcdiag/tests/chk-nfs-mount/chk-nfs-mount.sh
      timeout     = 10
      args        = /home nfs4 /shared nfs
```

### *6.2.23 chk-noping*

```
NAME
    chk-noping - checks if the nodes that are booted are pinging

SYNOPSIS
    chk-noping.sh [-v] [h] [noderange]


INVOCATION
    hcdiag_run.py -test chk-noping -target <noderange>



DESCRIPTION
     It issues the command nodestat and lsdef to check if nodes
     that are booted are pinging.

     It can run concurrently with user jobs.

     NOTE: Run in Management mode only.
           User running this test has to have xCAT privileges.
           The xCAT-client package is required


ARGUMENTS
     noderange     One of more nodes, in XCAT noderange format.

     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS all nodes are booted/sshd and pinging.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-noping]
      description = check if the nodes that are booted are pinging
      executable  = /opt/ibm/csm/hcdiag/tests/chk-noping/chk-noping.sh
      targetType  = Management
      xcat_cmd    = yes
      timeout     = 100
      #args        = -v
```

### *6.2.24 chk-nvidia-clocks*

```
NAME
    chk-nvidia-clocks - check GPU application clocks.


SYNOPSIS
    chk-nvidia-clocks.sh [-v] [h]


INVOCATION
    hcdiag_run.py -test chk-nvidia-clocks [-target <noderange>]


DESCRIPTION
    Checks if the NVIDIA GPU Application Clocks (Graphics and Memory)
    matches the definition in clustconf.yaml file.
    It also checks if Persistence mode is Enabled.

    I can run concurrently with user jobs.


ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS if the values of the GPU application clocks
     matches the clustconf.yaml values.

CONFIGURATION FILE

    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-nvidia-clocks]
      description = check nvidia clocks
      executable  = /opt/ibm/csm/hcdiag/tests/chk-nvidia-clocks/chk-nvidia-
                    clocks.sh
      timeout     = 10


    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
      # /usr/bin/nvidia-smi --query-gpu=clocks.applications.gr,clocks.
      applications.mem,persistence_mode --format=csv
      gpu:
            clocks_applications_gr: 1530
            clocks_applications_mem: 877
            persistence_mode: Enabled
```

### *6.2.25 chk-nvidia-smi*

```
NAME
    chk-nvidia-smi – check possible stuck nvidia-smi.


SYNOPSIS
    chk-nvidia-smi.sh [-v] [h]


INVOCATION
    hcdiag_run.py –test chk-nvidia-smi [–target <noderange>]


DESCRIPTION
     Check if there are possible stuck nvidia-smi process. Process
     that is not in “R” (Runnable (or run queue)or “S” (Sleeping)state.
     I can run concurrently with user jobs.


ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS there is no possible stuck nvidia-smi process.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      tests.chk-nvidia-smi]
      description = check possible stuck nvidia-smi
      executable  = /opt/ibm/csm/hcdiag/tests/chk-nvidia-smi/
                    chk-nvidia-smi.sh
      timeout     = 10
```

### *6.2.26 chk-nvidia-vbios*

```
NAME
    chk-nvidia-vbios - check the GPUs vbios.


SYNOPSIS
    chk-nvidia-vbios.sh [-v] [h]


INVOCATION
    hcdiag_run.py -test chk-nvidia-vbios [-target <noderange>]



DESCRIPTION
     Checks if all the GPUs vbios marches what is defined in
     clustconf.yaml. It uses nvidia-smi command.

     It can run concurrently with user jobs.


ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS if all the GPUs vbios matches the definition
     in the clustconf.yaml file.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
       [tests.chk-nvidia-vbios]
       description = check nvidia vbios
       executable  = /opt/ibm/csm/hcdiag/tests/chk-nvidia-vbios/
                     chk-nvidia-vbios.sh
       timeout     = 10

    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
        #  nvidia-smi --query-gpu=gpu_name,gpu_bus_id,vbios_version
        #  --format=noheader,csv
        gpu:
           device: "Tesla V100-SXM2-16GB"
           vbios: "88.00.13.00.02"
```

### *6.2.27 chk-nvlink-speed*

```
NAME
    chk-nvlink-speed - check the nvlink status and speed.


SYNOPSIS
    chk-nvidia-speed.sh [-v] [h]


INVOCATION
    hcdiag_run.py -test chk-nvidia-speed  [ -target <noderange> ]



DESCRIPTION
     Checks the link state (active/inactive) and its link speed.

     It can run concurrently with user jobs.


ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS all the links are enabled, and the speed matches
     the clustconf.yaml definition.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-nvlink-speed]
      description = check nvlink speed
      executable  = /opt/ibm/csm/hcdiag/tests/chk-nvlink-speed/chk-nvlink-
      speed.sh
      timeout     = 10

    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
      # nvidia-smi nvlink -s
      gpu:
         link_speed : 25
         nlinks     : 6
```

### *6.2.28 chk-nvme*

```
NAME
    chk-nvme - check firmware revision and vendor of NVMe devices


SYNOPSIS
    chk-nvme.sh [-v|--verbose] [-h|--help] [-n|--no-json]


INVOCATION
    hcdiag_run.py -test chk-nvme [-target <noderange>]


DESCRIPTION
     If perl-JSON is installed, the test uses the nvme list command to
     find the firmware revision and vendor of NVMe devices on the node.
     Otherwise, it uses the lspci command in combination with sysfs.
     In both cases, it validates the revision and vendor with definitions
     in the clustconf.yaml file.

     It can run concurrently with user jobs.


ARGUMENTS
     -v|--verbose  Set the verbose mode.

     -h|--help     Display the help message.

     -n|--no-json  perl-JSON not installed on node.

     Default is use json, nvme-cli package is required, sudo privilege
     is required.


RETURNS
     Returns PASS if firmware revision and the vendor of the NVMe
     device on the node matches the clustconf.yaml definition.



CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-nvme]
      description = check nvme device vendor and firmware
      executable  = /opt/ibm/csm/hcdiag/tests/chk-nvme/chk-nvme.sh
      timeout     = 10

    clustonf.yaml
         # sudo nvme list -o json  or
         # lspci | grep 'Non-Volatile memory' and
         # cat /sys/devices/pci$bus*/*/*/nvme/nvme0/firmware_rev
         nvme:
            vendor: "Samsung"
            firmware_rev: "MN12MN12
```

### *6.2.29 chk-nvme-mount*

```
NAME
    Check-nvme-mount – check if NVMe device is mounted


SYNOPSIS
    chk-nvme-mount.sh [-v] [h] {mount_point]


INVOCATION
    hcdiag_run.py –test chk-nvme-mount [ –target <noderange> ]


DESCRIPTION
     Checks if the NVMe device is mounted correctly.
     I can run concurrently with user jobs.


ARGUMENTS
     -v           Set the verbose mode.

     -h           Display the help message.

     mount_point  NVMe device mount point. Default is /nvme.


RETURNS
     Returns PASS if the NVMe device is mounted correctly.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-nvme-mount]
      description = check if NVMe device is mounted
      executable  = /opt/ibm/csm/hcdiag/tests/chk-nvme-mount/chk-nvme-
      mount.sh
      timeout     = 10
      #args        = /nvme
```

### *6.2.30 chk-os*

```
NAME
    chk-os - check the node operating system and kernel version


SYNOPSIS
    chk-os.sh [-v|--verbose] [-h|--help]

INVOCATION
    hcdiag_run.py -test chk-os [-target <noderange>]


DESCRIPTION
      Checks the /etc/os-release file and uses the command uname -r to
      validate the operating system and kernel version with definitions in
      the clustconf.yaml file.

      It can run concurrently with user jobs.



ARGUMENTS
     -v|--verbose  Set the verbose mode.

     -h|--help     Display the help message.



RETURNS
     Returns PASS if the node operating system and kernel version
     matches the definition in clustconf.yaml.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-os]
      description = check node operating system and kernel version
      executable  = /opt/ibm/csm/hcdiag/tests/chk-os/chk-os.sh
      timeout     = 10

    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
        # cat /etc/os-release | grep PRETTY_NAME and uname -r
        os:
            name: "Red Hat Enterprise Linux Server"
            pretty_name: "Red Hat Enterprise Linux Server 7.5 (Maipo)"
        kernel:
            release: "4.14.0-49.8.1.el7a.ppc64le"
```

### *6.2.31 chk-power*

```
NAME
    chk-power - check current power capping/shifting values on node


SYNOPSIS
    chk-power.sh [-v] [-h]


INVOCATION
    hcdiag_run.py -test chk-power [-target <noderange>]


DESCRIPTION
     This test checks set power capping and shifting values on a node to
compare to current power information.

     Also, for information it prints the following:
        • configured power cap (from /sys/firmware/opal/powercap/system-
          powercap/powercap-current file)
        • cpu to gpu power shift ratios (from
          /sys/firmware/opal/psr/cpu_to_gpuX file)
        • current system power, chip power values (sensors command)
        • gpu power values (nvidia-smi command)

      It can run concurrently with user jobs.


ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS if there are no errors obtaining power information.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
       [tests.chk-power]
       description = check current power capping/shifting values on node
       executable  = /opt/ibm/csm/hcdiag/tests/chk-power/chk-power.sh
       timeout     = 10
```

### *6.2.32 chk-process*

```
NAME
    chk-process - check if a list of processes is running


SYNOPSIS
    chk-process.sh [process[ process]]


INVOCATION
    hcdiag_run.py -test chk-process [-target <noderange>]


DESCRIPTION
     It uses the ps command to check if the processes passed as
     argument are running on the node.

     I can run concurrently with user jobs.


ARGUMENTS
     process     One or a list of process.

     Default is: csmd nvidia-persistenced nv-hostengine opal-prd automount


RETURNS
     Returns PASS if all processes are running.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-process]
      description = check if a list of process are running on the node
      executable  = /opt/ibm/csm/hcdiag/tests/chk-process/chk-process.sh
      timeout     = 10
      #args = csmd nvidia-persistenced nv-hostengine opal-prd automount
```

### *6.2.33 chk-smt*

```
NAME
    chk-smt - check the node SMT (simultaneous multi-threading) state


SYNOPSIS
    chk-smt.sh [smt]


INVOCATION
    hcdiag_run.py -test chk-smt [-target <noderange>]


DESCRIPTION
     Uses the command ppc64_cpu to obtain the current SMT sate of the
     node and compares what is passed as argument.

     I can run concurrently with user jobs.


ARGUMENTS
     smt   SMT in numeric form. Default is 4.


RETURNS
     Returns PASS if the SMT state of the node matches what is passed
     as argument.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-smt]
      description = check the node SMT
      executable  = /opt/ibm/csm/hcdiag/tests/chk-smt/chk-smt.sh
      #args        = 4
```

### *6.2.34 chk-sw-level*

```
NAME
    chk-sw-level - check the node software levels


SYNOPSIS
    chk-sw-level.sh [-v|--verbose] [-h|--help]


INVOCATION
    hcdiag_run.py -test chk-sw-level [-target <noderange>]


DESCRIPTION
     Uses the rpm command to check if the software listed in
     the clustconf.yaml file, matches the version installed on the node.

     It can run concurrently with user jobs.


ARGUMENTS
     -v|--verbose  Set the verbose mode.

     -h|--help     Display the help message.


RETURNS
     Returns PASS if the list of software specified in the
     clustconf.yaml file matches what is installed in the node.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-sw-level]
      description = check the node software stack levels
      executable  = /opt/ibm/csm/hcdiag/tests/chk-sw-level/chk-sw-level.sh
      timeout     = 10

    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
         software:
              - ibm-csm: 1.1.1-163
              - gpfs.base: 5.0.1
              - lm_sensors: 3.4.0-4.20160601gitf9185e5
              - datacenter: 1.4.2-1
              - cuda: 9-2-9.2.148-1
              - ibm_smpi: 10.02.00.04rtm0
```

### *6.2.35 chk-sys-firmware*

```
NAME
    chk-sys-firmware – check the system firmware of the node


SYNOPSIS
    chk-sys-firmware.sh [-v|--verbose] [-h|--help] <noderange>


INVOCATION
    hcdiag_run.py –test chk-sys-firmare –target <noderange>


DESCRIPTION
     It uses XCAT rinv to get the node system firmware levels and compares
     with the definition in the clustconf.yaml file.

     I can run concurrently with user jobs.

     See also chk-sys-firmware-local.

     NOTE: This test only runs in Management mode.
           User running this test has to have xCAT privileges.
           The xCAT-client package is required.



ARGUMENTS
     -v|--verbose  Set the verbose mode.

     -h|--help     Display the help message.

     noderange     One of more nodes, in XCAT noderange format.


RETURNS
     Returns PASS if all nodes firmware level matches what is
     specified in the clustconf.yaml file.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-sys-firmware]
      description = check machine firmware levels
      executable  = /opt/ibm/csm/hcdiag/tests/chk-sys-firmware/chk-sys-
      firmware.sh
      timeout     = 10
      targetType  = Management
```

/opt/ibm/csm/hcdiag/etc/clustonf.yaml

```
    # rinv <node> firm
    firmware:
        name: 1742FxRev10
        versions:
         - 'BMC Firmware Product:   ibm-v2.0-0-r46-0-gbed584c (Active)*'
         - 'HOST Firmware Product:   IBM-witherspoon-ibm-OP9_v1.19_1.179 (Active)*'
         - 'HOST Firmware Product: -- additional info: buildroot-2018.02.1-6-ga8d1126'
         - 'HOST Firmware Product: -- additional info: capp-ucode-p9-dd2-v4'
         - 'HOST Firmware Product: -- additional info: hostboot-4d6dfdd-p2fc8b56'
         - 'HOST Firmware Product: -- additional info: hostboot-binaries-20c1829'
         - 'HOST Firmware Product: -- additional info: linux-4.16.13-openpower1-p3acb43b'
         - 'HOST Firmware Product: -- additional info: machine-xml-94a137f'
         - 'HOST Firmware Product: -- additional info: occ-f796766'
         - 'HOST Firmware Product: -- additional info: op-build-v2.0.3-308-ge0ef930'
         - 'HOST Firmware Product: -- additional info: petitboot-v1.7.1-pc15aad2'
         - 'HOST Firmware Product: -- additional info: sbe-1b143b5'
         - 'HOST Firmware Product: -- additional info: skiboot-v6.0.5'
```

### *6.2.36 chk-sys-firmware-local*

```
NAME
    chk-sys-firmare-local – check the firmware level on a node


SYNOPSIS
    chk-sys-firmware-local.sh [-v] [-h]

INVOCATION
    hcdiag_run.py -test chk-sys-firmware-local [--target <noderange>]


DESCRIPTION
     Uses the data from sysfs,
     /sys/firmware/devicetree/base/ivm,firmware-versions directory
     to get firmware levels and check if it is what is expected.

     I can run concurrently with user jobs.

     See also chk-sys-firmware.


ARGUMENTS
     -v    Set the verbose mode.

     -h     Display the help message.



RETURNS
     Returns PASS if the firmware versions match the clustconf.yaml
     definition.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-sys-firmware-local]
      description = check firmware levels on the node
      executable  = /opt/ibm/csm/hcdiag/tests/chk-sys-firmware-local/chk-sys-
      firmware-local.sh
      timeout     = 10

    /opt/ibm/csm/hcdiag/etc/clustonf.yaml
      # rinv <node> firm
```

```
firmware:
     name: 1742FxRev10
     versions:
       - 'BMC Firmware Product:   ibm-v2.0-0-r46-0-gbed584c (Active)*'
       - 'HOST Firmware Product:   IBM-witherspoon-ibm-OP9_v1.19_1.179 (Active)*'
       - 'HOST Firmware Product: -- additional info: buildroot-2018.02.1-6-ga8d1126'
       - 'HOST Firmware Product: -- additional info: capp-ucode-p9-dd2-v4'
       - 'HOST Firmware Product: -- additional info: hostboot-4d6dfdd-p2fc8b56'
       - 'HOST Firmware Product: -- additional info: hostboot-binaries-20c1829'
       - 'HOST Firmware Product: -- additional info: linux-4.16.13-openpower1-p3acb43b'
       - 'HOST Firmware Product: -- additional info: machine-xml-94a137f'
       - 'HOST Firmware Product: -- additional info: occ-f796766'
       - 'HOST Firmware Product: -- additional info: op-build-v2.0.3-308-ge0ef930'
       - 'HOST Firmware Product: -- additional info: petitboot-v1.7.1-pc15aad2'
       - 'HOST Firmware Product: -- additional info: sbe-1b143b5'
       - 'HOST Firmware Product: -- additional info: skiboot-v6.0.5'
```

### *6.2.37 chk-temp*

```
NAME
    chk-temp - check the temperature sensors on a node


SYNOPSIS
    chk-temp.sh [-v] [-h]


INVOCATION
    hcdiag_run.py -test chk-temp [ -target <noderange> ]


DESCRIPTION
     This test checks the node's temperature sensors with the sensors
     command and compares those values to thresholds set in the
     clustconf.yaml file.

      It can run concurrently with user jobs.

      NOTE: lm_sensors package is required.


ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS if the sensor values fall within the thresholds
     defined in the clustconf.yaml file.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
        [tests.chk-temp]
        description = Check the temperature sensors on a node
        executable  = /opt/ibm/csm/hcdiag/tests/chk-temp/chk-temp.sh
        timeout     = 10

    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
        temp:
          celsius_high: "35.0"
          celsius_low: "14.0"
```

### *6.2.38 chk-zombies*

```
NAME
    chk-zombies - check possible stuck zombie tasks


SYNOPSIS
    chk-zombies.sh [-v] [-h]


INVOCATION
    hcdiag_run.py -test chk-zombies [-target <noderange>]


DESCRIPTION
     Uses ps command to checks if there are possible stuck zombie process.

     It can run concurrently with user jobs.


ARGUMENTS
     -v       Set the verbose mode.

     -h       Display the help message.



RETURNS
     Returns PASS if there no stuck zombie process.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.chk-zombies]
      description = check possible stuck zombie tasks
      executable  = /opt/ibm/csm/hcdiag/tests/chk-zombies/chk-zombies.sh
      timeout     = 10
```

### *6.2.39 compdiag*

```
NAME
    compdiag – comprehensive fabric diagnostic test


SYNOPSIS
    compdiag.sh [-v] [-h]


INVOCATION
    hcdiag_run.py -test compdiag [-target <node>]


DESCRIPTION
     This test is a comprehensive fabric evaluation and only needs
     to run on one fabric connected node.
     It runs /usr/bin/ibdiagnet with expected link width and speed
     as set in clustconf.yaml and a bit error rate (BER) of 1e-14.

     NOTE: sudo privilege is required to run this test.
           Only run this test on Witherspoon.
           Only one node should be specified.

ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS if there are no errors in the fabric,
     as evaluated by ibdiagnet.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
        [tests.compdiag]
        description = comprehensive fabric diagnostic test
        executable  = /opt/ibm/csm/hcdiag/tests/compdiag/compdiag.sh
        group       = ib
        timeout     = 600

    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
        ib:
          link_width: “4x”
          link_speed: “25”
```

### *6.2.40 daxpy/daxpy-per-socket*

```
NAME
    daxpy - measure aggregate memory bandwidth in MB/s.
    daxpy-per-socket - measure aggregate memory bandwidth per socket MB/s.


SYNOPSIS
    daxpy [-s] [-c] [-v] [-h]


INVOCATION
    hcdiag_run.py --test daxpy [-target <noderange>]
    hcdiag_run.py -test daxpy-per-socket [-target <noderange>]



DESCRIPTION
     Invokes run.daxpy script from the IBM Spectrum-mpi package.

     NOTE: IBM Spectrum MPI package is required.
           sudo privileges is required, if -c argument is passed.
           Please read the
           /opt/ibm/spectrum_mpi/healthcheck/POWER[8|9]/daxpy/README.daxpy
           file.


ARGUMENTS
    -s      run daxpy per socket
    -c      clean up os caches before running daxpy
            If -c is passed, sudo privilege is required
    -v      set the verbose mode, prints the raw output files.
    -h      display help information


RETURNS
     Returns PASS if the daxpy meets the expected performance.
     Expected performance is defined in
     /opt/ibm/spectrum_mpi/healthcheck/POWER[8|9]/daxpy/run.daxpy script.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.daxpy]
      description = measure aggregate memory bandwidth in MB/s.
      executable  = /opt/ibm/csm/hcdiag/tests/daxpy/daxpy.sh
      group       = memory
      timeout     = 100
      targetType  = Compute


      [tests.daxpy-per-socket]
      description = measure aggregate memory bandwidth in MB/s, per socket.
      executable  = /opt/ibm/csm/hcdiag/tests/daxpy/daxpy.sh
      group       = memory
```

```
timeout     = 100
targetType  = Compute
args        = -s
```

### *6.2.41 dcgm-diag*

```
NAME
    dcgm-diag – NVIDIA DCGM comprehensive diagnostic


SYNOPSIS
    dcgm-diag.sh [run-level]


INVOCATION
    hcdiag_run.py -test dcgm-diag [-target <noderange>]



DESCRIPTION
     Invokes "dcgmi diag" command from the nvidia datacenter package.
     Validates device sub-components, interlink bandwidth, memory/ECC state,
and deployment software integrity.



ARGUMENTS
   level
       Set the level of diagnostic to run.
       Possible values are:
          1: quick diagnostic (Deployment group)
          2: extended diagnosis (Deployment and Performance groups)
          3: hardware diagnostic (Deployment, Performance and Hardware groups)
       Default it 3.


RETURNS
     Returns PASS if the dcgm returns all tests PASS.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.dcgm-diag]
      description = NVIDIA DCGM diagnostic test
      group       = gpu
      timeout     = 900
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/dcgm-diag/dcgm-diag.sh
      #args        = 3
```

### *6.2.42 dcgm-diag-double*

```
NAME
    dcgm-diag-double - NVIDIA DCGM comprehensive diagnostic with
    double precision test.


SYNOPSIS
    dcgm-diag.sh <level> <stats_path> <debug_path>


INVOCATION
    hcdiag_run.py -test dcgm-diag-double [ -target <noderange> ]


DESCRIPTION
     Invokes "dcgmi diag" command from the nvidia datacenter package.
     Same as dcgmi-diag, but includes the double precision test.


ARGUMENTS
     level        Set to 5 for  this test.
     stats_path   Path for the statistics file
     debug_path   Path for the debug file.

     NOTE: <debug_path>/<hostname>/dcgm.debug has 30 characters limitation.


RETURNS
     Returns PASS if the dcgm returns all tests PASS.


CONFIGURATION FILE
    [tests.dcgm-diag-double]
      description = NVIDIA DCGM double precision diagnostic test
      group       = gpu
      timeout     = 9000
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/dcgm-diag/dcgm-diag.sh
      # args: <which diag> <stats_path> <debug_path>
      ## Note debug_path+<hostname_s>/dcgm.debug has 30 characters limitation
      args        = 5 /tmp /tmp
```

### *6.2.43 dcgm-diag-single*

```
NAME
    dcgm-diag-single – NVIDIA DCGM comprehensive diagnostic with
    single precision test.


SYNOPSIS
    dcgm-diag.sh [run-level] <stats_path> <debug_path>



INVOCATION
    hcdiag_run.py –test dcgm-diag-single [ –target <noderange> ]


DESCRIPTION
     Invokes "dcgmi diag" command from the nvidia datacenter package.
     Same as dcgm-diag test, but includes single precision test.



ARGUMENTS
     level        Set to 4 for  this test.
     stats_path   Path for the statistics file
     debug_path   Path for the debug file.

     NOTE: <debug_path>/<hostname>/dcgm.debug has 30 characters limitation.



RETURNS
     Returns PASS if the dcgm returns all tests PASS.


CONFIGURATION FILE
    [tests.dcgm-diag-single]
      description = NVIDIA DCGM single precision diagnostic test
      group       = gpu
      timeout     = 9000
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/dcgm-diag/dcgm-diag.sh
      # args: <which diag> <stats_path> <debug_path>
      ## Note debug_path+<hostname_s>/dcgm.debug has 30 characters limitation
      args        = 4 /tmp /tmp
```

### *6.2.44 dcgm-health*

```
NAME
    dcgm-health – NVIDIA Health Monitoring


SYNOPSIS
    dcgm-health.sh [watches]


INVOCATION
    hcdiag_run.py –test dcgm-health [–target <noderange>]


DESCRIPTION
     Checks overall health for the GPU subsystems: PCIe, SM, MCU,
     PMU, Inforom, Power and thermal system.

     It can run concurrently with user jobs.


ARGUMENTS
     watches    what to check.
       Possible values are:
          a: all watches
          p: PCIe watches
          m: memory watches
          i: inforom watches
          t: thermal and power waches
          n: NVLink watches

        Default is all watches.


RETURNS
     Returns PASS if the dcgmi health returns “heathy”.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.dcgm-health]
      description = NVIDIA DCGM health check
      group       = gpu
      timeout     = 100
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/dcgm-health/dcgm-health.sh
      #args        = a
```

### *6.2.45 dgemm/dgemm-per-socket*

```
NAME
    dgemm  - check CPU performance.
    dgemm-per-socket - check CPU performance per socket.


SYNOPSIS
    dgemm.sh [-s] [-v] [-m] [-h]

INVOCATION
    hcdiag_run.py -test dgemm [ -target <noderange> ]
    hcdiag_run.py -test dgemm-per-socket [ -target <noderange> ]


DESCRIPTION
    Checks the CPU performance using DGEMM.
    (double-precison general matrix-matrix multiply).

    NOTE: Requires IBM Spectrum MPI package installed.


ARGUMENTS
     -s     runs dgemm in socket mode
     -v     sets the verbose mode, prints the raw output file(s).
     -m     uses mpirun instead of jsrun
     -h     print usage information


RETURNS
     Returns PASS if the CPU performance is what is expected.
     For more details on expected performance, please read
     /opt/ibm/spectrum_mpi/healthcheck/POWER[8|9]/dgemm/README.dgemm


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.dgemm]
      description = measure CPU performance.
      executable  = /opt/ibm/csm/hcdiag/tests/dgemm/dgemm.sh
      group       = cpu
      timeout     = 600
      targetType  = Compute


    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.dgemm-per-socket]
      description = measure CPU performance per socket.
      executable  = /opt/ibm/csm/hcdiag/tests/dgemm/dgemm.sh
      group       = cpu
      timeout     = 800
      targetType  = Compute
      args        = -s
```

### *6.2.46 dgemm-gpu*

```
NAME
    dgemm-gpu  - check GPU performance


SYNOPSIS
    dgemm_gpu.sh [-l] [-c] [-v] [-m] [-h]


INVOCATION
    hcdiag_run.py -test dgemm-gpu [-target <noderange>]


DESCRIPTION
    Checks the GPU performance invoking IBM Spectrum MPI's DGEMM
    (double-precision general matrix-matrix multiply) for gpu.
    It also reads the clock speeds, power and temperature of the GPUs
    while the test is running for information.  The clock speeds, power
    and temperature is informational only, no check PASS/FAIL is performed
    with this data.

    NOTE: By default, the user running this test requires "sudo"
          privileges,in order to set/unset the GPU clock frequency.

          Please read /opt/ibm/spectrum_mpi/healthcheck/POWER[8|9]/
          dgemm/README.dgemm for more details about expected performance,
          clock setting, etc.

          Requires IBM Spectrum MPI package installed.


ARGUMENTS
   -l         sets GPU application clocks should be set according
              to the README.dgemm_gpu file.
              If -l is passed, "sudo" privilege is required.
   -c v1,v2   sets the GPU application clocks <memory, graphics>.
              Run 'nvidia-smi -q -d SUPPORTED_CLOCKS' to see list of
              supported clock combinations.
              Only works with -l option.
              Default values is 877,1342
   -v         sets verbose mode, prints dgemm-gpu raw files
   -m         use mpirun instead of jsrun
   -h         print the help message (usage)


RETURNS
     Returns PASS if the GPU performance is what is expected.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.dgemm-gpu]
```

```
description  = measure GPU performance.
executable  = /opt/ibm/csm/hcdiag/tests/dgemm-gpu/dgemm-gpu.sh
group        = gpu
timeout      = 2000
targetType   = Compute
args         = -l
```

### 6.2.47 *fieldiag*

NAME
fieldiag – NVIDIA field diagnostic

SYNOPSIS
fieldiag.sh [args]

INVOCATION
fieldiag.sh [args]

**** not recommended** hcdiag_run.py –test fieldiag [–target <noderange>]

DESCRIPTION
Offline diagnostic. Tests in the package provide confirmation of the numerical processing engines in the GPU, integrity of data transfers to and from the GPU, and test coverage of the full onboard memory address space available to CUDA programs.

NVIDIA distributes fieldiag package in a tar compressed file (629-IBM04-UNIV-ALL.tgz). Diagnostic expects the contents of this file in /opt/ibm/csm/hcdiag/fieldiag directory by default. If not, /opt/ibm/csm/hcdiag/fieldiag/fieldiag.sh script has to be modified to point to the correct directory containing the contents of the tar file.

It performs the following steps:

- stops all daemons that uses the kernel modules: CSM, nv-hostengine, nvidia_persistenced
- remove nvidia modules (rmmod)
- invoke install_module.sh (in tgz file): compiles *mods* kernel module, install it.
- Invoke fieldiag (run fieldiag test)

This test is used to validate a Return Merchandise Authorization(RMA).

NOTE: sudo privilege is required

After running this test, machine has to be rebooted.

This test requires several hours for full coverage.

ARGUMENTS
args NVIDIA fieldiag argument.
Please refer to the NVIDIA documentation.

Default is “pex_lanes=2” : SXM2 connector based GPU

RETURNS

```
     Returns PASS if NVIDIA fieldiag returns 0. For details of fieldiag
     return codes, please refer to the NVIDIA documentation.


CONFIGURATION FILE
   /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.fieldiag]
       description = NVIDIA hardware diagnostic
       executable  = /opt/ibm/csm/hcdiag/tests/fieldiag/fieldiag.sh
       group       = gpu
       timeout     = 25000

   /opt/ibm/csm/hcdiag/tests/fieldiag/fieldiag.sh
       export thisdir=`pwd`
       export NVIDIA_FIELDIAG=$thisdir
```

### *6.2.48 gpfsperf*

```
NAME
    gpfsperf - GPFS benchmark to measure the performance of the Fileystem


SYNOPSIS
    gpfsperf.sh <gpfs_fs> [blocksize] [directory]


INVOCATION
    hcdiag_run.py –test gpfsperf [–target <noderange>]


DESCRIPTION
    gpfsperf is a simple benchmark program that can be used to
    measure the performance of GPFS for several common file access patterns.
    gpfs.base package is required.

    The source code for gpfsperf can be found in
    /usr/lpp/mmfs/samples/perf directory.

    Lots of customization is required to run this test: which test to run
    and thresholds.
    The changes should be done in place, opt/ibm/csm/hcdiag/tests/gpfsperf.sh
    file.


ARGUMENTS
     gpfs_fs       The name of the FILEystem

     blocksize     The block size of the test file.
                   Default is 1M.

     directory     The name of directory to create/write/read FILE.
                    Make sure the user running this test has r/w permission.
                   Default is <GPFS_FS>/diagadmin


RETURNS
     Returns PASS if all the tests selected: create, read sequential,
     read ranhit, are executed correctly and the performance are within
     the thresholds.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.gpfsperf]
      description = GPFS benchmark to measure the performance of the
      FILEystem
      executable  = /opt/ibm/csm/hcdiag/tests/gpfsperf/gpfsperf.sh
```

```
   group       = FILEystem
   #             <gpfs_fs>   [blocksize]  [directory]
   args        = /gpfs/gpfs0   1M  /gpfs/gpfs0/diagadmin
   timeout     = 25000
```

/opt/ibm/csm/hcdiag/tests/gpfsperf.sh

```
    # Define here the tests that you want to perform.
    # Use "-" if you don't care about the values data_rate, op_rate, avg_latency and
    thread_untilization
    # Example: "read randhint"   "read randhint ${TEST_FILE} -r 10000 -n 100m"
    569130.66  542.77   1.842     -
    TESTS=(
    # id              args                                                    data_rate
    op_rate  avg_latency thread_utilization
    "create"          "create seq ${TEST_FILE} -n 10g -r ${BLOCK_SIZE} -fsync"   720000.00
    690.20   2.000       -
    "read sequencial" "read seq ${TEST_FILE} -r 1M"                           549130.66
    540.77   2.000       -
    "read rand"       "read rand ${TEST_FILE} -r 10000 -n 100m"                 4500.00
    440.77   2.000       -
    "read randhint"   "read randhint ${TEST_FILE} -r 10000 -n 100m"             19000.00
    2500.00   1.000       -
    )
```

### *6.2.49 gpudirect*

```
NAME
    gpudirect - exercise GPUDirect RDMA and p2p data movement


SYNOPSIS
    gpudirect.sh <noderange>


INVOCATION
    hcdiag_run.py -test gpudirect -target <noderange>


DESCRIPTION
     Runs RDMA Write BW tests with /usr/bin/ib_write_bw and the flag
     --use_cuda to send data between nodes. The first node in the noderange
     is treated as a server, and the remaining nodes as clients. Displays
     full BW results for each node pair.


     NOTE: This test only runs in Management mode.
           User running this test has to have xCAT privileges.
           The xCAT-client package is required.


ARGUMENTS
        noderange   two or more nodes, in XCAT noderange format.


RETURNS
     Returns PASS if there is successful data movement between the
     client and server nodes.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.gpudirect]
      description = Exercise GPUDirect RDMA and p2p data movement
      executable  = /opt/ibm/csm/hcdiag/tests/gpudirect/gpudirect.sh
      targetType  = Management
      group       = ib
      timeout     = 150
```

### *6.2.50 gpu-health*

```
NAME
    gpu-health - check gpu memory bw, gpu dgemm flops, nvlink transfer speeds


SYNOPSIS
    gpu-health.sh [nvlink_speed] [gpu_mem_bw] [dgemm_flops]


DESCRIPTION
     Run GPU dgemm, GPU daxpy, host to device and device to host transfer
     rate on all gpus per socket.
     The gpu-health has to be compiled by the customer. Source code is
     Installed in the  /opt/ibm/csm/hcdiag/samples/gpu-health directory.
     The makefile is setup to compile for P9 GPUs, it not, make the
     appropriate changes ( see
     /opt/ibm/csm/hcdiag/samples/gpu-health/makefile file).



ARGUMENTS
     nvlink_speed    threshold for the device to host and host to device
                     transfer rate. Default is 65.
     gpu_mem_bw      threshold for the GPU daxpy bandwidth. Default is 800.
     dgemm_tflops    threshold for the GPU dgemm bandwidth. Default is 6.7.


RETURNS
     Returns PASS the GPU daxpy bandwidth, the trafer rates and the GPU
     dgemm are equal or greater than the threshold.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.gpu-health]
      description = checks gpu memory bw, gpu dgemm flops, nvlink transfer
                    speeds
      executable  = /opt/ibm/csm/hcdiag/tests/gpu-health/gpu-health.sh
      group       = gpu
      #args       = 40 800 6.7 for 6 GPUs
      #args       = 65 800 6.7 default fot 4 GPUs.
      timeout     = 1000
```

### *6.2.51 hcatest*

```
NAME
    hcatest - verify operating hca with verbs latency test


SYNOPSIS
    hcatest.sh <noderange>


INVOCATION
    hcdiag_run.py -test hcatest -target <noderange>


DESCRIPTION
     Runs send latency tests with /usr/bin/ib_send_lat to send data between
     nodes. The first node in the noderange is treated as a server, and the
     remaining nodes as clients.
     Displays full latency results for each node pair.

     NOTE: This test only runs in Management mode.
           User running this test has to have xCAT privileges.
           The xCAT-client package is required.


ARGUMENTS
      noderange     One of more nodes, in XCAT noderange format.



RETURNS
     Returns PASS if there is successful data movement between the client
     and server nodes.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hcatest]
      description = Verify operating hca with verbs latency test
      executable  = /opt/ibm/csm/hcdiag/tests/hcatest/hcatest.sh
      targetType  = Management
      group       = ib
      timeout     = 150
```

### *6.2.52 hxecache*

```
NAME
    hxecache – HTX L1/L2/L3 caches stress test


SYNOPSIS
    hxecache.sh <duration> [verbose]


INVOCATION
    hcdiag_run.py -test hxecache [-target <noderange>]



DESCRIPTION
     L1, L2 and L3 cache stress and prefetch irritator.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.



ARGUMENTS
     duration   number of the cycles to execute.
     verbose    0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxecache program return no errors.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxecache]
      description = HTX L1/L2/L3 caches stress test
      group       = memory
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxecache/hxecache.sh
      args        = 10
```

### *6.2.53 hxecpu*

```
NAME
    hxecpu - HTX processor core features test


SYNOPSIS
    hxecpu <duration> <level> [verbose]


INVOCATION
    hcdiag_run.py -test hxecpu [-target <noderange>]



DESCRIPTION
     Exerciser for generic processor core features testing.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.



ARGUMENTS
     duration
         Number of the cycles to execute.

     level
         It defines the level of diagnostic.
         It is set to 1 for this test, which means it is  pass1 test:
         a shorter test duration, and light resource usage.

     verbose
         0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxecpu program return no errors.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests. tests.hxecpu]
      description = HTX processor core cpu test
      group       = cpu
      timeout     = 800
      targetType  = Compute
      executable = /opt/ibm/csm/hcdiag/tests/hxecpu/hxecpu.sh
      args        = 10 1
```

### *6.2.54 hxecpu_pass2*

```
NAME
    hxecpu - HTX processor core features test


SYNOPSIS
    hxecpu <duration> <level> [verbose]


INVOCATION
    hcdiag_run.py -test hxecpu_pass2 [-target <noderange>]


DESCRIPTION
     Exerciser for generic processor core features testing.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.



ARGUMENTS
     duration
         Number of the cycles to execute.
     level
         It defines the level of diagnostic.
         It is set to 2 for this test, which means it is an pass2, a longer
         test duration, and used for in-deep analysis.
     verbose
         0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxecpu program return no errors.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxecpu_pass2]
      description = HTX processor core cpu test pass2
      group       = cpu
      timeout     = 800
      targetType  = Compute
      executable = /opt/ibm/csm/hcdiag/tests/hxecpu/hxecpu.sh
      args        = 10 2
```

### *6.2.55 hxediag_eth*

```
NAME
    hxediag_eth - HTX exerciser for Broadcom NIC adapters.


SYNOPSIS
    hxediag  <interface> <duration> [args]


INVOCATION
    hcdiag_run.py -test hxediag_eth [-target <noderange>]



DESCRIPTION
     Diagnostic exerciser for Broadcom NIC.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.


ARGUMENTS
     interface  "eth" is set for this test. test all Ethernet adapters

     duration   number of the cycles to execute.

     args       specify one or a list of adapters to be tested.
                Default is all.


RETURNS
     Returns PASS if hxecpu program return no errors.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxediag_eth]
      description = HTX test for Ethernet adapters
      group       = io
      timeout     = 800
      targetType  = Compute
      executable = /opt/ibm/csm/hcdiag/tests/hxediag/hxediag.sh
      args        = eth 10
```

### *6.2.56 hxediag_ib*

```
NAME
    hxediag - HTX exerciser for IB adapters.


SYNOPSIS
    hxediag <interface> <duration> [args]


INVOCATION
    hcdiag_run.py -test hxediag_ib [-target <noderange>]


DESCRIPTION
     Diagnostic exerciser for IB adapters.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.



ARGUMENTS
     interface  Set to ib for this test

     duration   Number of the cycles to execute.

     args       Specify one or a list of adapters to be tested.
                Default is all.

RETURNS
     Returns PASS if hxecpu program return no errors.



CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxediag_ib]
      description = HTX test for IB adapters
      group       = io
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxediag/hxediag.sh
      args        = ib 10
```

### *6.2.57 hxeewm_pass2*

```
NAME
    hxeewm - HTX Power and thermal testing


SYNOPSIS
    hxeesm <duration> [verbose]



INVOCATION
    hcdiag_run.py -test hxeewm_pass2 [-target <noderange>]


DESCRIPTION
    Stress test thermal aspects of Power server.
    -  Includes various workloads to stress processor core and memory
    -  Provides processor thread level workload control.
    -  Perform TDP, RDP, MSRP type of tests.

    This test is defined as pass2, long duration test and used for
    in-deep analyses.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.



ARGUMENTS
     duration   number of the cycles to execute.
     verbose    0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxeewm program return no errors.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests. tests.hxeewm_pass2]
      description = HTX thermal test
      group       = processor
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxeewm/hxeewm.sh
      args        = 10
```

### *6.2.58 hxefabricbus_pass2*

```
NAME
    hxefabricbus - HTX processor fabric bus test


SYNOPSIS
    hxefaricbus.sh <duration> [verbose]


INVOCATION
    hcdiag_run.py -test hxefabricbus_pass2 [-target <noderange>]



DESCRIPTION
   Perform intra and inter node links tests. The test sends various data
   patterns over the links.

    This test is defined as pass2, long duration test and used for
    in-deep analyses.

    NOTE: sudo privilege is required to run this test.
          HTX package is required for this test.
          We recommend not changing the test.properties file for this test.



ARGUMENTS
     duration   number of the cycles to execute.
     verbose    0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxefabricubus program return no errors.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxefabricbus_pass2]
      description = HTX processor fabric bus test
      group       = processor
      timeout     = 800
      targetType  = Compute
      executable = /opt/ibm/csm/hcdiag/tests/hxefabricbus/hxefabricbus.sh
      args        = 10
```

### *6.2.59 hxefpu64*

```
NAME
    hxefpu64 – HTX VSU (Vector Scalar Unit) test


SYNOPSIS
    Hxefpu64 <duration> <level> [verbose]



INVOCATION
    hcdiag_run.py –test hxefpu64 [–target <noderange>]


DESCRIPTION
     Test the VSU (Vector Scalar Unit) of power processors by generating
     pseudo random sequence of instructions.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.



ARGUMENTS
     duration   number of the cycles to execute.

     level      set to 1 for this test, which means it is a pass1 test:
                shorter test duration, and light resource usage.

     verbose    0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxefpu64 program return no errors.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxefpu64]
      description = HTX VSU (Vector Scalar Unit) test
      group       = processor
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxefpu64/hxefpu64.sh
      args        = 10 1
```

### *6.2.60 hxefpu64_pass2*

```
NAME
    hxefpu64 – HTX VSU (Vector Scalar Unit) test


SYNOPSIS
    Hxefpu64 <duration> <level> [verbose]


INVOCATION
    hcdiag_run.py –test hxefpu64_pass2 [–target <noderange>]


DESCRIPTION
     Test the VSU (Vector Scalar Unit) of power processors by generating
     pseudo random sequence of instructions.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.


ARGUMENTS
     duration   number of the cycles to execute.
     level      set to 2 for this test, pass2 test: a longer test duration,
                and used for in-deep analysis.
     verbose    0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxefpu64 program return no errors.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxefpu64_pass2]
      description = HTX VSU (Vector Scalar Unit) test pass2
      group       = processor
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxefpu64/hxefpu64.sh
      args        = 10 2
```

### *6.2.61 hxemem64*

```
NAME
    hxemem64 -  HTX memory subsystem test


SYNOPSIS
    hxemem64 <duration> <level> [verbose]


INVOCATION
    hcdiag_run.py -test hxemem64 [-target <noderange>]


DESCRIPTION
     Exerciser for memory subsystem of power system.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.


ARGUMENTS
     duration   number of the cycles to execute.

     level      set to 1 for this test, a pass1 test: shorter test duration,
                and light resource usage.

     verbose    0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxemem64 program return no errors.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxemem64]
      description = HTX memory subsystem test
      group       = memory
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxemem64/hxemem64.sh
      args        = 2 1
```

### *6.2.62 hxemem64_pass2*

```
NAME
    hxemem64 -  HTX memory subsystem test


SYNOPSIS
    hxemem64 <duration> <level> [verbose]



INVOCATION
    hcdiag_run.py -test hxemem64_pass2 [-target <noderange>]




DESCRIPTION
     Exerciser for memory subsystem of power system.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.



ARGUMENTS
     duration   number of the cycles to execute.
     level      set to 2 for this test, pass2test:  longer test duration, and
                used for in-deep analysis.
     verbose    0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxemem64 program return no errors.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxemem64_pass2]
      description = HTX memory subsystem test
      group       = memory
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxemem64/hxemem64.sh
      args        = 2 2
```

### *6.2.63 hxenvidia*

```
NAME
    hxenvidia -  HTX gpu test

SYNOPSIS
    hxenvidia <duration> <level> [verbose]

INVOCATION
    hcdiag_run.py -test hxenvidia [-target <noderange>]


DESCRIPTION
     Exerciser for Nvidia GPUs in IBM Power system.  Stress test on GPU's
     using various  workload provided by Nvidia.

     hxenvidia for Nvidia adapter (K40, K80) has following testcases :
         o  NBODY Workload - Compute intensive workload which simulates boby
            movement due gravitation forces exerted on it owing to other
            bodies in its neighbor.
         o  GEMM Workload -   Does General purpose matrix multiplication
            using CUDA CuBLAS library. A very power hungry and leads to large
            temperature spikes workload.
         o  PCIe Bandwith Test : Workload to transfer huge amount of data
            from Host memory to GPU memory and GPU memory to Host memory.
            Stresses PCIe bandwidth link
               ▪  connecting GPUs.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.


ARGUMENTS
     duration   number of the cycles to execute.

     level      set to 1 for this test, pass1 test: shorter test duration,
                and light resource usage.

     verbose    0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxenvidia program return no errors.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxenvidia]
      description = HTX gpu test
      group       = gpu
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxenvidia/hxenvidia.sh
      args        = 10 1
```

### *6.2.64 hxenvidia_pass2*

```
NAME
    hxenvidia -  HTX gpu test


SYNOPSIS
    hxenvidia <duration> <level> [verbose]


INVOCATION
    hcdiag_run.py -test hxenvidia_pass2 [-target <noderange>]


DESCRIPTION
     Exerciser for Nvidia GPUs in IBM Power system.  Stress test on GPU's
     using various  workload provided by Nvidia.

     hxenvidia for Nvidia adapter (K40, K80) has following testcases :
         o  NBODY Workload - Compute intensive workload which simulates boby
            movement due gravitation forces exerted on it owing to other
            bodies in its neighbor.
         o  GEMM Workload -   Does General purpose matrix multiplication
            using CUDA CuBLAS library. A very power hungry and leads to large
            temperature spikes workload.
         o  PCIe Bandwith Test : Workload to transfer huge amount of data
            from Host memory to GPU memory and GPU memory to Host memory.
            Stresses PCIe bandwidth link
               ▪  connecting GPUs.

    NOTE: sudo privilege is required to run this test.
          HTX package is required for this test.
          We recommend not changing the test.properties file for this test.


ARGUMENTS
     duration   number of the cycles to execute.
     level      set to 2 for this test, a pass2 test: longer test duration,
                and used for in-deep analysis.
     verbose    0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxenvidia program return no errors.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxenvidia_pass2]
      description = HTX gpu test
      group       = gpu
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxenvidia/hxenvidia.sh
      args        = 10 2
```

### *6.2.65 hxerng_pass2*

```
NAME
    hxerng – HTX random number generator engine test


SYNOPSIS
    hxerng.sh <duration> [verbose]


INVOCATION
    hcdiag_run.py –test hxerng_pass2 [–target <noderange>]


DESCRIPTION
    Exerciser for the random number generator engine.

    This test is defined as pass2, long duration test and used for in-deep
    analyses.


     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.


ARGUMENTS
     duration   number of the cycles to execute.
     verbose    0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxerng program return no errors.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxerng_pass2]
      description = HTX random number generator engine test
      group       = core
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxerng/hxerng.sh
      args        = 10
```

### *6.2.66 hxesctu_pass2*

```
NAME
    hxesctu - HTX process cache coherency test


SYNOPSIS
    hxexctu.sh <duration> [verbose]


INVOCATION
    hcdiag_run.py -test hxe [-target <noderange>]


DESCRIPTION
    Power processor cache coherency test.

    This test is defined as pass2, long duration test and used for
    in-deep analyses.

    NOTE: sudo privilege is required to run this test.
          HTX package is required for this test.
          We recommend not changing the test.properties file for this test.



ARGUMENTS
     duration   number of the cycles to execute.
     verbose    0|1 : turn the verbose mode off|on. Default is off.


RETURNS
     Returns PASS if hxesctu program return no errors.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxesctu_pass2]
      description = HTX processor cache coherency test
      group       = memory
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxesctu/hxesctu.sh
      args        = 10
```

### *6.2.67 hxestorage_nvme*

```
NAME
    hxestorage -  HTX storage subsystem test for NVMe.


SYNOPSIS
    hxestorage <subsystem> <duration> <level>


INVOCATION
    hcdiag_run.py -test hxestorage [-target <noderange>]


DESCRIPTION
     Test the storage subsystem by using end NVMe devices as target.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.


ARGUMENTS
     subsystem   is set to nvme for this test.

     duration    number of the cycles to execute.

     level       set to 1 for this test, a pass1 test: shorter test duration,
                 and light resource usage.



RETURNS
     Returns PASS if hxestorage program return no errors.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxestorage_nvme]
      description = HTX storage subsystem test
      group       = storage
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxestorage/hxestorage.sh
      args        = nvme 10 1
```

### *6.2.68 hxestorage_nvme_pass2*

```
NAME
    hxestorage -  HTX storage subsystem test for NVMe.


SYNOPSIS
    hxestorage <subsystem> <duration> <level>


INVOCATION
    hcdiag_run.py -test hxestorage_nvme_pass2 [--target <noderange>]




DESCRIPTION
     Test the storage subsystem by using end NVMe devices as target.

      NOTE: sudo privilege is required to run this test.
            HTX package is required for this test.
            We recommend not changing the test.properties file for this test.

ARGUMENTS
     subsystem   set to nvme for this test.

     duration    number of the cycles to execute.

     level       set to 2 for thist test, a pass2 test: longer test duration,
                 and used for in-deep analysis.


RETURNS
     Returns PASS if hxestorage program return no errors.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxestorage_nvme_pass2]
      description = HTX storage subsystem test
      group       = storage
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxestorage/hxestorage.sh
      args        = nvme 10 2
```

### *6.2.69 hxestorage_sd*

```
NAME
    hxestorage -  HTX storage subsystem test for disk.


SYNOPSIS
    hxestorage <subsystem> <duration> <level>


INVOCATION
    hcdiag_run.py -test hxestotage_sd [-target <noderange>]


DESCRIPTION
     Test the storage subsystem by using disk end devices as target.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.


ARGUMENTS
     subsystem   set to sd for this test.

     duration    number of the cycles to execute.

     level       set to 1 for this test, a pass1 test:  shorter test
                 duration, and light resource usage.



RETURNS
     Returns PASS if hxestorage program return no errors.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxestorage_sd]
      description = HTX storage subsystem test
      group       = storage
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxestorage/hxestorage.sh
      args        = sd 10 1
```

### *6.2.70 hxestorage_sd_pass2*

```
NAME
    hxestorage -  HTX storage subsystem test for disk.


SYNOPSIS
    hxestorage <subsystem> <duration> <level>


INVOCATION
    hcdiag_run.py -test hxestorage_sd_pass2 [-target <noderange>]


DESCRIPTION
     Test the storage subsystem by using disk end devices as target.

     NOTE: sudo privilege is required to run this test.
           HTX package is required for this test.
           We recommend not changing the test.properties file for this test.



ARGUMENTS
     subsystem   set to sd for this test.

     duration    number of the cycles to execute.

     level       set to 2 for this test, a pass2 test: longer test duration,
                 and used for in-deep analysis.


RETURNS
     Returns PASS if hxestorage program return no errors.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.hxestorage_sd_pass2]
      description = HTX storage subsystem test
      group       = storage
      timeout     = 800
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/hxestorage/hxestorage.sh
      args        = sd 10 2
```

### *6.2.71 ibcredit*

```
NAME
    ibcredit - checks for credit loops


SYNOPSIS
    ibcredit.sh [-v] [-h]


INVOCATION
    hcdiag_run.py -test ibcredit [-target <node>]


DESCRIPTION
     This test checks for credit loops and only needs to run on one fabric
     connected node. It runs /usr/bin/ibdiagnet with a configuration set to
     skip any checks unrelated to credit loops.

     NOTE: sudo privilege is required to run this test.
           Run only on Witherspoon.


ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS if there are no credit loops, as determined by ibdiagnet.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.ibcredit]
      description = Checks for credit loops
      executable  = /opt/ibm/csm/hcdiag/tests/ibcredit/ibcredit.sh
      group       = ib
      timeout     = 10
```

### *6.2.72 ibverbs*

```
NAME
    ibverbs - verify verbs point to point communication between nodes


SYNOPSIS
    ibverbs.sh <noderange>


INVOCATION
    hcdiag_run.py -test ibverbs -target <noderange>


DESCRIPTION
     Runs Send Bidirectional BW tests with /usr/bin/ib_send_bw to send data
     between nodes. The first node in the noderange is treated as a server,
     and the remaining nodes as clients. Displays full BW results for each
     node pair.

      NOTE: This test only runs in Management mode.
            User running this test has to have xCAT privileges.
            The xCAT-client package is required.


ARGUMENTS
     noderange     One of more nodes, in XCAT noderange format.


RETURNS
     Returns PASS if there is successful data movement between the
     client and server nodes.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.ibverbs]
      description = Verify verbs point to point communication between nodes
      executable  = /opt/ibm/csm/hcdiag/tests/ibverbs/ibverbs.sh
      targetType  = Management
      group       = ib
      timeout     = 150
```

### *6.2.73 ipoib*

```
NAME
    ipoib - shows the state, MTU, and mode of IPoIB devices on a node


SYNOPSIS
    ipoib.sh [-v] [-h]


INVOCATION
    hcdiag_run.py -test ipoib [-target <noderange>]


DESCRIPTION
     Checks the device directories in /sys/class/net/ to print
     information about the state, MTU, and mode of IPoIB devices on a node.

     It can run concurrently with user jobs.


ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS if there were no errors obtaining information
     from the FILE in the device directories.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
       [tests.ipoib]
       description = Shows the state, MTU, and mode of IPoIB devices
       executable  = /opt/ibm/csm/hcdiag/tests/ipoib/ipoib.sh
       timeout     = 10
```

### *6.2.74 jlink*

```
NAME
    jlink - measures the bandwidth between nodes, checking for bad or low-
performing links.


SYNOPSIS
    jlink.sh [-j] [-v] -h] <noderange>


INVOCATION
    hcdiag_run.py -test jlink -target <noderange>


DESCRIPTION
      Tests bandwidth for all possible node pairings and report bad or low
      performing links.

      NOTE: This test does not run with user root
            IBM Spectrum mpi package is required to run this test.


ARGUMENTS
      -j            Use jsrun instead of mpirun (not working)
      -v            Set verbose mode, print raw output files
      -h            Display help message
      noderange     Two of more nodes, in XCAT noderange format


RETURNS
     Returns PASS no bad link, or underperforming links are found.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.jlink]
      description = JLINK measures the bandwidth between nodes, and to
      discover bad or low-performing links.
      executable  = /opt/ibm/csm/hcdiag/tests/jlink/jlink.sh
      group       = ib
      timeout     = 50
      targetType  = Compute
      clustertest = yes
```

### *6.2.75 lnklwls*

```
NAME
    lnklwls - verify link width and speed


SYNOPSIS
    lnklwls.sh [-v] [-h]


INVOCATION
    hcdiag_run.py -test lnklwls [ -target <node> ]


DESCRIPTION
     This test verifies link width and speed and only needs to run on
     one fabric connected node. It runs /usr/bin/ibdiagnet with a
     configuration set to skip any checks unrelated to link width or speed.

     NOTE: sudo privilege is required to run this test.
           Only run this test on Witherspoon.

ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS if link width and speed values found by ibdiagnet
     match values configured in the clustconf.yaml file.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.lnklwls]
      description = Verify link width and speed
      executable  = /opt/ibm/csm/hcdiag/tests/lnklwls/lnklwls.sh
      group       = ib
      timeout     = 10

    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
        ib:
          link_width: "4x"
          link_speed: "25"
```

### *6.2.76 lnkqual*

```
NAME
    lnkqual - check for marginal link error


SYNOPSIS
    lnkqual.sh [-v] [-h]


INVOCATION
    hcdiag_run.py -test lnkqual [ -target <node> ]


DESCRIPTION
     This test checks for marginal link error and only needs to run on one
     fabric connected node. It runs /usr/bin/ibdiagnet with link speed as
     configured in the clustconf.yaml file and a bit error rate
     (BER) of 1e-14.

     NOTE: sudo  privilege is required to run this test.
           Only run this test on Witherspoon.

ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS if there are no errors in ibdiagnet's BER rate check.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
       [tests.lnkqual]
       description = Check for marginal link error
       executable  = /opt/ibm/csm/hcdiag/tests/lnkqual/lnkqual.sh
       group       = ib
       timeout     = 10

    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
       ib:
         link_speed: "25"
```

### *6.2.77 mmhealth*

```
NAME
    mmhealth - verify status of the gpfs subsystem in the node

SYNOPSIS
    mmhealth.sh

INVOCATION
    hcdiag_run.py -test mmhealth [-target <noderange>]


DESCRIPTION
     Invokes the program /usr/lpp/mmfs/bin/mmhealth and checks the health
     status of a node and services that are hosted on the node (IBM Spectrum
     Scale).

     It can run concurrently with user jobs.

     NOTE: Requires sudo privileges to run this test.
           It requires IBM Spectrum Scale installed.


ARGUMENTS
     None


RETURNS
     Returns PASS if mmhealth returns no error.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
       [tests. tests.mmhealth]
       description = Verify status of the gpfs in the node
       executable  = /opt/ibm/csm/hcdiag/tests/mmhealth/mmhealth.sh
       timeout     = 10
```

### *6.2.78 nsdperf*

```
NAME
    nsdperf -  Test network performance, simulating GPFS NSD client/server

SYNOPSIS
    nsdperf.sh <noderange> [-s NSERVERS] [-i FILE] [-p PORT] [-r RDMAPORTS]
               [-t NRCV] [-w NWORKERS] [-6] [-d] [-h]


INVOCATION
    hcdiag_run.py -test nsdperf [-target <noderange>]


DESCRIPTION
    The nsdperf program can be used to measure network throughput in a cluster
    and to test network behavior under heavy load.  All network communication
    is done using TCP socket connections or RDMA verbs
    (InfiniBand/iWARP).Messages are sent between client nodes and server
    nodes, with traffic patterns that can mimic the way GPFS NSD clients
    communicate with NSD servers.  The intention is to provide a way to shake
    out the kinds of problems that GPFS typically produces when setting up a
    new cluster,without involving GPFS.

    NOTE: This test only runs in Management mode.

          User running this test has to have xCAT privileges.

          The xCAT-client and gpfs.base packages are required.

          The source code for nsdperf can be found in
          /usr/lpp/mmfs/samples/net directory. Customer has to compile and
          either copy the binary to /opt/ibm/csm/hcdiag/tests/nsdperf
          directory or update the script indicating the path.



ARGUMENTS
      -s NSERVERS   Number of servers (remaining will be clients). Default 1
      -i FILE       nsdperf test input file. Default ./nsdperf-test.input
      -p PORT       TCP port to use (default 6668)
      -r RDMAPORTS  RDMA devices and ports to use.Default first device,
                    port 1
      -t NRCV       Number of receiver threads. Default nCPUs, min 2
      -w NWORKERS   Number of message worker threads. Default 32
      -6            Use IPv6 rather than IPv4
      -d            Include debug output
      -h            Print help message and exit


RETURNS
     Returns PASS the all the tests meet the thresholds.
```

```
CONFIGURATION FILE
   /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.nsdperf]
      description = network performance test
      group       = performance
      timeout     = 300
      targetType  = Management
      executable  = /opt/ibm/csm/hcdiag/tests/nsdperf/nsdperf.sh


   /opt/ibm/csm/hcdiag/tests/nsdperf/nsdper.sh
      thisdir=`pwd`
      # The definitions here should be customized
      NSDPERF=${thisdir}/nsdperf


   /opt/ibm/csm/hcdiag/tests/nsdperf/nsdperf-test.input
      status
      version                    # Show program version
      #debug [LEVEL]             # Set debugging output level
      #buffsize NBYTES           # Set I/O data buffer size for tests
      #socksize NBYTES           # Set size of TCP send/receive buffer space
      #threads NTHREADS          # Set number of tester threads to use on clients.
      Default 4
      #parallel N                # Set number of parallel socket connections
      (default 1)
      #rdma [on|off|all|inline  ] # Enable or disable RDMA for sending data blocks
      #maxrdma N                 # Set maximum number of RDMA ports to use per node
      #usecm [on|off]            # Use Connection Manager to establish RDMA
      connections
      #sinline [on|off]          # Use inline data in RDMA send
      #verify [on|off]           # Verify that contents of data messages are correct
      #ttime        120          # Set run time (in seconds) for tests
      test nwrite                # Same as write test, but use NSD-style writing
      (default)
      test read                  # Clients read round-robin from all servers
      (default)
      test sread                 # Each tester thread reads from only one server
      test rw                    # Half of the tester threads read and half write
      # test write               # Clients write round-robin to all servers
      # test swrite              # Each tester thread writes to only one server
      # threshold should follow the same order as the test.
      threshold_nwrite     99
      threshold_read      110
      threshold_sread     110
      threshold_rw        100
      #threshold_write    100
      #threshold_swrite   100
      killall
      quit
```

### *6.2.79 nvvs*

```
NAME
    nvvs -NVIDIA validation suite


SYNOPSIS
    nvvs.sh {level]

INVOCATION
    hcdiag_run.py -test nvvs [-target <noderange>]


DESCRIPTION
      The NVIDIA Validation Suite (NVVS) is the system administrator and
      cluster manager's tool for detecting and troubleshooting common
      problems affecting NVIDIA® Tesla™ GPUs in a high-performance computing
      environment. NVVS focuses on software and system configuration issues,
      diagnostics, topological concerns, and relative performance.

      It requires NVIDIA datacenter-bpu-manager package installed.


ARGUMENTS
      level  1|2|3 translates to nvvs short|medium|long test. Default is 3.
             1: quick system validation (~seconds)
             2: extended system validation (~2min)
             3: system HW diagnostic (~15min)

RETURNS
     Returns PASS if all the nvvs tests pass. Warnings (WARN) is also
     considered PASS.

CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.nvvs]
      description = NVVS (NVIDIA Validation Suite).
      group       = gpu
      timeout     = 300
      targetType  = Compute
      executable  = /opt/ibm/csm/hcdiag/tests/nvvs/nvvs.sh
      args        = 3
```

### *6.2.80 p2pBandwidthLatencyTest*

```
NAME
    p2pBandwidthLatencyTest - NVIDIA GPU bandwidth test


SYNOPSIS
    p2pBandwidthLatencyTest.sh [verbose]


INVOCATION
    hcdiag_run.py -test p2pBandwidthLatencyTest [-target <noderange>]


DESCRIPTION
      This application demonstrates the CUDA Peer-To-Peer (P2P) data
      transfers between pairs of GPUs and computes latency and bandwidth.
      Tests on GPU pairs using P2P and without P2P are tested.

      This test is distributed by NVIDIA as samples, source code only.
      Diagnostic expects the executable to be in /opt/ibm/csm/hcdiag/tests/
      p2pBandwidthLatencyTest directory. If not, p2pBandwidthLatencyTest.sh
      must be updated.

      The cuda-samples package should be installed.


ARGUMENTS
      verbose    0|1 : set the verbose off|on. Default is off.

RETURNS
     Returns PASS if all the results meet the thresholds.


CONFIGURATION FILE
   /opt/ibm/csm/hcdiag/etc/test.properties
     [tests.p2pBandwidthLatencyTest]
      description = parallel ping from nodes to other nodes in the cluster
      executable  = /opt/ibm/csm/hcdiag/tests/p2pBandwidthLatencyTest/
                    p2pBandwidthLatencyTest.sh
      timeout     = 200
      group       = gpu


  # set the path for the p2pBandwidthLatencyTest binary, if
  # not in /opt/ibm/csm/hcdiag/p2pBandwidthLatencyTest
  /opt/ibm/csm/hcdiag/tests/p2pBandwidthLatencyTest/
  p2pBandwidthLatencyTest.sh
      thisdir=`dirname $0`
      LATENCY_TEST=$thisdir/p2pBandwidthLatencyTest
```

```
# The script is configured to support machines with 4 GPUs, and 6 GPUs.
# To change the defaults provide by IBM
/opt/ibm/csm/hcdiag/tests/p2pBandwidthLatencyTest/
p2pBandwidthLatencyTest.sh
    # default values for 4 GPUs.
      UN_GPU_LOCAL=700
      UN_GPU_GPU=48
      UN_GPU_PPC=24
      BI_GPU_LOCAL=700
      BI_GPU_GPU=96
      BI_GPU_PPC=24

    #default values for 6 GPUs.
       UN_GPU_LOCAL=730
       UN_GPU_GPU=45
       UN_GPU_PPC=26
       BI_GPU_LOCAL=750
       BI_GPU_GPU=91
       BI_GPU_PPC=22
```

### *6.2.81 ppping*

```
NAME
    ppping - parallel ping from nodes to other nodes in the cluster.


SYNOPSIS
    ppping.sh <args>


INVOCATION
    hcdiag_run.py -test ppping -target <noderange>


DESCRIPTION
      Determines whether the configured IPoIB devices are configured
      correctly and reply to local ping requests. All given nodes ping each
      other, showing that the nodes are reachable.

      It issues the command /usr/xcat/bin/ppping command.

      NOTE: This test only runs in Management mode.
            User running this test has to have xCAT privileges.
            The xCAT-client package is required.

ARGUMENTS
      args
        ppping command arguments: [-i|--interface interfaces]
                                  [-d|--debug] [-V|--verbose] [-q|--quiet]
                                  [-s|--serial]
       -s   ping serially instead of in parallel
       -i   interfaces: an comma separated list of network interface names
            that should be pinged instead of the interface represented by
            nodename/hostname. The following name resolution convention is
            assumed: an interface is reachable by the hostname <nodename>-
            <interface>. For example, the ib2 interface on node3 has a
            hostname of node3-ib2.
            If more than one interface is specified, each interface will be
            combined with the nodenames as described above and will be pinged
             in turn.
       -d   print debug information
       -h   show usage invormation
       -v   display the installed version of xCAT
       -q   display minimum output

      noderange
            One of more nodes, in XCAT noderange format.


RETURNS
     Returns PASS if no links report problem (noping).


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.ppping]
```

```
description = parallel ping from nodes to other nodes in the cluster
executable  = /opt/ibm/csm/hcdiag/tests/ppping/ppping.sh
targetType  = Management
timeout     = 200
group       = ib
args        = -i ib0,ib1
```

### *6.2.82 rpower*

```
NAME
    rpower - check rpower status


SYNOPSIS
    rpower.sh [-v] -h] <noderange>


INVOCATION
    hcdiag_run.py -test rpower -target <noderange>



DESCRIPTION
     It issues the command /opt/xcat/bin/rpower <noderange> status and
     check if all the nodes in the <noderange> is "on".

     NOTE: This test only runs in Management mode.
           User running this test has to have xCAT privileges.
           The xCAT-client package is required


ARGUMENTS
     -v|--verbose  Set the verbose mode.

     -h|--help     Display the help message.

     noderange     One of more nodes, in XCAT noderange format.


RETURNS
     Returns PASS if rpower returns "on" to all nodes in the <noderange>.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.rpower]
      description = check rpower status
      executable  = /opt/ibm/csm/hcdiag/tests/rpower/rpower.sh
      timeout     = 100
      targetType  = Management
      xcat_cmd    = yes
      #args        = -v
```

### *6.2.83 rvitals*

```
NAME
    rvitals -  check if rvitals results match the specification


SYNOPSIS
    rvitals.sh [-v] [-h] <noderange>


INVOCATION
    hcdiag_run.py -test rvitals -target <noderange>



DESCRIPTION
     It issues the command /opt/xcat/bin/rvitals to get the vital data
     from the nodes and matches the specificagtion in clustconf.yaml.
     Vital data is hardware related data: temperature, voltage,
     frequency, etc.

     I can run concurrently with user jobs.

     NOTE: This test runs in Management mode only.
           User running this test has to have xCAT privileges.
           The xCAT-client package is required


ARGUMENTS
     -v|--verbose  Set the verbose mode.

     -h|--help     Display the help message.

     noderange     One of more nodes, in XCAT noderange format.


RETURNS
     Returns PASS if rvitals  data of all the nodes match the specification.


CONFIGURATION FILE
   /opt/ibm/csm/hcdiag/etc/test.properties
      [tests.rvitals]
      description = check if rvitals match the spec
      executable  = /opt/ibm/csm/hcdiag/tests/rvitals/rvitals.sh
      timeout     = 300
      targetType  = Management
      xcat_cmd    = yes
      #args        = -v


   /opt/ibm/csm/hcdiag/etc/clustconf.yaml
      wspoon_dd2:
          - {id: 'Ambient', regex: (\S+), range: [10,40] }
          - {id: 'Fan1 \d',  regex: (\S+), range: [0,24000] }
```

```
- {id: 'Fan[0-3] \d',  regex: (\S+), range: [2500,14000] }
- {id: 'P\d Vcs Temp',  regex: (\S+), range: [15,80]}
- {id: 'P\d Vdd Temp',  regex: (\S+), range: [15,80]}
- {id: 'P\d Vddr Temp',  regex: (\S+), range: [15,80]}
- {id: 'P\d Vdn Temp',  regex: (\S+), range: [15,80]}
- {id: 'Ambient', regex: (\S+), range: [10,40] }
- {id: 'DIMM\d+ Temp', regex: (\S+), range: [15,75,N/A]}
- {id: 'P\d Core6 Temp',    regex: (\S+), range: [0,90,N/A]}
 - {id: 'P\d Core7 Temp',    regex: (\S+), range: [0,90,N/A]}

- {id: 'P\d Core\d Temp',    regex: (\S+), range: [10,90,N/A]}
- {id: 'P\d GPU Power',  regex: (\S+), range: [10,1800,N/A] }
- {id: 'P\d Io Power',  regex: (\S+), range: [10,500,N/A] }
- {id: 'P\d Mem Power',  regex: (\S+), range: [10,500,N/A] }
- {id: 'P\d Power',  regex: (\S+), range: [1,1800,N/A] }
- {id: 'Ps\d Input Power',  regex: (\S+), range: [10,1800,N/A] }
- {id: 'Ps\d Input Voltage',  regex: (\S+), range: [200,285,N/A] }
- {id: 'Ps1 Output Current',  regex: (\S+), range: [0,100,N/A] }
- {id: 'Ps\d Output Voltage',  regex: (\S+), range: [10,400,0] }
- {id: 'Ps\d Output Current',  regex: (\S+), range: [10,100,N/A] }
- {id: 'Ps\d Output Voltage',  regex: (\S+), range: [300,400,N/A] }
- {id: 'Storage A Power',  regex: (\S+), range: [10,500,N/A] }
- {id: 'Storage B Power',  regex: (\S+), range: [10,500,N/A] }
- {id: 'Total Power',  regex: (\S+), range: [10,2000,N/A] }
```

### *6.2.84 swhealth*

```
NAME
    swhealth – run switch health check report


SYNOPSIS
    swhealth.sh [-v] [-h]


INVOCATION
    hcdiag_run.py –test swhealth [ –target <noderange> ]


DESCRIPTION
     This test runs a switch health report. It uses values configured in the
clustconf.yaml file to query the UFM and generates a health report for each
switch by passing relevant information to
http://[ufm_ip]/ufmRest/actions/provisioning/Show-Health-Report.

It can run concurrently with user jobs.


ARGUMENTS
     -v   Set the verbose mode.

     -h   Display the help message.


RETURNS
     Returns PASS if there are no errors in the switch health report.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
       [tests.swhealth]
       description = Run switch health check report
       executable  = /opt/ibm/csm/hcdiag/tests/swhealth/swhealth.sh
       timeout     = 60

    /opt/ibm/csm/hcdiag/etc/clustconf.yaml
       ufm:
         ip_address: "10.7.0.41"
         user: "admin"
         password: "123456
```

### *6.2.85 test-simple*

```
NAME
    test-simple – simple hello test


SYNOPSIS
    test-simple.sh


INVOCATION
    hcdiag_run.py –test test-simple [–target <noderange>]


DESCRIPTION
     It is a simple test to validate the installation and the
     Diagnostic’s user setup.

     It can run concurrently with user jobs.


ARGUMENTS
     None

RETURNS
     Always return PASS.


CONFIGURATION FILE
    /opt/ibm/csm/hcdiag/etc/test.properties
        [tests.test-simple]
        description = simple hello test
        executable  = /opt/ibm/csm/hcdiag/tests/test-simple/test-simple.sh
        timeout     = 1
```

# 7. Appendix C: Running scenarios

To demonstrate the Diagnostics framework, we run the following different scenarios.

- Standalone mode: with and without CSM, Management mode and Node mode
- From a Job prolog/epilog
- In an RAS event handler

We used the development cluster with the following machines:

- c650mnp01: management node
- c650f02p07: utility node
- c650f02p[09,11,13,15,17,19,21,23,25,27]: compute nodes

We used the Diagnostic default properties file installed at /opt/ibm/csm/hcdiag/etc directory.
Default values are: csm=yes (with allocation), logdir=/tmp
Also the Diagnostic query tool was used in all scenarios to show its functionality.

## 7.1 Management Mode

### 7.1.1 *Running Diagnostic with CSM*

Only the mandatory arguments were provided: test and target.

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test test_memsize hxecache --target c650f02p1[1,3]
INFO: xcat seems to be installed in /opt/xcat/bin. Running in Management mode
Health Check Diagnostics version beta2-2017, running on Linux 3.10.0-514.el7.ppc64le, c650mnp01
machine.
Using configuration file /opt/ibm/csm/hcdiag/etc/hcdiag.properties.
Using tests configuration file /opt/ibm/csm/hcdiag/etc/test.properties.
Health Check Diagnostics, run id 170731220443804601, initializing...
Validating command argument target.
Validating command argument test.
Allocation request successful, id= 259
Preparing to run test_memsize.
Executable: /opt/ibm/csm/hcdiag/tests/test_memsize/test_memsize.sh exists on remote node(s).
test_memsize started on 2 node(s) at 2017-07-31 22:04:45.651563. It might take up to 10s.

test_memsize ended on 2 node(s) at 2017-07-31 22:04:46.736125, rc= 0, elapsed time:
0:00:01.084562
test_memsize PASS on node c650f02p11, serial number: 78165BA.
test_memsize PASS on node c650f02p13, serial number: 781660A.
Preparing to run hxecache.
Executable: /opt/ibm/csm/hcdiag/tests/hxecache/hxecache.sh exists on remote node(s).
hxecache started on 2 node(s) at 2017-07-31 22:04:46.767420. It might take up to 800s.
.....................................................................
hxecache ended on 2 node(s) at 2017-07-31 22:07:15.557682, rc= 0, elapsed time: 0:02:28.790262
hxecache PASS on node c650f02p11, serial number: 78165BA.
hxecache PASS on node c650f02p13, serial number: 781660A.
Request csmd to release the allocation 259.
Allocation 259 delete request successful.
Health Check Diagnostics ended, exit code 0.
```

Running with CSM environment, run and results are stored in the database and an allocation was created for the run.

```
$ ls -oR /tmp/*170731220443804601* |grep -v total
```

```
-rw-r--r-- 1 diagadmin 24149 Jul 31 22:07 /tmp/hcdiag_run-170731220443804601.log

/tmp/170731220443804601:
drwxr-xr-x 2 diagadmin 96 Jul 31 22:04 hxecache
drwxr-xr-x 2 diagadmin 96 Jul 31 22:04 test_memsize

/tmp/170731220443804601/hxecache:
-rw-r--r-- 1 diagadmin 3426 Jul 31 22:07 c650f02p11-2017-07-31-22_04_47.output
-rw-r--r-- 1 diagadmin 3426 Jul 31 22:07 c650f02p13-2017-07-31-22_04_47.output

/tmp/170731220443804601/test_memsize:
-rw-r--r-- 1 diagadmin 106 Jul 31 22:04 c650f02p11-2017-07-31-22_04_46.output
-rw-r--r-- 1 diagadmin 106 Jul 31 22:04 c650f02p13-2017-07-31-22_04_46.output
```

Below the contents of the database:

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_query.py --runid 170731220443804601 --results
/opt/ibm/csm/hcdiag/bin/hcdiag_query.py, version beta2, running on Linux 3.10.0-
514.el7.ppc64le, c650mnp01 machine.
run_id was specified, all other arguments will be ignored
---
runData:
  history_time:     2017-07-31 22:07:15.607746
  run_id:           170731220443804601
  allocation_id:    259
  begin_time:       2017-07-31 22:04:45.649329
  end_time:         2017-07-31 22:07:15.607746
  diag_status:      COMPLETED
  inserted_ras:     f
  log_dir:          /tmp/170731220443804601
  cmd_line:         /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test test_memsize hxecache --target
c650f02p1[1,3]
Total_Result_Records: 4
Result_Record_1:
  history_time:     2017-07-31 22:07:15.607746
  run_id:           170731220443804601
  test_name:        test_memsize
  node_name:        c650f02p11
  serial_number:    78165BA
  begin_time:       2017-07-31 22:04:45.651563
  end_time:         2017-07-31 22:04:46.75597
  status:           PASS
  log_file:         /tmp/170731220443804601/test_memsize/c650f02p11-2017-07-31-22_04_46.output
Result_Record_2:
  history_time:     2017-07-31 22:07:15.607746
  run_id:           170731220443804601
  test_name:        test_memsize
  node_name:        c650f02p13
  serial_number:    781660A
  begin_time:       2017-07-31 22:04:45.651563
  end_time:         2017-07-31 22:04:46.765146
  status:           PASS
  log_file:         /tmp/170731220443804601/test_memsize/c650f02p13-2017-07-31-22_04_46.output
Result_Record_3:
  history_time:     2017-07-31 22:07:15.607746
  run_id:           170731220443804601
  test_name:        hxecache
  node_name:        c650f02p11
  serial_number:    78165BA
  begin_time:       2017-07-31 22:04:46.76742
  end_time:         2017-07-31 22:07:15.576145
  status:           PASS
  log_file:         /tmp/170731220443804601/hxecache/c650f02p11-2017-07-31-22_04_47.output
Result_Record_4:
  history_time:     2017-07-31 22:07:15.607746
  run_id:           170731220443804601
  test_name:        hxecache
  node_name:        c650f02p13
  serial_number:    781660A
```

```
  begin_time:       2017-07-31 22:04:46.76742
  end_time:         2017-07-31 22:07:15.585256
  status:           PASS
  log_file:         /tmp/170731220443804601/hxecache/c650f02p13-2017-07-31-22_04_47.output
...
Done
```

### 7.1.2 *Running Diagnostic with CSM and no allocation*

Since performing the allocation is the default behavior when running with CSM environment, we pass the option *–noallocation.*

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test test_memsize hxecache --target c650f02p1[1,3] --
noallocation
INFO: xcat seems to be installed in /opt/xcat/bin. Running in Management mode
Health Check Diagnostics version beta2-2017, running on Linux 3.10.0-514.el7.ppc64le, c650mnp01
machine.
Using configuration file /opt/ibm/csm/hcdiag/etc/hcdiag.properties.
Using tests configuration file /opt/ibm/csm/hcdiag/etc/test.properties.
Health Check Diagnostics, run id 170731221031585077, initializing...
Validating command argument target.
Validating command argument test.
Preparing to run test_memsize.
Executable: /opt/ibm/csm/hcdiag/tests/test_memsize/test_memsize.sh exists on remote node(s).
test_memsize started on 2 node(s) at 2017-07-31 22:10:33.423081. It might take up to 10s.

test_memsize ended on 2 node(s) at 2017-07-31 22:10:34.516066, rc= 0, elapsed time:
0:00:01.092985
test_memsize PASS on node c650f02p11, serial number: 78165BA.
test_memsize PASS on node c650f02p13, serial number: 781660A.
Preparing to run hxecache.
Executable: /opt/ibm/csm/hcdiag/tests/hxecache/hxecache.sh exists on remote node(s).
hxecache started on 2 node(s) at 2017-07-31 22:10:34.548611. It might take up to 800s.
.....................................................................
hxecache ended on 2 node(s) at 2017-07-31 22:13:15.055093, rc= 0, elapsed time: 0:02:40.506482
hxecache PASS on node c650f02p11, serial number: 78165BA.
hxecache PASS on node c650f02p13, serial number: 781660A.
Health Check Diagnostics ended, exit code 0.
```

Below the files/directories created in *logdir*.

```
$ ls -oR /tmp/*170731221031585077* |grep -v total
-rw-r--r-- 1 diagadmin 21838 Jul 31 22:13 /tmp/hcdiag_run-170731221031585077.log

/tmp/170731221031585077:
drwxr-xr-x 2 diagadmin 96 Jul 31 22:10 hxecache
drwxr-xr-x 2 diagadmin 96 Jul 31 22:10 test_memsize

/tmp/170731221031585077/hxecache:
-rw-r--r-- 1 diagadmin 3152 Jul 31 22:13 c650f02p11-2017-07-31-22_10_35.output
-rw-r--r-- 1 diagadmin 3134 Jul 31 22:12 c650f02p13-2017-07-31-22_10_35.output

/tmp/170731221031585077/test_memsize:
-rw-r--r-- 1 diagadmin 106 Jul 31 22:10 c650f02p11-2017-07-31-22_10_33.output
-rw-r--r-- 1 diagadmin 106 Jul 31 22:10 c650f02p13-2017-07-31-22_10_33.output
```

Diagnostic run and results are stored in the database. *allocation_id* is set to zero, indicating that allocation was not created by Diagnostic.

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_query.py --runid 170731221031585077 --results
/opt/ibm/csm/hcdiag/bin/hcdiag_query.py, version beta2, running on Linux 3.10.0-
514.el7.ppc64le, c650mnp01 machine.
run_id was specified, all other arguments will be ignored
---
runData:
  history_time:     2017-07-31 22:13:15.100185
  run_id:           170731221031585077
  allocation_id:    0
  begin_time:       2017-07-31 22:10:33.420726
  end_time:         2017-07-31 22:13:15.100185
  diag_status:      COMPLETED
  inserted_ras:     f
  log_dir:          /tmp/170731221031585077
  cmd_line:         /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test test_memsize hxecache --target
c650f02p1[1,3] --noallocation
Total_Result_Records: 4
Result_Record_1:
  history_time:     2017-07-31 22:13:15.100185
  run_id:           170731221031585077
  test_name:        test_memsize
  node_name:        c650f02p11
  serial_number:    78165BA
  begin_time:       2017-07-31 22:10:33.423081
  end_time:         2017-07-31 22:10:34.537493
  status:           PASS
  log_file:         /tmp/170731221031585077/test_memsize/c650f02p11-2017-07-31-22_10_33.output
Result_Record_2:
  history_time:     2017-07-31 22:13:15.100185
  run_id:           170731221031585077
  test_name:        test_memsize
  node_name:        c650f02p13
  serial_number:    781660A
  begin_time:       2017-07-31 22:10:33.423081
  end_time:         2017-07-31 22:10:34.546514
  status:           PASS
  log_file:         /tmp/170731221031585077/test_memsize/c650f02p13-2017-07-31-22_10_33.output
Result_Record_3:
  history_time:     2017-07-31 22:13:15.100185
  run_id:           170731221031585077
  test_name:        hxecache
  node_name:        c650f02p11
  serial_number:    78165BA
  begin_time:       2017-07-31 22:10:34.548611
  end_time:         2017-07-31 22:13:15.082604
  status:           PASS
  log_file:         /tmp/170731221031585077/hxecache/c650f02p11-2017-07-31-22_10_35.output
Result_Record_4:
  history_time:     2017-07-31 22:13:15.100185
  run_id:           170731221031585077
  test_name:        hxecache
  node_name:        c650f02p13
  serial_number:    781660A
  begin_time:       2017-07-31 22:10:34.548611
  end_time:         2017-07-31 22:13:15.091979
  status:           PASS
  log_file:         /tmp/170731221031585077/hxecache/c650f02p13-2017-07-31-22_10_35.output
...
Done
```

### 7.1.3 *Running Diagnostic without CSM*

*--nocsm* argument explicitly requests Diagnostic to not use CSM environment. As a result, a summary of the run is displayed on the screen and saved in the file *<logdir>/runid/result_summay*.

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test test_memsize hxecache --target c650f02p1[1,3] --
nocsm
INFO: xcat seems to be installed in /opt/xcat/bin. Running in Management mode
Health Check Diagnostics version beta2-2017, running on Linux 3.10.0-514.el7.ppc64le, c650mnp01
machine.
Using configuration file /opt/ibm/csm/hcdiag/etc/hcdiag.properties.
Using tests configuration file /opt/ibm/csm/hcdiag/etc/test.properties.
Health Check Diagnostics, run id 170731221720128668, initializing...
Validating command argument target.
Validating command argument test.
Preparing to run test_memsize.
Executable: /opt/ibm/csm/hcdiag/tests/test_memsize/test_memsize.sh exists on remote node(s).
test_memsize started on 2 node(s) at 2017-07-31 22:17:21.926255. It might take up to 10s.

test_memsize ended on 2 node(s) at 2017-07-31 22:17:22.996286, rc= 0, elapsed time:
0:00:01.070031
test_memsize PASS on node c650f02p11, serial number: 78165BA.
test_memsize PASS on node c650f02p13, serial number: 781660A.
Preparing to run hxecache.
Executable: /opt/ibm/csm/hcdiag/tests/hxecache/hxecache.sh exists on remote node(s).
hxecache started on 2 node(s) at 2017-07-31 22:17:22.997069. It might take up to 800s.
.....................................................................................
hxecache ended on 2 node(s) at 2017-07-31 22:20:01.834361, rc= 0, elapsed time: 0:02:38.837292
hxecache PASS on node c650f02p11, serial number: 78165BA.
hxecache PASS on node c650f02p13, serial number: 781660A.

============================== Results summary  ==============================

22:17:21 ================================================================================

test_memsize PASS on 2 node(s):

c650f02p11,c650f02p13


22:17:22 ================================================================================

hxecache PASS on 2 node(s):

c650f02p11,c650f02p13


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

Health Check Diagnostics ended, exit code 0.
```

*result_summary* is stored in *logdir* in addition to the log and output files.

```
$ ls -oR /tmp/*170731221720128668* |grep -v total
-rw-r--r-- 1 diagadmin 20449 Jul 31 22:20 /tmp/hcdiag_run-170731221720128668.log

/tmp/170731221720128668:
drwxr-xr-x 2 diagadmin  96 Jul 31 22:17 hxecache
-rw-r--r-- 1 diagadmin 442 Jul 31 22:20 result_summary
drwxr-xr-x 2 diagadmin  96 Jul 31 22:17 test_memsize

/tmp/170731221720128668/hxecache:
-rw-r--r-- 1 diagadmin 3152 Jul 31 22:20 c650f02p11-2017-07-31-22_17_23.output
-rw-r--r-- 1 diagadmin 3134 Jul 31 22:19 c650f02p13-2017-07-31-22_17_23.output

/tmp/170731221720128668/test_memsize:
```

```
-rw-r--r-- 1 diagadmin 106 Jul 31 22:17 c650f02p11-2017-07-31-22_17_22.output
-rw-r--r-- 1 diagadmin 106 Jul 31 22:17 c650f02p13-2017-07-31-22_17_22.output
```

Since CSM was not used, there are no records in the database.

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_query.py --runid 170731221720128668 --results
/opt/ibm/csm/hcdiag/bin/hcdiag_query.py, version beta2, running on Linux 3.10.0-
514.el7.ppc64le, c650mnp01 machine.
run_id was specified, all other arguments will be ignored
---
Total_Records: 0
# No matching records found.
...
done
```

### 7.1.4 *Running Diagnostics in RAS event handling*

Using the same environment used to demonstrate the *RAS Integration*, we changed the *diag_node_test_fail* script to:

- check if the test failed is *test_memsize*
- if it is *test_memsize*, invoke Diagnostic test *rinv_memory.*
  *rinv_memory* checks the inventory (via xcat rinv command) and prints the attributes related to the node's memory.

Below the new content of the *hcdiag.test.fail*'s control action, *diag_node_test_fail.*

```
$ cat /opt/ibm/csm/ras/actions/diag_node_test_fail
#! /bin/bash
readonly ACTION=$0

echo "Running $ACTION"
message=`echo $1 | sed -e 's/^.*"message"[ ]*:[ ]*"//' -e 's/".*//'`
test=`echo $message | awk '{print $1}'`
node=`echo $message | awk '{print $5}'`

if [ "$test" == "test_memsize" ]; then
  newtest=rinv_memory
fi

/opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test $newtest --target $node
rc=$?

echo "Diagnostic test $newtest finished, rc= $rc"
```

Executing Diagnostic to create a test failure:

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test test_memsize --target c650f02p11
INFO: xcat seems to be installed in /opt/xcat/bin. Running in Management mode
Health Check Diagnostics version beta2-2017, running on Linux 3.10.0-514.el7.ppc64le, c650mnp01
machine.
Using configuration file /opt/ibm/csm/hcdiag/etc/hcdiag.properties.
Using tests configuration file /opt/ibm/csm/hcdiag/etc/test.properties.
Health Check Diagnostics, run id 170802195040902807, initializing...
Validating command argument target.
Validating command argument test.
Allocation request successful, id= 945
Preparing to run test_memsize.
Executable: /opt/ibm/csm/hcdiag/tests/test_memsize/test_memsize.sh exists on remote node(s).
test_memsize started on 1 node(s) at 2017-08-02 19:50:42.714649. It might take up to 10s.
```

```
test_memsize ended on 1 node(s) at 2017-08-02 19:50:43.786056, rc= 1, elapsed time:
0:00:01.071407
test_memsize FAIL on node c650f02p11, serial number: 78165BA, rc= 1. (details in
/tmp/170802195040902807/test_memsize/c650f02p11-2017-08-02-19_50_43.output)
A RAS event was created in the database for this message
Request csmd to release the allocation 945.
Allocation 945 delete request successful.
Health Check Diagnostics ended, exit code 100.
```

Checking the output of the control action being executed, */var/log/ibm/csm/ras/actions* directory:

```
$ sudo cat /var/log/ibm/csm/ras/actions/diag_node_test_fail.log
[2017-08-02 19:50:43] Running /opt/ibm/csm/ras/actions/diag_node_test_fail
[2017-08-02 19:50:43] Health Check Diagnostics version beta2-2017, running on Linux 3.10.0-
514.el7.ppc64le, c650mnp01 machine.
[2017-08-02 19:50:43] Using configuration file /opt/ibm/csm/hcdiag/etc/hcdiag.properties.
[2017-08-02 19:50:43] Using tests configuration file /opt/ibm/csm/hcdiag/etc/test.properties.
[2017-08-02 19:50:43] Health Check Diagnostics, run id 170802195043850232, initializing...
[2017-08-02 19:50:43] Validating command argument target.
[2017-08-02 19:50:44] Validating command argument test.
[2017-08-02 19:50:44] Allocation request successful, id= 946
[2017-08-02 19:50:44] Preparing to run rinv_memory.
[2017-08-02 19:50:44] rinv_memory started on 1 node(s) at 2017-08-02 19:50:44.680936. It might
take up to 10s.
[2017-08-02 19:50:46] INFO: xcat seems to be installed in /opt/xcat/bin. Running in Management
mode
.[2017-08-02 19:50:52] rinv_memory ended on 1 node(s) at 2017-08-02 19:50:52.673101, rc= 0,
elapsed time: 0:00:07.992165
[2017-08-02 19:50:52] rinv_memory PASS on node c650mnp01, serial number: 214D6AA.
[2017-08-02 19:50:52] Request csmd to release the allocation 946.
[2017-08-02 19:50:52] Allocation 946 delete request successful.
[2017-08-02 19:50:52] Health Check Diagnostics ended, exit code 0.
[2017-08-02 19:50:52] Diagnostic test rinv_memory finished, rc= 0
$
```

## 7.2 Node mode

### 7.2.1 *Running Diagnostic with CSM*

Diagnostic is not allowed to run in Node mode, with allocation request.

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test test_memsize hxecache
INFO: running in Node mode.
hcdiag_run : error: when running in Node mode, allocation cannot be requested. Use --
noallocation option.
```

### 7.2.2 *Running Diagnostic with CSM and no allocation*

Since performing the allocation is the default behavior when running with CSM environment, we pass the option *–noallocation.*

```
$  /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test test_memsize hxecache --noallocation
INFO: running in Node mode.
Info: tests.ppping ignored, can not run in Node mode.
Info: bucket.ib ignored, can not run in Node mode.
Info: bucket.beta1 ignored, can not run in Node mode.
Health Check Diagnostics version beta2-2017, running on Linux 3.10.0-514.el7.ppc64le,
c650f02p11 machine.
Using configuration file /opt/ibm/csm/hcdiag/etc/hcdiag.properties.
XCAT fanout ignored, allocation disabled.
```

```
Using tests configuration file /opt/ibm/csm/hcdiag/etc/test.properties.
Health Check Diagnostics, run id 170731223048686245, initializing...
Validating command argument target.
Validating command argument test.
Preparing to run test_memsize.
Executable: /opt/ibm/csm/hcdiag/tests/test_memsize/test_memsize.sh exists on target node(s).
test_memsize started on 1 node(s) at 2017-07-31 22:30:48.955705. It might take up to 10s.

test_memsize ended on 1 node(s) at 2017-07-31 22:30:49.012559, rc= 0, elapsed time:
0:00:00.056854
test_memsize PASS on node c650f02p11, serial number: 78165BA.
Preparing to run hxecache.
Executable: /opt/ibm/csm/hcdiag/tests/hxecache/hxecache.sh exists on target node(s).
hxecache started on 1 node(s) at 2017-07-31 22:30:49.090724. It might take up to 800s.
.....................................................................
hxecache ended on 1 node(s) at 2017-07-31 22:33:29.778613, rc= 0, elapsed time: 0:02:40.687889
hxecache PASS on node c650f02p11, serial number: 78165BA.
Health Check Diagnostics ended, exit code 0.
```

Below the contents of *logdir*:

```
$ ls -oR /tmp/*170731223048686245* | grep -v total
-rw-r--r-- 1 diagadmin 3739 Jul 31 22:33 /tmp/hcdiag_run-170731223048686245.log

/tmp/170731223048686245:
drwxr-xr-x 2 diagadmin 51 Jul 31 22:30 hxecache
drwxr-xr-x 2 diagadmin 51 Jul 31 22:30 test_memsize

/tmp/170731223048686245/hxecache:
-rw-r--r-- 1 diagadmin 3062 Jul 31 22:33 c650f02p11-2017-07-31-22_30_49.output

/tmp/170731223048686245/test_memsize:
-rw-r--r-- 1 diagadmin 81 Jul 31 22:30 c650f02p11-2017-07-31-22_30_48.output
```

Diagnostic run and results are stored in the database. *allocation_id* is set to zero, indicating that allocation was not created by Diagnostic.

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_query.py --runid 170731223048686245 --results
/opt/ibm/csm/hcdiag/bin/hcdiag_query.py, version beta2, running on Linux 3.10.0-
514.el7.ppc64le, c650f02p11 machine.
run_id was specified, all other arguments will be ignored
---
runData:
  history_time:     2017-07-31 22:33:35.93547
  run_id:           170731223048686245
  allocation_id:    0
  begin_time:       2017-07-31 22:30:54.946238
  end_time:         2017-07-31 22:33:35.93547
  diag_status:      COMPLETED
  inserted_ras:     f
  log_dir:          /tmp/170731223048686245
  cmd_line:         /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test test_memsize hxecache --
noallocation
Total_Result_Records: 2
Result_Record_1:
  history_time:     2017-07-31 22:33:35.93547
  run_id:           170731223048686245
  test_name:        test_memsize
  node_name:        c650f02p11
  serial_number:    78165BA
  begin_time:       2017-07-31 22:30:48.955705
  end_time:         2017-07-31 22:30:55.085606
  status:           PASS
  log_file:         /tmp/170731223048686245/test_memsize/c650f02p11-2017-07-31-22_30_48.output
Result_Record_2:
  history_time:     2017-07-31 22:33:35.93547
```

```
   run_id:           170731223048686245
   test_name:        hxecache
   node_name:        c650f02p11
   serial_number:    78165BA
   begin_time:       2017-07-31 22:30:49.090724
   end_time:         2017-07-31 22:33:35.846188
   status:           PASS
   log_file:         /tmp/170731223048686245/hxecache/c650f02p11-2017-07-31-22_30_49.output
...
done
```

### 7.2.3 *Running Diagnostic without CSM*

*--nocsm* argument explicitly requests to not use the CSM environment. As a result, a summary of the run is displayed on the screen and saved in the file *<logdir>/runid/result_summay* and there are no entries in the database.

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test test_memsize hxecache --nocsm
INFO: running in Node mode.
Info: tests.ppping ignored, can not run in Node mode.
Info: bucket.ib ignored, can not run in Node mode.
Info: bucket.beta1 ignored, can not run in Node mode.
Health Check Diagnostics version beta2-2017, running on Linux 3.10.0-514.el7.ppc64le,
c650f02p11 machine.
Using configuration file /opt/ibm/csm/hcdiag/etc/hcdiag.properties.
XCAT fanout ignored, allocation disabled.
Using tests configuration file /opt/ibm/csm/hcdiag/etc/test.properties.
Health Check Diagnostics, run id 170731224536503107, initializing...
Validating command argument target.
Validating command argument test.
Preparing to run test_memsize.
Executable: /opt/ibm/csm/hcdiag/tests/test_memsize/test_memsize.sh exists on target node(s).
test_memsize started on 1 node(s) at 2017-07-31 22:45:36.607539. It might take up to 10s.

test_memsize ended on 1 node(s) at 2017-07-31 22:45:36.664333, rc= 0, elapsed time:
0:00:00.056794
test_memsize PASS on node c650f02p11, serial number: 78165BA.
Preparing to run hxecache.
Executable: /opt/ibm/csm/hcdiag/tests/hxecache/hxecache.sh exists on target node(s).
hxecache started on 1 node(s) at 2017-07-31 22:45:36.664644. It might take up to 800s.
.....................................................................
hxecache ended on 1 node(s) at 2017-07-31 22:48:12.685823, rc= 0, elapsed time: 0:02:36.021179
hxecache PASS on node c650f02p11, serial number: 78165BA.

============================== Results summary  ==============================

22:45:36 ========================================================================

test_memsize PASS on 1 node(s):

c650f02p11


22:45:36 ========================================================================

hxecache PASS on 1 node(s):

c650f02p11


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

Health Check Diagnostics ended, exit code 0.
```

*logdir* contains the *result_summary* file in addition to log and output files.

```
$ ls -oR /tmp/*170731224536503107* |grep -v total
-rw-r--r-- 1 diagadmin 2980 Jul 31 22:48 /tmp/hcdiag_run-170731224536503107.log

/tmp/170731224536503107:
drwxr-xr-x 2 diagadmin  51 Jul 31 22:45 hxecache
-rw-r--r-- 1 diagadmin 420 Jul 31 22:48 result_summary
drwxr-xr-x 2 diagadmin  51 Jul 31 22:45 test_memsize

/tmp/170731224536503107/hxecache:
-rw-r--r-- 1 diagadmin 3062 Jul 31 22:48 c650f02p11-2017-07-31-22_45_36.output

/tmp/170731224536503107/test_memsize:
-rw-r--r-- 1 diagadmin 81 Jul 31 22:45 c650f02p11-2017-07-31-22_45_36.output
```

Without CSM, Diagnostic does not populate the database.

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_query.py --runid 170731224536503107
/opt/ibm/csm/hcdiag/bin/hcdiag_query.py, version beta2, running on Linux 3.10.0-
514.el7.ppc64le, c650f02p11 machine.
run_id was specified, all other arguments will be ignored
---
Total_Records: 0
# No matching records found.
...
done
```

### 7.2.4 *Running Diagnostic via job prolog*

To demonstrate a job epilog invoking Diagnostic, we defined the bucket *node_health* in the /opt/ibm/csm/hcdiag/etc/test.properties file as follow:

```
[bucket.node_health]
description = A set of tests that checks the health of the node.
tests       = test_memsize
```

We apply the following patches to the /opt/ibm/csm/share/prologs/privileged_prolog and /opt/ibm/csm/share/prologs/privileged_epilog and copied them to /opt/ibm/csm/prologs.
The patch to the *privileged_epilog* adds the code that invokes the Diagnostics when user flag is set to “DIAG”.

```
26a27
> import subprocess
64c65
<     "long"  : ["UFM-vs","SMT","sms"]
---
>     "long"  : ["UFM-vs","SMT","sms","DIAG"]
124a126,135
>         elif flag == "DIAG":
>             if epilog_type == TYPES['allocation']:
>                 logger.info( "'DIAG' flag has been set." )
>                 rc=execute_diag()
>                 logger.info( "Diagnostic epilog completed, rc=%s." %rc )
>                 return rc
>             else:
>                 logger.error( "'DIAG' flag is not supported for the step type." )
>
>
160a172,177
> def execute_diag():
>     cmd='/opt/ibm/csm/hcdiag/bin/hcdiag_run.py --noallocation --bucket node_health'
```

```
>     proc = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE, shell=True )
>     proc.wait()
>     return proc.returncode
>
```

The patch to the *privileged_prolog* adds the code that only logs a message when user flag is set to “DIAG”.

```
64c64
<     "long"  : ["UFM-vs","SMT","sms"]
---
>     "long"  : ["UFM-vs","SMT","sms","DIAG"]
124a125,130
>         elif flag == "DIAG":
>             if prolog_type == TYPES['allocation']:
>                 logger.info( "'DIAG' flag has been set." )
>             else:
>                 logger.error( "'DIAG' flag is not supported for the step type." )
>
```

Then submitted the following job:

```
#!/bin/bash
# sample bsub script for single script to both submit the job and
# run the job.
#
source /opt/ibm/spectrumcomputing/lsf/conf/profile.lsf

export CSM_USR_FLAGS=DIAG
if [ -z $LSB_BATCH_JID ]; then
    bsub  \
        -R "1*{select[LN]}+20*{select[CN]}"  \
        -m "c650f02p07 c650f02p27" \
        -q csm_queue \
        -o $PWD/log/lsf-%J.out $0
else
    echo PWD=$PWD
    $PWD/hello_world
    echo RC=$?
fi
```

Management node’s  /var/log/messages logs the privileged_prolog and privileged_epilog execution:

```
Aug 11 11:45:09 c650f02p27 csmd: [COMPUTE]2017-08-11 11:45:09 csmapi::INFO | privileged_prolog:
Prolog script begin.
Aug 11 11:45:09 c650f02p27 csmd: [COMPUTE]2017-08-11 11:45:09 csmapi::INFO | privileged_prolog:
'DIAG' flag has been set.
Aug 11 11:45:09 c650f02p27 csmd: [COMPUTE]2017-08-11 11:45:09 csmapi::INFO | privileged_prolog:
Prolog script ended cleanly.
Aug 11 11:45:18 c650f02p27 csmd: [COMPUTE]2017-08-11 11:45:18 csmapi::INFO | privileged_epilog:
Epilog script begin.
Aug 11 11:45:18 c650f02p27 csmd: [COMPUTE]2017-08-11 11:45:18 csmapi::INFO | privileged_epilog:
'DIAG' flag has been set.
Aug 11 11:45:18 c650f02p27 systemd: Started Session 668 of user root.
Aug 11 11:45:18 c650f02p27 systemd: Starting Session 668 of user root.
Aug 11 11:45:19 c650f02p27 csmd: [COMPUTE]2017-08-11 11:45:19 csmapi::INFO | privileged_epilog:
Diagnostic epilog completed, rc=0.
Aug 11 11:45:19 c650f02p27 csmd: [COMPUTE]2017-08-11 11:45:19 csmapi::INFO | privileged_epilog:
Epilog script ended cleanly.
```

Bellow the output of the job:

```
Sender: LSF System <lsfadmin@c650f02p07>
Subject: Job 3095: <./lsf_prolog_epilog.sh> in cluster <c650mnp01> Done

Job <./lsf_prolog_epilog.sh> was submitted from host <c650f02p07> by user <aldas> in cluster
<c650mnp01>.
Job was executed on host(s) <1*c650f02p07>, in queue <csm_queue>, as user <aldas> in cluster
<c650mnp01>.
                            <20*c650f02p27>
</u/aldas> was used as the home directory.
</u/diagadmin/sw45> was used as the working directory.
Started at Results reported on
Your job looked like:

------------------------------------------------------------
# LSBATCH: User input
./lsf_prolog_epilog.sh
------------------------------------------------------------

Successfully completed.

Resource usage summary:

    CPU time :                                   0.05 sec.
    Max Memory :                                 35.56 MB
    Average Memory :                             35.56 MB
    Total Requested Memory :                     -
    Delta Memory :                               -
    Max Swap :                                   -
    Max Processes :                              1
    Max Threads :                                3
    Run time :                                   9 sec.
    Turnaround time :                            0 sec.

The output (if any) follows:

PWD=/u/diagadmin/sw45
hello world! From c650f02p07 :)
RC=0
hostname=c650f02p07
~
~
```

Diagnostic was invoked from allocation delete's *privileged_epilog* on c650f02p27.
The Diagnostic output/log files were created on c650f02p27 node's *logdir*.

```
$ ls -oR /tmp/*170811114518534194*|grep -v total
-rw-r--r-- 1 root 3258 Aug 11 11:45 /tmp/hcdiag_run-170811114518534194.log

/tmp/170811114518534194:
drwxr-xr-x 2 root 51 Aug 11 11:45 test_memsize

/tmp/170811114518534194/test_memsize:
-rw-r--r-- 1 root 81 Aug 11 11:45 c650f02p27-2017-08-11-11_45_18.output
```

And the database contents shows the successful run of the *node_health* bucket.

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_query.py -r 170811114518534194 --results
/opt/ibm/csm/hcdiag/bin/hcdiag_query.py, version beta2, running on Linux 3.10.0-
514.el7.ppc64le, c650f02p27 machine.
```

```
run_id was specified, all other arguments will be ignored
---
runData:
  history_time:     2017-08-11 11:45:20.553478
  run_id:           170811114518534194
  allocation_id:    0
  begin_time:       2017-08-11 11:45:20.325414
  end_time:         2017-08-11 11:45:20.553478
  diag_status:      COMPLETED
  inserted_ras:     f
  log_dir:          /tmp/170811114518534194
  cmd_line:         /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --noallocation --bucket node_health
Total_Result_Records: 1
Result_Record_1:
  history_time:     2017-08-11 11:45:20.553478
  run_id:           170811114518534194
  test_name:        test_memsize
  node_name:        c650f02p27
  serial_number:    781621A
  begin_time:       2017-08-11 11:45:18.803174
  end_time:         2017-08-11 11:45:20.464567
  status:           PASS
  log_file:         /tmp/170811114518534194/test_memsize/c650f02p27-2017-08-11-11_45_18.output
...
done
```

### 7.2.5 *Running Diagnostic via LSF*

To demonstrate we can run Diagnostic via LSF, we create the script below:

```
#!/bin/bash
# sample bsub script for single script to both submit the job and
# run the job.
#

source /opt/ibm/spectrumcomputing/lsf/conf/profile.lsf

if [ -z $LSB_BATCH_JID ]; then
    bsub  \
        -q csm_queue \
        -R "1*{LN}+20*{CN}"  \
        -o $PWD/log/lsf-%J.out $0
else
    hs=`hostname`;
    echo hostname=$hs
    echo PWD=$PWD
    /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --noallocation --test test_memsize
    echo RC=$?
fi
```

Formatted: English (US)

Running the script above, the following output is produced:

```
$ /u/diagadmin/sw45 > cat log/lsf-2601.out
Sender: LSF System <lsfadmin@c650f02p07>
Subject: Job 2601: <./lsf_diag.sh> in cluster <c650mnp01> Done

Job <./lsf_diag.sh> was submitted from host <c650f02p07> by user <diagadmin> in cluster
<c650mnp01>.
Job was executed on host(s) <1*c650f02p07>, in queue <csm_queue>, as user <diagadmin> in
cluster <c650mnp01>.
                            <20*c650f02p25>
```

```
</u/diagadmin> was used as the home directory.
</u/diagadmin/sw45> was used as the working directory.
Started at Results reported on
Your job looked like:

------------------------------------------------------------
# LSBATCH: User input
./lsf_diag.sh
------------------------------------------------------------

Successfully completed.

Resource usage summary:

    CPU time :                                   0.15 sec.
    Max Memory :                                 -
    Average Memory :                             -
    Total Requested Memory :                     -
    Delta Memory :                               -
    Max Swap :                                   -
    Max Processes :                              -
    Max Threads :                                -
    Run time :                                   7 sec.
    Turnaround time :                            0 sec.

The output (if any) follows:

hostname=c650f02p07
PWD=/u/diagadmin/sw45
Health Check Diagnostics version beta2-2017, running on Linux 3.10.0-514.el7.ppc64le,
c650f02p07 machine.
Using configuration file /opt/ibm/csm/hcdiag/etc/hcdiag.properties.
XCAT fanout ignored, allocation disabled.
Using tests configuration file /opt/ibm/csm/hcdiag/etc/test.properties.
Health Check Diagnostics, run id 170731100153805960, initializing...
Validating command argument target.
Validating command argument test.
Preparing to run test_memsize.
Executable: /opt/ibm/csm/hcdiag/tests/test_memsize/test_memsize.sh exists on target node(s).
test_memsize started on 1 node(s) at 2017-07-31 10:01:54.045271. It might take up to 10s.
INFO: running in Node mode.
Info: tests.ppping ignored, can not run in Node mode.
Info: bucket.ib ignored, can not run in Node mode.
Info: bucket.beta1 ignored, can not run in Node mode.

test_memsize ended on 1 node(s) at 2017-07-31 10:01:54.104203, rc= 0, elapsed time:
0:00:00.058932
test_memsize PASS on node c650f02p07, serial number: 78164AA.
Health Check Diagnostics ended, exit code 0.
RC=0
```

The same directory and files structure were created in *logdir*.

```
c650f02p07:/ > ls -oR /tmp/*170731100153805960* |grep -v total
-rw-r--r-- 1 diagadmin 2386 Jul 31 10:01 /tmp/hcdiag_run-170731100153805960.log

/tmp/170731100153805960:
drwxr-xr-x 2 diagadmin 51 Jul 31 10:01 test_memsize

/tmp/170731100153805960/test_memsize:
-rw-r--r-- 1 diagadmin 81 Jul 31 10:01 c650f02p07-2017-07-31-10_01_54.output
```

Below the content of the database associated to the run. Node allocation zero, means Diagnostic did not create an allocation.

```
c650f02p07:/ > /opt/ibm/csm/hcdiag/bin/hcdiag_query.py --runid 170731100153805960 --results
/opt/ibm/csm/hcdiag/bin/hcdiag_query.py, version beta2, running on Linux 3.10.0-
514.el7.ppc64le, c650f02p07 machine.
run_id was specified, all other arguments will be ignored
---
runData:
  history_time:     2017-07-31 10:02:12.368932
  run_id:           170731100153805960
  allocation_id:    0
  begin_time:       2017-07-31 10:02:12.140217
  end_time:         2017-07-31 10:02:12.368932
  diag_status:      COMPLETED
  inserted_ras:     f
  log_dir:          /tmp/170731100153805960
  cmd_line:         /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --noallocation --test test_memsize
Total_Result_Records: 1
Result_Record_1:
  history_time:     2017-07-31 10:02:12.368932
  run_id:           170731100153805960
  test_name:        test_memsize
  node_name:        c650f02p07
  serial_number:    78164AA
  begin_time:       2017-07-31 10:01:54.045271
  end_time:         2017-07-31 10:02:12.279698
  status:           PASS
  log_file:         /tmp/170731100153805960/test_memsize/c650f02p07-2017-07-31-10_01_54.output
...
done
```

## 7.3 RAS Integration

To demonstrate the Diagnostics integration with the RAS subsystem, we created a simple script, *diag_node_test_fail* as shown below in */opt/ibm/csm/ras/actions* dir.
The script only prints the parameters received by the RAS subsystem.

```
$ cat /opt/ibm/csm/ras/actions/diag_node_test_fail
#! /bin/bash
readonly ACTION=$0

echo "Running $ACTION"
echo "RAS event recorded"

echo $1 | /usr/bin/python -m json.tool
```

Diagnostics currently has only one RAS message type registered in the database.
By setting the *hcdiag.test.fail* entry's *control_action* with the script above, we define the flow:

- When a test initiated by Diagnostics fails, it generates the RAS event *hcdiag_test_fail*

- An entry in the csm_ras_event_action is created
- The RAS subsystem executes the control_action associated with the event: *diag_note_test_fail*
- The *diag_note_test_fail* script receives the RAS message as an argument in json format and prints it
- The log of the execution of *diag_note_test_fail* script is stored in */var/log/ibm/csm/ras/actions/diag_note_test_fail*

```
$ /opt/ibm/csm/bin/csm_ras_msg_type_query -m hcdiag%
---
Total_Records: 1
RECORD_1:
  msg_id:           hcdiag.test.fail
  control_action:   diag_node_test_fail
  description:      NONE
  enabled:          t
  message:          $(test) FAIL on node $(node) serial number: $(sn) (details in $(file)).
  set_not_ready:    f
  set_ready:        f
  severity:         WARNING
  threshold_count:  1
  threshold_period: 5
  visible_to_users: t
...
```

We changed the *args* value of test_memsize test in the properties file, to make sure the test will fail.

```
[tests.test_memsize]
description = This tests checks the available system memory size
executable  = /opt/ibm/csm/hcdiag/tests/test_memsize/test_memsize.sh
args        = 520
#args        = 250
```

```
$ /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test test_memsize --target c650f02p11
INFO: xcat seems to be installed in /opt/xcat/bin. Running in Management mode
Health Check Diagnostics version beta2-2017, running on Linux 3.10.0-514.el7.ppc64le, c650mnp01
machine.
Using configuration file /opt/ibm/csm/hcdiag/etc/hcdiag.properties.
Using tests configuration file /opt/ibm/csm/hcdiag/etc/test.properties.
Health Check Diagnostics, run id 170801153816755860, initializing...
Validating command argument target.
Validating command argument test.
Allocation request successful, id= 275
Preparing to run test_memsize.
Executable: /opt/ibm/csm/hcdiag/tests/test_memsize/test_memsize.sh exists on remote node(s).
test_memsize started on 1 node(s) at 2017-08-01 15:38:17.576284. It might take up to 10s.

test_memsize ended on 1 node(s) at 2017-08-01 15:38:18.655463, rc= 1, elapsed time:
0:00:01.079179
test_memsize FAIL on node c650f02p11, serial number: 78165BA, rc= 1. (details in
/tmp/170801153816755860/test_memsize/c650f02p11-2017-08-01-15_38_18.output)
A RAS event was created in the database for this message
Request csmd to release the allocation 275.
Allocation 275 delete request successful.
Health Check Diagnostics ended, exit code 100.
```

An entry was inserted in the database's *csm_ras_event_action* table.

```
$ /opt/ibm/csm/bin/csm_ras_event_query --msg_id=hcdiag%
---
Total_Records: 1
RECORD_1:
  msg_id:           hcdiag.test.fail
  severity:         WARNING
  time_stamp:       2017-08-01 15:38:18
  location_name:    c650mnp01
  count:            0
  control_action:   diag_node_test_fail
  message:          test_memsize FAIL on node c650f02p11 serial number: 78165BA (details in
/tmp/170801153816755860/test_memsize/c650f02p11-2017-08-01-15_38_18.output).
  raw_data:
...
```

The *diag_node_test_fail* script output is stored in the /var/log/ibm/csm/ras/actions directory.

```
$ cat /var/log/ibm/csm/ras/actions/diag_node_test_fail.log
[2017-08-01 15:38:18] Running /opt/ibm/csm/ras/actions/diag_node_test_fail
[2017-08-01 15:38:18] RAS event recorded
[2017-08-01 15:38:18] {
[2017-08-01 15:38:18]     "control_action": "diag_node_test_fail",
[2017-08-01 15:38:18]     "ctxid": "26",
[2017-08-01 15:38:18]     "description": "NONE",
[2017-08-01 15:38:18]     "enabled": "1",
[2017-08-01 15:38:18]     "file": "/tmp/170801153816755860/test_memsize/c650f02p11-2017-08-01-
15_38_18.output",
[2017-08-01 15:38:18]     "location_name": "c650mnp01",
[2017-08-01 15:38:18]     "message": "test_memsize FAIL on node c650f02p11 serial number:
78165BA (details in /tmp/170801153816755860/test_memsize/c650f02p11-2017-08-01-
15_38_18.output).",
[2017-08-01 15:38:18]     "msg_id": "hcdiag.test.fail",
[2017-08-01 15:38:18]     "node": "c650f02p11",
[2017-08-01 15:38:18]     "raw_data": "",
[2017-08-01 15:38:18]     "rc": "1",
[2017-08-01 15:38:18]     "set_not_ready": "0",
[2017-08-01 15:38:18]     "set_ready": "0",
[2017-08-01 15:38:18]     "severity": "WARNING",
[2017-08-01 15:38:18]     "sn": "78165BA",
[2017-08-01 15:38:18]     "test": "test_memsize",
[2017-08-01 15:38:18]     "threshold_count": "1",
[2017-08-01 15:38:18]     "threshold_period": "5",
[2017-08-01 15:38:18]     "time_stamp": "2017-08-01T15:38:18 -0400",
[2017-08-01 15:38:18]     "visible_to_users": "1"
[2017-08-01 15:38:18] }
```

## 7.4 Big Data Store Integration

Diagnostics application will send the log messages to Big Data Store by writing them to the syslog; syslog is then pushed to the Big Data Store with rsyslog daemon.
Syslog among other Data Sources were aggregated for the functional demonstration of the Big Data Store use cases.
Details were described in the SW31 milestone report.

### *7.5 Tests running time*

The data presented in the Table 15: running time, is a result of multiple runs on four similar machines from IBM WSC system. Machine has the following characteristics:

- Witherspoon, machine model 8335-GTW
- 6 with 6 GPUs
- 606 GiB
- CSM 12.0

WSC system:

- 1 management none
- Aggregator node is the same as management
- 4 utility nodes: 1 launch, 1 workload, 2 logging nodes
- 57 compute nodes

The machines were dedicated to the runs, i.e., no other jobs were running on the machine, even for tests that are not intrusive.
The purpose of this data is to give an idea of the duration of the test and it is not meant to use as guideline. Most of the tests are system, machine (hardware) and environment dependent.

It the test requires more than one node; two nodes were used for the measurement.

The reported time is the average of all the runs, rounded up.
The tests ran using Diagnostic in Node mode, command:

```
/opt/ibm/csm/hcdiag_run.py -test <test_name> --noallocation
```

except tests that runs only on Management mode, command:

```
/opt/ibm/csm/hcdiag_run.py -test <test_name> --target <node>
```

Those tests are marked with (*) in the Table 15: running time.

**Table 15: running time**

| NAME | RUNNING TIME (seconds) |
|---|---|

| | | |
|---|---|---|
| 1. | | .08 |
| | *7.5.1 chk-aslr*<br>chk-aslr | |
| 2. | chk-boot-time (*) | .85 |
| 3. | chk-capi | 3.70 |
| 4. | chk-cpu | 3.00 |
| 5. | chk-cpu-count | 0.30 |
| 6. | chk-csm-health | 0.80 |
| 7. | chk-free-memory | 0.08 |
| 8. | chk-gpfs-mount | 0.30 |
| 9. | chk-gpu-ats | 0.08 |
| 10. | chk-hca-attributes | 0.03 |
| 11. | chk-ib-node | 750.00 |
| 12. | chk-ib-pcispeed | 0.21 |
| 13. | chk-idle-state | 0.08 |
| 14. | chk-kworker | 0.18 |
| 15. | chk-led | 3.00 |
| 16. | chk-load-average | 0.08 |
| 17. | chk-load-cpu | 0.13 |
| 18. | chk-load-mem | 0.13 |
| 19. | chk-memory | 1.30 |
| 20. | chk-mlnx-pci | 60.00 |
| 21. | chk-mlxlink-pci | 0.50 |
| 22. | chk-nfs-mount | 0.10<br>(2 filesystems) |
| 23. | chk-noping | 1.28 |
| 24. | chk-nvidia-clocks | 0.30 |

| | | |
|---|---|---|
| 25. | chk-nvidia-smi | 0.17 |
| 26. | chk-nvidia-vbios | 0.28 |
| 27. | chk-nvlink-speed | 0.28 |
| 28. | chk-nvme | 0.24 |
| 29. | chk-nvme-mount | 0.8<br>(1 filesystem) |
| 30. | chk-os | 0.22 |
| 31. | chk-power (*) | 0.37 |
| 32. | chk-process | 0.38 |
| 33. | chk-smt | 0.12 |
| 34. | chk-sw-level | 0.30 |
| 35. | chk-sys-firmware (*) | 3.80 |
| 36. | chk-sys-firmware-local | 0.30 |
| 37. | chk-temp | 0.24 |
| 38. | chk-zombies | 0.18 |
| 39. | compdiag | 507.96 |
| 40. | daxpy | 55.00 |
| 41. | **Error! Reference source not found.** | 55.00 |
| 42. | dcgm-diag | 594.00 |
| 43. | dcgm-diag-double | 192 |
| 44. | dcgm-diag-single | 588 |
| 45. | dcgm-health | 1.36 |
| 46. | dgemm | 118.00 |
| 47. | dgemm-gpu | 600.00 |
| 48. | dgemm-per-socket | 400.00 |
| 49. | fieldiag | full – 8 hours<br>p0only – 4 hours |

| | | |
|---|---|---|
| 50. | gpfsperf<br>very environment dependent. Run on /gpfs/wscgpfs01 (no ESS): 102T size, 82TB free, with default gpfsperf configuration. | 94.1 |
| 51. | gpudirect | 25.1 |
| 52. | gpu-health | 16.00 |
| 53. | hcatest | 21.06 |
| 54. | hxecache | 60.00 |
| 55. | hxecpu | 60.00 |
| 56. | hxecpu_pass2 | 1102.00 |
| 57. | hxediag_eth | 600.00 |
| 58. | hxediag_ib | 2700.00 |
| 59. | hxeewm_pass2 | 70.00 |
| 60. | hxefabricbus_pass2 | 1094.00 |
| 61. | hxefpu64 | 80.00 |
| 62. | hxefpu64_pass2 | 3:15 |
| 63. | hxemem64 | 60.00 |
| 64. | hxemem64_pass2 | 275.00 |
| 65. | hxenvidia | 60.00 |
| 66. | hxenvidia_pass2 | 900.00 |
| 67. | hxerng_pass2 | 92.00 |
| 68. | hxesctu_pass2 | 65.00 |
| 69. | hxestorage_nvme | 80.00 |
| 70. | hxestorage_nvme_pass2 | 150.00 |
| 71. | hxestorage_sd | 300.00 |
| 72. | hxestorage_sd_pass2 | 2284.00 |
| 73. | ibcredit | 2.20 |

| | | |
|---|---|---|
| 74. | ibverbs | 5.01 |
| 75. | ipoib | 0.17 |
| 76. | jlink | 0.95 |
| 77. | lnklwls | 1.50 |
| 78. | lnkqual | 6.42 |
| 79. | mmhealth | 0.13 |
| 80. | nsdperf | 7.52 |
| 81. | nvvs | 588.00 |
| 82. | p2pBandwidthLatencyTest | 8.00 |
| 83. | ppping (*) | 3.5 |
| 84. | rpower | 2.70 |
| 85. | rvitals | 13.15 |
| 86. | swhealth | 9.10 |
| 87. | test-simple | 0.07 |

# 8. Appendix D: Diagnostic RAS events

Diagnostic plays the “Publisher” role, generating RAS events for Diagnostic application and test failures. RAS events initiated by diagnostic programs will have the top level string “hcdiag.”, RAS event ID format is “toplevel.sublevel1.sublevel2”.

Table 16 - Diagnostic RAS events lists all events defined by Diagnostic application. Currently, at installation time, by default, only the “hcdiag.test.fail” event is configured to be generated by Diagnostic.

If more Diagnostic events are desired, use the CSM API *csm_ras_msg_type_create,* to add the events listed in Table 16 - Diagnostic RAS events.

**Table 16 - Diagnostic RAS events**

| Message_id | Severity | Message |
|---|---|---|
| 'hcdiag.fwk.signal | INFO | 'Signal $(signal) caught. Exiting... |
| 'hcdiag.fwk.started' | INFO | Health Check Diagnostics version $(version), running on $(os) $(osversion), $(location_name) machine |
| 'hcdiag.fwk.ended' | INFO | Health Check Diagnostics ended, exit code $(rc) |
| 'hcdiag.fmk.sn_err' | FATAL | Could not retrieve the Serial Number of node $(node).' |
| 'hcdiag.fmk.node_err' | FATAL | Node $(node) has issue: $(status). |
| 'hcdiag.csmi.err' | FATAL | 'Error invoking csmi $(api), rc= $(rc). Exiting...'} |
| hcdiag.csmi.warn' | WARNING | Error invoking csmi $(api), rc= $(rc). |
| hcdiag.csmi.alloc_err' | FATAL | Error invoking csmi_allocation_create, node: $(node), rc=$(rc). Check permission or if nodes are in use |
| hcdiag.csmi.allocdel_err | WARNING | Error invoking csmi csm_allocation_delete, allocation_id= $(id), rc= $(rc).'}, |
| hcdiag.test.fail | ERROR | $(test) FAIL on node $(node), serial number: $(sn), rc= $(rc). (details in $(file)) |
| hcdiag.xcat.noderange_err | FATAL | Invalid nodes and/or groups in noderange: $(noderange), or user has persmission issues |
| hcdiag.xcat.sn_err | FATAL | Could not retrieve the serial number of the nodes $(nodes), rc= $(rc). Check noderange syntax/node with problem, xdsh permission. |

# 9. Appendix E: How to add a new test to Diagnostic

This is the sequence of steps to create a simple test that prints the argument, and if the argument is the word “pass” (or any combination of upper/lower case) , returns zero (PASS) , otherwise it returns non-zero ( FAIL).

1. Create *mytest* directory in your sandbox, under hcdiag/tests

```
$ mkdir <sandbox>/hcdiag/tests/mytest
```

2. Create the test script under the directory created in step 1., adding first the mandatory lines, highlighted in bold:

```
$ cat <sandbox>/tests/mytest/mytest.sh
#!/bin/bash

## These lines are mandatory, so the framework knows the name of the log file
## This is necessary when invoked standalone --with xcat-- and common_fs=yes
if [ -n "$HCDIAG_LOGDIR" ]; then
   [ ! -d "$HCDIAG_LOGDIR" ] && echo "Invalid directory. Exiting" && exit 1
   THIS_LOG=$HCDIAG_LOGDIR/`hostname -s`-`date +%Y-%m-%d-%H_%M_%S`.output
   echo $THIS_LOG
   exec 2>$THIS_LOG 1>&2
fi

[ $# -ne 1 ] && echo "Argument is missing. Usage $0 word" && exit 1
echo $1
[ `echo "$1" | tr a-z A-Z` == "PASS" ] && exit 0
exit 1
```

3. Add the following lines to your <sandbox>/etc/test.properties file

```
# mytest test
[tests.mytest]
description = mytest test
executable  = <sandbox>/tests/mytest/mytest.sh
args=PaSs
```

4. Testing *mytest*

Console output, with *args=PaSs* :

```
c699mgt00:/ > sudo /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test mytest --target c699c001 --
testproperties /home/aldas/CAST/work/csm/hcdiag/etc/test.properties --noallocation
INFO: xcat seems to be installed in /opt/xcat/bin. Running in Management mode
Health Check Diagnostics version 1.2.1, running on Linux 4.14.0-98.el7a.bz1611676.ppc64le,
c699mgt00 machine.
Using configuration file /opt/ibm/csm/hcdiag/etc/hcdiag.properties.
Using tests configuration file /home/aldas/CAST/work/csm/hcdiag/etc/test.properties.
Health Check Diagnostics, run id 180914114705880469, initializing...
Validating command argument test.
Validating command argument target.
Preparing to run mytest.
mytest started on 1 node(s) at 2018-09-14 11:47:08.028738. It might take up to 10s.

mytest ended on 1 node(s) at 2018-09-14 11:47:16.080514, rc= 0, elapsed time: 0:00:08.051776
mytest PASS on node c699c001, serial number: 131A74A.
Health Check Diagnostics ended, exit code 0.
```

Test output:

```
c699mgt00:/ > cat /tmp/180914114705880469/mytest/c699c001-2018-09-14-11_47_14.output
 PaSs
 Remote_command_rc = 0
```

Modify <sandbox/etc/testproperties, mytest args to: *args=doNotPass*

Console output:

```
c699mgt00:/ > sudo /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test mytest --target c699c001 --
testproperties /home/aldas/CAST/work/csm/hcdiag/etc/test.properties --noallocation
INFO: xcat seems to be installed in /opt/xcat/bin. Running in Management mode
Health Check Diagnostics version 1.2.1, running on Linux 4.14.0-98.el7a.bz1611676.ppc64le,
c699mgt00 machine.
Using configuration file /opt/ibm/csm/hcdiag/etc/hcdiag.properties.
Using tests configuration file /home/aldas/CAST/work/csm/hcdiag/etc/test.properties.
Health Check Diagnostics, run id 180914114927383469, initializing...
Validating command argument test.
Validating command argument target.
Preparing to run mytest.
mytest started on 1 node(s) at 2018-09-14 11:49:29.540856. It might take up to 10s.

mytest ended on 1 node(s) at 2018-09-14 11:49:37.299409, rc= 1, elapsed time: 0:00:07.758553
mytest FAIL on node c699c001, serial number: 131A74A, rc= 1. (details in
/tmp/180914114927383469/mytest/c699c001-2018-09-14-11_49_35.output)
A RAS event was created in the database for this message
Health Check Diagnostics ended, exit code 100.
```

Test output:

```
c699mgt00:/ > cat /tmp/180914114927383469/mytest/c699c001-2018-09-14-11_49_35.output
 doNotPass
 Remote_command_rc = 1
```

Modify <sandbox/etc/testproperties, remove mytest args:

Console output:

```
c699mgt00:/ > sudo /opt/ibm/csm/hcdiag/bin/hcdiag_run.py --test mytest --target c699c001 --
testproperties /home/aldas/CAST/work/csm/hcdiag/etc/test.properties --noallocation
INFO: xcat seems to be installed in /opt/xcat/bin. Running in Management mode
Health Check Diagnostics version 1.2.1, running on Linux 4.14.0-98.el7a.bz1611676.ppc64le,
c699mgt00 machine.
Using configuration file /opt/ibm/csm/hcdiag/etc/hcdiag.properties.
Using tests configuration file /home/aldas/CAST/work/csm/hcdiag/etc/test.properties.
Health Check Diagnostics, run id 180914115047611983, initializing...
Validating command argument test.
Validating command argument target.
Preparing to run mytest.
mytest started on 1 node(s) at 2018-09-14 11:50:49.753886. It might take up to 10s.

mytest ended on 1 node(s) at 2018-09-14 11:50:57.920680, rc= 1, elapsed time: 0:00:08.166794
mytest FAIL on node c699c001, serial number: 131A74A, rc= 1. (details in
/tmp/180914115047611983/mytest/c699c001-2018-09-14-11_50_55.output)
A RAS event was created in the database for this message
Health Check Diagnostics ended, exit code 100.
```

Test output:

```
c699mgt00:/ > cat /tmp/180914115047611983/mytest/c699c001-2018-09-14-11_50_55.output
 Argument is missing. Usage ./mytest.sh word
 Remote_command_rc = 1
```