___READ THIS DOCUMENT SECOND___

Cluster System Management (CSM):

Version 1.0.0

Installation and Configuration Guide

April 2018

Author:
Fernando Pizzano

Co-Authors:

Nathan Besaw, Nicholas Buonarota, John Dunham, Patrick Lundgren, William Morrison, Alda Ohmacht, Sandy Kaur, and Lars Schneidenbach.

Table of Contents

1 Introduction

The purpose of this document is to guide the system administrator through the installation and configuration of Cluster System Management (CSM).

CSM is developed for use on POWER systems only. These POWER systems consist of:

- Witherspoon as compute and utility (Login, Launch and Workload manager) nodes
- Boston as management nodes and Big Data Store

This document assumes POWER systems are fully operational and running Red Hat Pegas 1.0 or higher.

To obtain support for CSM, please contact IBM Support Service.

2 Pre-Requisites

All nodes participating in CSM must be running Red Hat Enterprise Linux for PPC64LE.

2.1 Software Dependencies

CSM has software dependencies. Verify that the following packages are installed:

Hard dependencies
Without these dependencies CSM cannot run.

Software	Version	Comments
xCAT	2.13.3 or higher	
Postgres SQL	9.2.18 or higher	xCAT documentation for migration
openssl-libs	1.0.1e-60 or higher	
perl-YAML	0.84-5 or higher	Required by the Diagnostic's tests.
perl-JSON	2.59 or higher	Required by the Diagnostic's tests that get information from the UFM.
cast-boost	1.60.0-4	Found on Box
P9 Witherspoon firmware level	**BMC:** ibm-v2.0-0-r26-0-g7285b52 **Host:** IBM-witherspoon-ibm-OP9_v1.19_1.111 or higher	Found on Box

Soft dependencies
These dependencies are highly suggested.

Software	Version	Comments
NVIDIA DCGM	datacenter-gpu-manager-1.4.1-1.ppc64le.rpm	• Needed by the Diagnostics and health check and CSM GPU inventory. • All nodes with GPUs.
NVIDIA Cuda Toolkit	cuda-repo-rhel7-9-2-local-9.2.64-1.ppc64le.rpm	• All nodes with GPUs
NVIDIA Driver	cuda-drivers-396.19-1.ppc64le	• Needed by NVIDIA Data Center GPU Manager (DCGM). • All nodes with GPUs.
IBM HTX	htxrhel72le-475-LE.ppc64le	• Needed by the Diagnostics and health check. • All nodes • HTX requires: ○ net-tools package (ifconfig command)

		○ mesa-libGLU-devel and mesa-libGLU packages
Spectrum MPI	10.02	• Needed by the Diagnostic tests: dgemm, dgemm-gpu, jlink and daxpy tests. • Spectrum MPI requires: ○ IBM ESSL (IBM Engineering and Scientific Subroutine Libray) ○ IBM XL Fortran
sudo	sudo-1.8.19p2-13.el7.ppc64le	• Required by the Diagnostic's tests that needs to run as root
lm-sensors	3.4.0	• Required by the Diagnostic's temperature tests.

Software with dependencies on CSM

Software	Version	Comments
Spectrum MPI	10.02	
Spectrum LSF	10.1	
Burst Buffer	1.0.0	
IBM POWER HW Monitor	ibm-crassd-0.7-3	If installed on the service nodes, ibmpowerhwmon can be configured to create CSM RAS events from BMC sources via csmrestd.

IBM Spectrum LSF (LSF), Burst Buffer (BB), and Job Step Manager (JSM) all have dependencies on CSM. Whenever a new version of CSM is installed, all dependent software must be restarted to reload the updated version of the CSM API library.

If any of these packages are not already installed or for more information about these packages, please refer to CORAL Software Overview (CORAL_Readme_v1.0.dox) located in Box. In this document you will find where these packages are located and how to install them.

3 Updating from a previous version

CSM 1.0.0 does not support incremental upgrades, fixes, and patches from prerelease versions (for example, from PRPQ to CSM 1.0.0). System administrator should completely uninstall CSM and its sub components (for example, the CSM DB), then freshly install the new version.

4 Installation

4.1 List of CSM RPMs

The following RPMs are all of the required RPMs for CSM. These RPMs can be found on Box in the same folder as this document. Please verify that you have all required RPMs.

For CSM, all CSM rpms have a version of ibm-csm-*component*-1.0.0-*commit_sequence*.ppc64le.rpm, for example, ibm-csm-core-1.0.0-9460.ppc64le.rpm. The *commit_sequence* portion is an increasing value that changes every time a new commit is added to the repository. Because the *commit_sequence* changes so frequently, this document makes use of a wildcard for this portion of the rpm name.

ibm-csm-core holds CSM infrastructure daemon and configuration file examples.

ibm-csm-api holds all CSM APIs and the corresponding command line interfaces.

ibm-csm-hcdiag holds the diagnostics and health check framework with all available tests interface.

ibm-csm-db holds CSM DB configuration files and scripts.

ibm-csm-bds holds BDS configuration files, scripts, and documentation.

ibm-csm-restd holds the CSM REST daemon, configuration file example, and a sample script.

Below is a chart indicating where each of these RPMs must be installed.

	Management node	**Service node**	**Login node**	**Launch node**	**Compute node**	**ESS Servers**	**UFM Server**	**BDS Server**
ibm-csm-core	X	X	X	X	X			
ibm-csm-api	X	X	X	X	X			
ibm-csm-hcdiag	X	X	X	X	X			
ibm-csm-db	X							
ibm-csm-bds								X
ibm-csm-restd		X						

Follow the steps outlined below to ensure a successful installation of these RPMs.

4.2 Installing CSM onto the Management Node

On the management node:

Install the cast-boost RPMs, which can be found on Box

```
$ rpm -ivh cast-boost-*.rpm
```

Install the flightlog RPM and the following CSM RPMs, which can be found on Box.

- ibm-flightlog
- ibm-csm-core
- ibm-csm-api
- ibm-csm-db
- ibm-csm-hcdiag

```
$ rpm -ivh ibm-flightlog-1.0.0-*.ppc64le.rpm \
ibm-csm-core-1.0.0-*.ppc64le.rpm \
ibm-csm-api-1.0.0-*.ppc64le.rpm  \
ibm-csm-db-1.0.0-*.noarch.rpm \
ibm-csm-hcdiag-1.0.0-*.noarch.rpm
```

4.3 Installing CSM onto the Service Nodes

For service nodes, CSM supports diskless and full disk install:

Diskless

Clone existing images, add the following packages to the "otherpkgs" directory and list, then run genimage. See Appendix 7.1 for more details.

```
cast-boost-*
ibm-flightlog-1.0.0-*.ppc64le
ibm-csm-core-1.0.0-*.ppc64le
ibm-csm-api-1.0.0-*.ppc64le
ibm-csm-db-1.0.0-*.noarch
ibm-csm-hcdiag-1.0.0-*.noarch
ibm-csm-restd-1.0.0-*.ppc64le
```

Full disk

Install the cast-boost RPMs, which can be found on Box

```
$ rpm -ivh cast-boost-*.rpm
```

Install the flightlog RPM and the following CSM RPMs, which can be found on Box.

- ibm-flightlog
- ibm-csm-core
- ibm-csm-api
- ibm-csm-hcdiag
- ibm-csm-restd

```
$ rpm -ivh ibm-flightlog-1.0.0-*.ppc64le.rpm \
ibm-csm-core-1.0.0-*.ppc64le.rpm \
ibm-csm-api-1.0.0-*.ppc64le.rpm \
ibm-csm-hcdiag-1.0.0-*.noarch.rpm \
ibm-csm-restd-1.0.0-*.ppc64le.rpm
```

4.4 Installing CSM onto the Login, Launch, and Workload manager Nodes

For login, launch and workload manager nodes, CSM supports diskless and full disk install:

Diskless

Clone existing images, add the following packages to the “otherpkgs” directory and list, then run genimage. See Appendix 7.1 for more details.

```
cast-boost-*
ibm-flightlog-1.0.0-*.ppc64le
ibm-csm-core-1.0.0-*.ppc64le
ibm-csm-api-1.0.0-*.ppc64le
ibm-csm-hcdiag-1.0.0-*.noarch
```

Full disk

Install the cast-boost RPMs, which can be found on Box

```
$ rpm -ivh cast-boost-*.rpm
```

Install the flightlog RPM and the following CSM RPMs, which can be found on Box.

- ibm-flightlog
- ibm-csm-core
- ibm-csm-api
- ibm-csm-hcdiag

```
$ rpm -ivh ibm-flightlog-1.0.0-*.ppc64le.rpm \
ibm-csm-core-1.0.0-*.ppc64le.rpm \
ibm-csm-api-1.0.0-*.ppc64le.rpm \
ibm-csm-hcdiag-1.0.0-*.noarch.rpm
```

4.5 Installing CSM onto the Compute Nodes

For compute nodes, CSM supports diskless and full disk install:

Diskless

Clone existing images, add the following packages to the “otherpkgs” directory and list, then run genimage. See Appendix 7.1 for more details.

```
cast-boost-*
ibm-flightlog-1.0.0-*.ppc64le
ibm-csm-core-1.0.0-*.ppc64le
ibm-csm-api-1.0.0-*.ppc64le
ibm-csm-hcdiag-1.0.0-*.noarch
```

Full disk

Install the cast-boost RPMs, which can be found on Box

Note: replace "/path/to/rpms" with the appropriate location for your system.

```
$ xdsh compute "cd /path/to/rpms; rpm -ivh cast-boost-*.rpm"
```

Install the flightlog RPM and the following CSM RPMs, can be found on Box.

Note: replace "/path/to/rpms" with the appropriate location for your system.

- ibm-flightlog
- ibm-csm-core
- ibm-csm-api
- ibm-csm-hcdiag

```
$ xdsh compute "cd /path/to/rpms; \
rpm -ivh ibm-flightlog-1.0.0-*.ppc64le.rpm \
ibm-csm-core-1.0.0-*.ppc64le.rpm \
ibm-csm-api-1.0.0-*.ppc64le.rpm \
ibm-csm-hcdiag-1.0.0-*.noarch"
```

5 Configuration

Now that everything needed for CSM has been installed, CSM needs to be configured.

5.1 General Configuration

Suggested general configuration to verify:

Open files limit:

```
$ ulimit -n
500000
```

5.2 CSM DB Configuration

On the management node create the *csmdb* schema by running the *csm_db_script.sh*. This script assumes that xCAT has migrated to postgresql. Details on this migration process can be found in the xCAT read the docs.

```
$ /opt/ibm/csm/db/csm_db_script.sh
-----------------------------------------------------------------------------------------------------------------------
[Start   ] Welcome to CSM database automation script.
[Info    ] PostgreSQL is installed
[Info    ] csmdb database user: csmdb already exists
[Complete] csmdb database created.
[Complete] csmdb database tables created.
[Complete] csmdb database functions and triggers created.
[Complete] csmdb table data loaded successfully into csm_db_schema_version
[Complete] csmdb table data loaded successfully into csm_ras_type
[Info    ] csmdb DB schema version (15.0)
-----------------------------------------------------------------------------------------------------------------------
```

This will create *csmdb* database and configure it with default settings.

5.3 Default Configuration Files

On the management node copy default configuration and ACL (Access Control List) files from */opt/ibm/csm/share/etc* to */etc/ibm/csm*.

```
$ cp /opt/ibm/csm/share/etc/*.cfg /etc/ibm/csm/
$ cp /opt/ibm/csm/share/etc/csm_api.acl /etc/ibm/csm/
```

Review the configuration and ACL files. Make the following suggested updates (note that hostnames can also be IP addresses, especially if a particular network interface is desired for CSM communication):

1. Substitute all "__MASTER__" occurrences in the configuration files with the management node hostname.

2. On aggregator configurations, substitute "__AGGREGATOR__" with the corresponding service node hostname.
3. On compute configurations, substitute "__AGGREGATOR_A__" with the assigned primary aggregator.
4. On compute configurations, substitute "__AGGREGATOR_B__" with the secondary aggregator or leave it untouched if you set up a system without failover.
5. If an aggregator is run on the management node too, make sure to provide a unique entry for csm.net.local_client_listen.socket in order to avoid name collision and strange behavior.
6. Create a new Linux group for privileged access.
    a. Add users to this group.
    b. Make this group privileged in the ACL file. (For more information see Section "6.3.1 Configuring user control, security, and access level" of the "CSM User Guide")

Review all configuration and ACL files.

Copy the configuration files to the proper nodes:

On management node:

```
$ xdcp compute /etc/ibm/csm/csm_compute.cfg /etc/ibm/csm/csm_compute.cfg
$ xdcp login,launch /etc/ibm/csm/csm_utility.cfg /etc/ibm/csm/csm_utility.cfg
$ xdcp compute,login,launch /etc/ibm/csm/csm_api.acl /etc/ibm/csm/csm_api.acl
$ xdcp compute,login,launch /etc/ibm/csm/csm_api.cfg /etc/ibm/csm/csm_api.cfg
```

5.3.1 SSL Configuration

If an SSL setup is desired, the csm.net.ssl section of the config file(s) needs to be set up.

```
"ssl":
{
    "ca_file" : "<full path to CA file>",
    "cred_pem": "<full path to credentials in pem format>"
}
```

If the strings are non-empty, the daemon assumes that SSL is requested. This means if the SSL setup fails, it will not fall back to non-SSL communication.

5.3.2 Heartbeat interval

CSM daemons heartbeat interval can be configure on csm.net section of the config file(s).

```
        "net" :
        {
            "heartbeat_interval" : 15,
            "local_client_listen" :
            {
                "socket"      : "/run/csmd.sock",
                "permissions" : 777,
                "group"       : ""
            },
```

The heartbeat interval setting defines the time between 2 subsequent unidirectional heartbeat messages in case there's no other network traffic on a connection. The default is 15 Seconds. If two daemons are configured with a different interval, they will use the minimum of the two settings for the heartbeat on the connection. This allows to configure a short interval between Aggregator and Master and a longer interval between Aggregator and Compute to reduce network interference on compute nodes.
It might take up to 3 intervals to detect a dead connection because of the following heartbeat process: After receiving a message the daemon waits one interval to sent its heartbeat one way. If it doesn't get any heartbeat after one more interval, it will retry and wait for another interval before declaring the connection broken. This setting needs to balance the requirements between fast detection of dead connections and network traffic overhead. Note that if a daemon fails or is shut down, the closed socket will be detected immediately in many cases. The heartbeat-based detection is mostly only needed for errors in the network hardware itself (e.g. broken or disconnected cable, switch, port).

5.3.3 Environmental Buckets

The daemons are enabled to execute predefined environmental data collection. The execution is controlled in the configuration files in section *csm.data_collection* with can list a number of buckets each with a list of the predefined items. (Note that the current version of CSM only provides a GPU item that logs data into the daemon log file.) Each bucket has an execution interval.

5.3.4 Prolog/Epilog Scripts Compute

In order to create allocations both a ***privileged_prolog*** and ***privileged_epilog*** script must be present in */opt/ibm/csm/prologs/* on the compute nodes.

Review the sample scripts and make any customization required.

To use the packaged sample scripts (*/opt/ibm/csm/share/prologs/*) run the following on the management node:

```
$ xdcp compute /opt/ibm/csm/share/prologs/* /opt/ibm/csm/prologs/
```

This will copy the following files to the compute nodes:

- privileged_prolog (access: 700)
- privileged_prolog (access: 700)
- privileged.ini (access: 600)

The *privileged_prolog* and *privileged_epilog* files are python scripts that have command line arguments for type, user flags and system flags. The type is either *allocation* or *step,* and the flags are space delimited alpha-numeric strings of flags. If a new version of one of these scripts are written it must implement the options as below:

```
--type [allocation|step]

--user_flags "[string of flags, spaces allowed]"

--sys_flags "[string of flags, spaces allowed]"
```

The *privileged.ini* file configures the logging levels for the script. It is only needed if extending or using the packaged scripts. For more details see the comments in the bundled scripts, the packaged *POST_README.md* file or the ***Configuring allocation prolog and epilog scripts*** section in the *CSM User Guide*.

5.3.5 CSM PAM Module

The *ibm-csm-core* rpm does install a PAM module that may be enabled by a system administrator. This module performs two operations: prevent unauthorized users from obtaining access to a compute node and placing users who have active allocations into the correct cgroup.

To enable the CSM PAM Module for ssh sessions:

1. Add always authorized users to */etc/pam.d/csm/whitelist* (newline delimited).
2. Uncomment the following line from */etc/pam.d/sshd*:

```
session    required     libcsmpam.so
```

3. Restart the ssh daemon:

```
$ systemctl restart  sshd.service
```

Non root users who do not have an active allocation on the node and are not whitelisted will now be prevented from logging into the node via ssh. Users who have an active allocation will be placed into the appropriate cgroup when logging in via ssh.

For more details on the behavior and configuration of this module please refer to */etc/pam.d/csm/README.md* or the ***Configuring the CSM PAM module*** section in the *CSM User Guide*.

WARNING: The CSM PAM module should only be enabled on nodes that will run the CSM compute daemon, as ssh logins will be restricted to root and users specified in whitelist.

5.4 Start CSM Daemons

Before we start CSM daemon we need to start CSM daemons dependency:

Login, Launch and Compute node

Start NVIDIA persistence and DCGM

```
$ xdsh compute,service,utility "systemctl start nvidia-persistenced"
$ xdsh compute,service,utility "systemctl start dcgm"
```

Management node

Start the master daemon

```
$ systemctl start csmd-master
```

Start the aggregator daemon

```
$ systemctl start csmd-aggregator
```

Login and Launch node

Start the utility daemon

```
$ xdsh login,launch "systemctl start csmd-utility"
```

Compute node

Start the compute daemon

```
$ xdsh compute "systemctl start csmd-compute"
```

5.5 Run the Infrastructure Health Check

Run CSM infrastructure health check on the login / launch node to verify the infrastructure status:

```
# /opt/ibm/csm/bin/csm_infrastructure_health_check -v
Starting. Contacting local daemon...
Connected. Checking infrastructure... (this may take a moment. Please be patient...)

###### RESPONSE FROM THE LOCAL DAEMON #######
MASTER: c650mnp06.pok.stglabs.ibm.com (bounced=0; version=3645511)
        DB_free_conn_size: 10
        DB_locked_conn_pool_size: 0
        Timer_test: success
        DB_sql_query_test: success
        Multicast_test: success
        Network_vchannel_test: success
        User_permission_test: success
        UniqueID_test: success

Aggregators:4
    AGGREGATOR: c650f08p05 (bounced=0; version=3645511)
        Active_primary: 3
        Unresponsive_primary: 0
        Active_secondary: 3
        Unresponsive_secondary: 0

        Primary Nodes:
                Active: 3
                        COMPUTE: c650f08p13 (bounced=0; version=3645511; link=PRIMARY)
                        COMPUTE: c650f08p15 (bounced=0; version=3645511; link=PRIMARY)
                        COMPUTE: c650f08p17 (bounced=0; version=3645511; link=PRIMARY)
                Unresponsive: 0

        Secondary Nodes:
                Active: 3
                        COMPUTE: c650f08p23 (bounced=0; version=3645511; link=SECONDARY)
                        COMPUTE: c650f08p25 (bounced=0; version=3645511; link=SECONDARY)
                        COMPUTE: c650f08p27 (bounced=0; version=3645511; link=SECONDARY)
                Unresponsive: 0

    AGGREGATOR: c650f08p07 (bounced=0; version=3645511)
        Active_primary: 3
        Unresponsive_primary: 0
        Active_secondary: 3
        Unresponsive_secondary: 0

        Primary Nodes:
                Active: 3
                        COMPUTE: c650f08p23 (bounced=0; version=3645511; link=PRIMARY)
                        COMPUTE: c650f08p25 (bounced=0; version=3645511; link=PRIMARY)
                        COMPUTE: c650f08p27 (bounced=0; version=3645511; link=PRIMARY)
                Unresponsive: 0

        Secondary Nodes:
                Active: 3
                        COMPUTE: c650f08p13 (bounced=0; version=3645511; link=SECONDARY)
                        COMPUTE: c650f08p15 (bounced=0; version=3645511; link=SECONDARY)
                        COMPUTE: c650f08p17 (bounced=0; version=3645511; link=SECONDARY)
                Unresponsive: 0

    AGGREGATOR: c650f07p11 (bounced=0; version=3645511)
        Active_primary: 4
        Unresponsive_primary: 0
        Active_secondary: 4
        Unresponsive_secondary: 0
```

```
        Primary Nodes:
                Active: 4
                        COMPUTE: c650f07p15 (bounced=0; version=3645511; link=PRIMARY)
                        COMPUTE: c650f07p19 (bounced=0; version=3645511; link=PRIMARY)
                        COMPUTE: c650f07p17 (bounced=0; version=3645511; link=PRIMARY)
                        COMPUTE: c650f07p21 (bounced=0; version=3645511; link=PRIMARY)
                Unresponsive: 0

        Secondary Nodes:
                Active: 4
                        COMPUTE: c650f08p09 (bounced=0; version=3645511; link=SECONDARY)
                        COMPUTE: c650f08p11 (bounced=0; version=3645511; link=SECONDARY)
                        COMPUTE: c650f08p19 (bounced=0; version=3645511; link=SECONDARY)
                        COMPUTE: c650f08p21 (bounced=0; version=3645511; link=SECONDARY)
                Unresponsive: 0

    AGGREGATOR: c650f07p13 (bounced=0; version=3645511)
        Active_primary: 4
        Unresponsive_primary: 0
        Active_secondary: 4
        Unresponsive_secondary: 0

        Primary Nodes:
                Active: 4
                        COMPUTE: c650f08p19 (bounced=0; version=3645511; link=PRIMARY)
                        COMPUTE: c650f08p09 (bounced=0; version=3645511; link=PRIMARY)
                        COMPUTE: c650f08p11 (bounced=0; version=3645511; link=PRIMARY)
                        COMPUTE: c650f08p21 (bounced=0; version=3645511; link=PRIMARY)
                Unresponsive: 0

        Secondary Nodes:
                Active: 4
                        COMPUTE: c650f07p15 (bounced=0; version=3645511; link=SECONDARY)
                        COMPUTE: c650f07p19 (bounced=0; version=3645511; link=SECONDARY)
                        COMPUTE: c650f07p17 (bounced=0; version=3645511; link=SECONDARY)
                        COMPUTE: c650f07p21 (bounced=0; version=3645511; link=SECONDARY)
                Unresponsive: 0


  Unresponsive Aggregators: 0

Utility Nodes:1
    UTILITY: c650f07p09 (bounced=0; version=3645511)

  Unresponsive Utility Nodes: 0

Local_daemon: MASTER: c650mnp06.pok.stglabs.ibm.com (bounced=0; version=3645511)
        Status:
##########################################

Finished. Cleaning up...
Test complete: rc=0
```

Note that is some cases the list and status of nodes might not be 100% accurate if there were infrastructure changes immediately before or during the test. This usually results in timeout warnings and a rerun of the test should return an updated status.

Another important thing to note happens if there are any unresponsive compute nodes. First, unresponsive nodes will not show a daemon build version and will also not list the connection type as primary or secondary. Additionally, the unresponsive nodes are unable to provide info about their configured primary or secondary aggregator. Instead the aggregators report the last known connection status of those compute nodes. For example, if the compute node did use a

connection as the primary link even if the compute configuration defines the connection as secondary, the aggregator will show this compute as an unresponsive primary node.

5.6 Environment Setup for Job Launch

Use CSM API command line *csm_node_attributes_update* to update the compute nodes state to IN_SERVICE.

```
$ /opt/ibm/csm/bin/csm_node_attributes_update –s IN_SERVICE -n c650f99p08
```

5.7 CSM REST Daemon Installation and Configuration (Optional)

The CSM REST daemon is optional, but can be installed and configured on the service nodes. CSM REST daemon enables CSM RAS events to be created by IBM POWER HW Monitor for events detected from compute node BMCs. CSM REST daemon is only used for functional demonstration and is not required for normal cluster operation. It is recommended to enable CSM REST daemon and IBM POWER HW Monitor to start collecting BMC RAS events.

On the service nodes:

Install the ibm-csm-restd rpm if it is not already installed:

```
$ rpm -ivh ibm-csm-restd-1.0.0-*.ppc64le.rpm
```

Copy the default configuration file from */opt/ibm/csm/share/etc* to */etc/ibm/csm*:

```
$ cp /opt/ibm/csm/share/etc/csmrestd.cfg /etc/ibm/csm/
```

Edit /etc/ibm/csm/csmrestd.cfg and replace "__CSMRESTD_IP__" with "127.0.0.1".

The CSM REST daemon requires that the local CSM daemon is running before it is started. In addition, csmrestd must be restarted whenever the local CSM daemon is restarted.

Start csmrestd using systemctl:

```
$ systemctl start csmrestd
```

On the management node (optional):

If the CSM DB on your management node was not re-created as part of the CSM installation and you intend to enable IBM POWER HW Monitor collection of BMC RAS events from your service nodes, you can manually update the CSM RAS types in the CSM DB using the following process:

With all daemons stopped:

```
$ /opt/ibm/csm/db/csm_db_ras_type_script.sh -l csmdb csm_ras_type_data.csv
```

This will import any CSM RAS types into the CSM DB that were added on later releases, and it is a no-op for any events that already exist in the CSM DB.

6 Uninstallation

Stop all CSM daemons

```
$ xdsh compute "systemctl stop csmd-compute"
$ xdsh utility "systemctl stop csmd-utility"
$ systemctl stop csmd-aggregator
$ systemctl stop csmd-master
```

Delete CSMDB

```
$ /opt/ibm/csm/db/csm_db_script.sh -d csmdb
```

Remove rpms

```
$ xdsh compute,utility "rpm -e ibm-csm-core-1.0.0-*.ppc64le ibm-csm-api-1.0.0-*.ppc64le
ibm-flightlog-1.0.0-*.ppc64le ibm-csm-hcdiag-1.0.0-*.noarch"

$ rpm -e ibm-csm-core-1.0.0-*.ppc64le ibm-csm-hcdiag-1.0.0-*.noarch ibm-csm-db-1.0.0-
*.noarch ibm-csm-api-1.0.0-*.ppc64le ibm-csm-restd-1.0.0-*.ppc64le ibm-flightlog-1.0.0-
*.ppc64le
```

Clean up log and configuration files

```
$ xdsh compute,utility "rm -rf /etc/ibm /var/log/ibm"
$ rm -rf /etc/ibm/csm /var/log/ibm/csm
```

Stop NVIDIA host engine (DCGM)

```
$ xdsh compute,service,utility "systemctl start nvidia-persistenced"
$ xdsh compute,utility /usr/bin/nv-hostengine –t
```

7 Appendices

7.1 Diskless images

Please ensure the following steps have been completed on the xCAT Management node:

1. xCAT has been installed and the basic configuration has been completed.
2. Section 2.2 has been completed and the cast-boost rpms are currently accessible at **${HOME}/rpmbuild/RPMS/ppc64le**.
3. The **ibm-flightlog-1.0.0-*.ppc64le** is present on the xCAT Management node.
4. Install **createrepo** for building the other packages directory.

After verifying the above steps have been completed do the following:

1. Generate the osimages for your version of red hat:

```
$ copycds RHEL-7.3-Server-ppc64le-dvd1.iso
$ lsdef –t osimage
```

2. Copy the “netboot” image of the osimages created in the previous step and rename it:

```
$ image_input="rhels7.4-ppc64le-netboot-compute"
$ image_output="rhels7.4-ppc64le-diskless-compute"
$ lsdef -t osimage -z $image_input | sed -e "s/$image_input/$image_output/g" | mkdef -z
```

3. Move the cast-boost rpms to the **otherpkgdir** directory for the generation of the diskless image:

```
$ lsdef -t osimage rhels7.4-ppc64le-diskless-compute
$ cp cast-boost* /install/post/otherpkgs/rhels7.4/ppc64le/cast-boost
$ createrepo /install/post/otherpkgs/rhels7.4/ppc64le/cast-boost
```

4. Move the CSM rpms to the **otherpkgdir** directory:

```
$ cp csm_rpms/* /install/post/otherpkgs/rhels7.4/ppc64le/csm
$ createrepo /install/post/otherpkgs/rhels7.4/ppc64le/csm
```

5. Run creatrepo one last time:

```
$ createrepo /install/post/otherpkgs/rhels7.4/ppc64le
```

6. Add the following to a package list in the **otherpkgdir**, then add the package list to the osimage:

```
$ vi /install/post/otherpkgs/rhels7.4/ppc64le/csm.pkglist
       cast-boost/cast-boost-*
       csm/ibm-flightlog-1.0.0-*.ppc64le
       csm/ibm-csm-core-1.0.0-*.ppc64le
       csm/ibm-csm-api-1.0.0-*.ppc64le

$ chdef -t osimage rhels7.4-ppc64le-diskless-compute
otherpkglist=/install/post/otherpkgs/rhels7.4/ppc64le/csm.pkglist
```

7. Generate and package the diskless image:

```
$ genimage rhels7.4-ppc64le-diskless-compute
$ packimage rhels7.4-ppc64le-diskless-compute
```