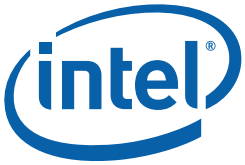

Node Manager IPMItool Test Scripts

User Guide

June 2009

Revision 0.3

Intel Confidential

Document Number: 102-020-GUD

Contents

Tables

Revision History

Document Number	Revision Number	Description	Revision Date
102-020-GUD	0.1	Initial release.	March 2009
102-020-GUD	0.2	Changes in scripts nomenclature.	May 2009
102-020-GUD	0.3	Added support for KCS	June 2009

§

1 Introduction

The purpose of this document is to describe the Node Manager IPMItool test scripts. This document helps to understand impact of these tests to the Node Manager base system.

1.1 System Requirements

- Linux system
- Bash Shell
- IPMItool (http://ipmitool.sourceforge.net/)

Install a Linux system. If you do not have the IPMItool, download it from http://ipmitool.sourceforge.net/.

1.2 Test Environment

1.2.1 Script Parameters

There are a number of mandatory parameters that must be set in order to run tests properly. The following variables are declared in SPSlib file.

Table 1. Script Parameters

Heading Cells	Variable	Default Value
ME address	ME_ADDRRESS	0x88
BMC address	BMC_IP	172.28.191.21
Network Function code	NM_NET_FUNC	0x2E
Network Function code	PECI_NET_FUNC	0x30
Debug level	DEBUG_LEVEL	7
Use KCS transport	USEKCS	0

The default values should be changed to meet those of the test system.

ME_ADDRESS

IPMI Address of the Manageability Engine (ME) used to send IPMI messages by BMC to ME.

BMC_IP

IP Address of the Baseboard Management Controller. The BMC provides access to the ME through standard IPMI bridging. The BMC forwards commands to the ME and responses back to the originator (acting as a transport conversion device).

NM_NET_FUNC

IPMI net function used in Node Manager IPMI commands. The default value is 2Eh (OEM/Non IPMI group request).

PECI_NET_FUNC

IPMI net function used in General Application IPMI commands. The default value is 30h (SDK General Application IPMI group request).

DEBUG_LEVEL

Table 2. Debug Levels

Debug Level	Description
10	Results of low Command function
9	Results of SendIPMICmd function
8	Results of SendNMCmd function
7	Request description
6	Response description
5	Names of executed functions
4	Names of executed Scripts
0	Errors

The debug level blocks all echo messages. The chosen debug level 6 allows all messages of that level and lower. Errors are displayed at debug level 0 and cannot be blocked. You can run scripts in debug mode by simply adding the debug level parameter after the name of the script. For example:

```
./NmPolicyControl.sh 8
```

If you do not specify any debug level, the script will be executed with default debug level defined by the DEBUG_LEVEL parameter in SPSLlib file.

USEKCS

Use KCS transport set to 1 change the default communication LAN channel to KCS. Use of this transport requires start of KCS driver under Linux. It can be done in this way:

```
chkconfig ipmi on

service ipmi start
```

1.2.2 IPMItool Parameters

The IPMItool is used to communicate with the BMC and ME. By default the IPMItool is run with the following parameters:

```
rsp=$($IPMITOOL1 -P "$PASSWORD" $IPMITOOL2 $1)

IPMITOOL1="ipmitool -I lan -H $BMC_IP -A PASSWORD"

IPMITOOL2="-b 06 -t $ME_ADDRESS"

PASSWORD=""
```

Table 3. IPMItool Parameters

Parameter	Description
-I lan	Lan is selected as an IPMI interface
-H $BMC_IP	BMC_IP variable defines address of the remote server. This option is required if you use lan or lanplus interfaces.
-A PASSWORD	Password authentication type is used
-b 06	channel 6 is a destination channel for a bridge request
-t $ME_ADDRESS	IPMI requests are bridged to the remote target with $ME_ADDRESS address.

The user can run IPMItool with a different set of parameters than the default one. For more information on how to use the IPMItool, see the IPMItool manual.

1.3 Script Usage

Copy the script files to any folder on your PC. You can run the scripts by simply typing their names (with path) in a Linux console. For example:

```
./NmVersion.sh
```

Scripts communicate with a user by sending messages to the system console. Different types of messages have a different font color.

Table 4. Font Colors

Message Type	Font Color
Error, "test failed" messages	red
script name	blue
"test passed" messages	blue
function name	green
other types of messages	white, yellow

1.4 Script Files Included in the Package

Table 5. Script Files

File	Description
GetDeviceId.sh	Shows how to obtain device ID
HostCpuData.sh	Shows how to use the `Set Host CPU Data` function
MaxAllowedCpuStates.sh	Shows how to use the `Set Max Allowed CPU P-State/T-State` and `Get Max Allowed CPU P-State/T-State` functions.
NmAlertDestination.sh	Shows how to use the `Set Node Manager Alert Destination` and `Get Node Manager Alert Destination` functions.
NmAlertThresholds.sh	Shows how to use the `Set Node Manager Alert Thresholds` and `Get Node Manager Alert Thresholds` functions.
NmBasicTests.sh	Runs all other scripts except `TestScenario1`
NmPolicyControl.sh	Shows how to use the `Enable/Disable Node Manager Policy Control`, `Set Node Manager Policy`, `Get Node Manager Policy` functions
NmPolicySuspendPeriods.sh	Shows how to use `Set Node Manager Policy Suspend Periods`, `Get Node Manager Policy Suspend Periods` functions.
NmPowerDrawRange.sh	Shows how to use the `Set Node Manager Power Draw Range` function.
NmStatistics.sh	Shows how to use the `Reset Node Manager Statistics` and `Get Node Manager Statistics` functions.
NmVersion.sh	Shows how to use the `Get Node Manager Version` function.
NumberOfCpuStates.sh	Shows how to use the `Get Number Of P-States/T-States` function.
PowerBudget.sh	Shows how to use the `Set Total Power Budget` and `Get Total Power Budget` functions.
PsuConfiguration.sh	Shows how to use the `Set PSU Configuration` and `Get PSU Configuration` functions.
TestScenario1.sh	Shows how to limit the Power consumption by using Node Manager functions
LoopTest.sh	Executes the chosen command a specified number of times and generates log file.

§

2 GetDeviceId.sh

This script shows how to execute the “Get Device Id” command, which can be useful to obtain the information about the current ME firmware. The Get Device Id command response provides the following information: firmware version, device and firmware revision, type of currently active image, additional supported functionalities.

2.1 Getting the Device ID

To obtain the information about the current ME firmware, send the `Get Device Id` IPMI command.

```
GetDeviceId
```

3 NmPolicyControl.sh

This script shows how to set, use and remove a Node Manager Policy.

3.1 Setting and Enabling Node Manager Policy

To set and enable the Node Manager Policy perform the following steps:

1. Send the `Disable Node Manager Policy Control` command.

   Before setting a new Node Manager Policy Control the old one should be disabled.

```
##### Disable Node Manager Policy Control #####
b4=0 # Global Disable Node Manager Policy control
EnDisNMPolicyControl "0x$b4 0x$domainId 0x$policyId"
if [ $error -eq 0 ];then
Print " Global Disable Node Manager policy control" 5
Print " " 5
fi
################################################
```

2. Set the Node Manager Policy parameters

   It is a good practice to check if values to be set are in the correct range. The `NMpolicyControl` script uses the `GetNMCapabilities` command to obtain information about system performance.

```
##### Get Node Manager Capabilities #####
# Policy Type | Policy Trigger Type
  b5=10 # Power Control Policy | No Policy Trigger
GetNmCapabilities "0x$domainId 0x$b5"
########################################
```

   In result a set of variables $c6, … are created. (Variables obtained in result of the `GetNMCapabilities` command are described in the `SPSLib`). The script chooses the minimum values of Power Limit, Correction Time and Statistics reporting period for the parameters in the `Set Node Manager Policy` command.

```
##### Set Node Manager Policy                    #####
#
#  Domain Id | Policy Enabled
   sb4=$(( $domainId | $(( 0x10 ))  )) # policy is enabled by default
#during policy creation/modification
#  Policy Type | Policy Trigger Type
   sb6=10 # Power Control Policy| No Policy Trigger
#  Policy Exception Actions

   sb7=0 # Send alert
#  Power Limit[Watts]:
   sb8=$c8
   sb9=$c9
#  Correction time Limit[ms]:
   sb10=$c10
   sb11=$c11
   sb12=$c12
```

```
   sb13=$c13
#  Policy Trigger Limit[Celsius]:
   sb14=56
   sb15=0 # Policy trigger Value will be ignored
#  Statistics Reporting Period[s]:
   sb16=$c18
   sb17=$c19
```

3. Send the `Set Node Manager Policy` command

```
SetNMPolicy "$sb4 0x$policyId 0x$sb6 0x$sb7 0x$sb8 0x$sb9 0x$sb10
0x$sb11 0x$sb12 0x$sb13 0x$sb14 0x$sb15 0x$sb16 0x$sb17 "
```

4. To check if all the parameters have been set properly execute the `Get Node Manager Policy` command.

   Next part of script compares the results with expected values. In result of execution of the `Get Node Manager Policy` command we obtain $b6 $b7 $b8 $b9 $b10 $b11 $b12 $b13 $b16 $b17 variables that can be compare with variables $sb6 $sb7 $sb8 $sb9 $sb10 $sb11 $sb12 $sb13 $sb16 $sb17 that we used with the `Set Node Maneger Policy` command.

```
GetNMPolicy "0x$domainId 0x$policyId"
stab=($sb6 $sb7 $sb8 $sb9 $sb10 $sb11 $sb12 $sb13 $sb16 $sb17)
tab=($b6 $b7 $b8 $b9 $b10 $b11 $b12 $b13 $b16 $b17)
num=10 # number of tab elements
tabError=0
if [ $error -eq 0 ];then
    if [ $(( $sb4 & 0x10 )) -ne $(( 0x$b5 & 0x10 )) ]; then
     tabError=1
    fi
    if [ $(( $sb6 & 0x01 )) -eq 1 ]; then # if Inlet temperature
Limit is Enabled
     if [ $(( 0x$sb15$sb14 )) -ne $(( 0x$b15$b14)) ];then # check if
Trigger Limit is set correct
          tabError=1
     fi
    fi
    index=0
    while [ $index -lt $num ];do
     if [ $(( 0x${tab[$index] } )) -ne $(( 0x${stab[$index] } ))
];then
          tabError=1
     fi
     let "index = index + 1"
    done
fi
if [ $tabError -eq 0  ];then
    Print " Policy $policyId for domain $domainId has been set" 5
    Print " " 5
else
    Print " Unable to set policy $policyId for domain $domainId" 0 31
    Print " " 0
    scriptError=1
fi
#######################################################
```

   If any of parameter is not set correctly the tabError flag is set and the script generates message:

   `Unable to set policy $policyId for domain $domainId`

5. Send the `Enable Manager Policy Control` command

After a new policy is created it could be enabled. The next few lines show how to enable a new policy and check if any error has occurred.

```
##### Enable Node Manager Policy Control #####
#
b4=3 # Per Domain Enable Node Manager Policy control
EnDisNMPolicyControl "0x$b4 0x$domainId 0x$policyId"
if [ $error -eq 0 ];then
    Print " Policy for domain $domainId has been enabled" 5
    Print " " 5
fi
b4=1 # Global Enable Node Manager Policy control
EnDisNMPolicyControl "0x$b4 0x$domainId 0x$policyId"
if [ $error -eq 0 ];then
    Print " Global Enable Node Manager policy control" 5
    Print " " 5
fi
```

3.2 Removing Node Manager Policy:

To remove Node Manager Policy perform the following steps:

6. Send the `Disable Node Manager Policy` Command

   After execution of this command the policy will be disabled and ready to remove.

```
#### Disable Node Manager Policy Control ##
b4=2 #  Per Domain Disable Node Manager policy control
EnDisNMPolicyControl "0x$b4 0x$domainId 0x$policyId"
if [ $error -eq 0 ];then
    Print " Node Manager policy control for domain $domainId has been
disabled" 5
    Print " " 5
fi
b4=0 #  Global Disable Node Manager policy control
EnDisNMPolicyControl "0x$b4 0x$domainId 0x$policyId"
if [ $error -eq 0 ];then
    Print " Global Disable Node Manager policy control" 5
    Print " " 5
fi
##############################################
```

7. Send Set Node Manager Policy command

   This command will remove the policy.

```
SetNMPolicy "0x$domainId 0x$policyId 0x00 0x00 0x$c8 0x$c9 0x$c10
0x$c11 0x$c12 0x$c13 0x00 0x00 0x$c18 0x$c19 "
if [ $error -eq 0 ]; then
    Print "Policy $policyId for domain $domainId has been removed" 5
    Print " " 5
fi
```

   At the beginning of the script variable $scriptError is set to zero.

   If any error occur the variable “$scriptError” will be set automatically to one. To check if there were any errors during execution of the script you have to test if the “$scriptError” variable is equal to 1

```
if [ $scriptError -eq 1 ];then
```

```
    Print " " 0
    Print "TEST FAILED " 0 31
    Print " " 0
    exit 1
else
    Print " " 4
    Print "TEST PASSED " 4 34
    Print " " 4
fi
```

4 NmAlertThresholds.sh

This script shows how to set, use and remove the Node Manager Alert Thresholds

4.1 Setting the New Node Manager Alert Thresholds

To set the new Node Manager Alert Thresholds perform the following steps:

1. Send the `Set Node Manager Alert Thresholds` command

   Following command sets 2 new Alert Thresholds. The max thresholds the user can set is 3.

```
##### Set Node Manager Alert Thresholds ####
#
#  Number of alert thresholds
   num=2 # Number of alert thresholds
#  Alert threshold array:
   sb7=$c8  # Alert threshold number 1: b8b7
   sb8=$c9
   sb9=$c6  # Alert threshold number 2:
   sb10=$c7
#   sb11=$c8 # Alert threshold number 3:
#   sb12=$c9
#
SetNMAlertThresholds "0x$domainId 0x$policyId 0x$num 0x$sb7 0x$sb8
0x$sb9 0x$sb10 "
```

2. Check if the new alert thresholds have been set properly.

   The script checks if a response is compliant with a request.

```
GetNMAlertThresholds "0x$domainId 0x$policyId"
stab=($sb7 $sb8 $sb9 $sb10 $sb11 $sb12)
tab=($b6 $b7 $b8 $b9 $b10 $b11)
if [ $error -eq 0 ]; then
    if [ $num -eq $b5 ] ;then
     if [ $num -gt 0 ] ;then
         temp=$(( 2 * $num))
         index=0
         tabError=0
         while [ $index -lt $temp ];do
              if [ $(( 0x${tab[$index]} )) -ne  $(( 0x${stab[$index]}
)) ];then
                  tabError=1
              fi
              let " index = index + 1"
         done
         if [ $tabError -eq 0 ];then
              Print " Alert Thresholds for policy $policyId domain
$domainId have been set" 5
              Print " " 5
         else
              Print " Unable to set Alert Thresholds for policy
$policyId domain $domainId" 0 31
```

```
                Print " " 0
                scriptError=1
            fi
        else
            Print " Alert Thresholds for policy $policyId domain
$domainId have been set" 5
            Print " " 5
        fi
       else
        Print " Unable to set Alert Thresholds for policy $policyId
domain $domainId" 0 31
        Print " " 0
        scriptError=1
       fi
fi
##########################################
```

4.2 Removing the Node Manager Alert Thresholds

To Remove the Node Manager Alert Thresholds perform the following steps:

1. Send the `Set Node Manager Alert Thresholds` command

   The third parameter (number of alert thresholds) should be set to zero.

   ```
   num=0
   SetNMAlertThresholds "0x$domainId 0x$policyId 0x$num"
   ```

2. Send the `Get Node Manager Alert Thresholds` command to check if the Node Manager Alert Thresholds are removed properly

   ```
   GetNMAlertThresholds "0x$domainId 0x$policyId"
   if [ $error -eq 0 ];then
       if [ $num -eq $b5 ] ;then
        Print " Alert thresholds for policy $policyId and domain
   $domainId have been removed" 5
        Print " " 5
       else
        Print " Unable to remove Alert Thresholds" 0 31
        Print " " 0
        scriptError=1
       fi
   fi
   ```

5 NmPolicySuspendPeriods.sh

This script shows how to set, use and remove the Node Manager Policy Suspend Periods. The suspend periods indicate when the policy shall not be active during the week.

5.1 Setting the Node Manager Policy Suspend Periods

To set the Node Manager Policy Suspend Periods send the `Set Node Manager Policy Suspend Periods` command.

```
##### Set Node Manager Policy suspend Periods ####
#
#  Number of policy suspend periods(max 5):
   num=5 #  1 policy suspend periods
#  Array of policy suspend periods
#       Policy suspend period number 1:
#                Policy suspend start time:
                 sb7=2
#                Policy suspend stop time:
                 sb8=8
#                Suspend period recurrence pattern:
                 sb9=3 # Monday | tuesday
#       Policy suspend period number2:
#                Policy suspend start time:
                 sb10=3
#                Polic suspend stop time:
                 sb11=8
#                Suspend period recurrence pattern:
                 sb12=7 # Monday | Tuesday | Wednesday
#       Policy suspend period number 3:
#                Policy suspend start time:
                 sb13=4
#                Policy suspend stop time:
                 sb14=12
#                Suspend period recurrence pattern:
                 sb15=40  #Sunday
#       Policy suspend period number 4:
#                Policy suspend start time:
                 sb16=15
#                Policy suspend stop time:
                 sb17=19
#                Suspend period recurrence pattern:
                 sb18=9  # Thursday | Monday
#       Policy suspend period number 5:
#                Policy suspend start time:
                 sb19=2
#                Policy suspend stop time:
                 sb20=8
#                Suspend period recurrence pattern:
                 sb21=8 # Thursday
#
```

```
SetNMPolicySuspendPeriods "0x$domainId 0x$policyId 0x$num 0x$sb7 0x$sb8
0x$sb9 0x$sb10 0x$sb11 0x$sb12 0x$sb13 0x$sb14 0x$sb15 0x$sb16 0x$sb17
0x$sb18 0x$sb19 0x$sb20 0x$sb21"
```

5.2 Checking if the New Policy Suspend Periods are Set Correctly

To check if the new Policy Suspend Periods are set correctly perform the following steps:

1. Send the `Get Node Manager Policy Suspend Periods` command

   ```
   GetNMPolicySuspendPeriods " 0x$domainId 0x$policyId"
   ```

2. Execute the following code

   ```
   stab=($sb7 $sb8 $sb9 $sb10 $sb11 $sb12 $sb13 $sb14 $sb15 $sb16 $sb17
   $sb18 $sb19 $sb20 $sb21)
   tab=($b6 $b7 $b8 $b9 $b10 $b11 $b12 $b13 $b14 $b15 $b16 $b17 $b18
   $b19 $b20)
   if [ $error -eq 0 ]; then
       if [ $num -eq $b5 ] ;then
        if [ $num -gt 0 ] ;then
            temp=$(( 3 * $num))
            index=0
            tabError=0
            while [ $index -lt $temp ];do
                 if [ $(( 0x${tab[$index]} )) -ne  $(( 0x${stab[$index]}
   )) ];then
                     tabError=1
                 fi
                 let " index = index + 1"
            done
            if [ $tabError -eq 0 ];then
                 Print " Node Manager Policy Suspend Periods for policy
   $policyId domain $domainId have been set" 5
                 Print " " 5
            else
                 Print " Unable to set Node Manager Policy Suspend
   Periods" 0 31
                 Print " " 0
                 scriptError=1
            fi
        else
            Print " Node Manager Policy Suspend Periods for policy
   $policyId domain $domainId have been set" 5
            Print " " 5
        fi
       else
        Print " Unable to set Node Manager Policy Suspend Periods" 0 31
        Print " " 0
        scriptError=1
       fi
   fi
   ####################################################
   ```

5.3 Removing the Node Manager Policy Suspend Periods

To remove the Node Manager Policy Suspend Periods send the `Set Node Manager Policy Suspend Periods` command with the num parameter set to zero.

```
num=0
SetNMPolicySuspendPeriods "0x$domainId 0x$policyId 0x$num"
if [ $error -eq 0 ]; then
    Print "Suspend Periods for policy $policyId and domain $domainId has
been removed" 5
    Print " " 5
fi
```

6 NmStatistics.sh

This script shows how to obtain the different kinds of the Node Manager Statistics.

6.1 Obtaining the Node Manager Statistics

To obtain the Node Manager Statistics perform the following steps:

1. Send the Reset Node Manager Statistics command

```
##### Reset Node Manager Statistics #####
#
#  Mode
   mode=0 # Reset global statistics including power statistics and
inlet temperature statistics
#
ResetNMStatistics "0x$mode 0x$domainId 0x$policyId "
########################################
```

2. Send the Get Node Manager statistics command

```
##### Get Node Manager Statistics ####
#
#  Mode
   mode=1 # Global power statistics
GetNMStatistics "0x$mode 0x$domainId 0x$policyId"
#####################################
```

7 NmVersion.sh

This script shows how to receive the information about the Node Manager version.

7.1 Obtaining the Node Manager Version

To obtain the Node Manager Version send the `Get Node Manager Version` command:

```
GetNMVersion
```

8 NmPowerDrawRange.sh

This script shows how to set the Node Manager Power Draw Range that specifies the Min/Max allowed power consumption ranges.

8.1 Setting the Node Manager Power Draw Range

To set the Node Manager Power Draw Range send the `Set Node Manager Power Draw Range` command:

```
##### Set Node Manager Power Draw Range ####################
#
#  Minimum power Draw: b6b5  [Watts]
   b5=$c8
   b6=$c9
#  Maximum power Draw: b8b7[Watts]
   b7=$c6
   b8=$c7
#
#############################################################
SetNMPowerDrawRange " 0x$domainId 0x$b5 0x$b6 0x$b7 0x$b8 "
```

9 NmAlertDestination.sh

The Set Node Manager Alert Destination IPMI command provides the alert destination information for the Node Manager to send direct alerts that bypass the BMC SEL. This script shows how to set the Node Manager Alert Destination.

9.1 Setting the Node Manager Alert Destination

To set the Node Manager Alert Destination send the `Set Node Manager Alert Destination` command:

```
##### Set Node Manager Alert Destination ####
#
#  Channel number: BMC channel number over which to send the alert from
BMC to management console | Destination information operation:
   sb4=1 # BMC channel number:1 | Destination infomration operation:
register alert receiver
#  Destination information:
#    for channel medium IPMB /  for channel medium 802.3 LAN
   sb5=0 # destiantion selector: use volatile destination info
#  Alert String Selector
   sb6=0 # use volatile Alert String | don't send Alert String
#
SetNMAlertDestination " 0x$sb4 0x$sb5 0x$sb6 "
```

10 PowerBudget.sh

This script shows how to set and remove the Total Power Budget. Setting the Total Power Budget different than zero causes the ME firmware to start to control the P/T states.

10.1 Setting the New Power Budget

To set the new Power Budget perform the following steps:

1. Send the `Disable Node Manager Policy Control` command

   It is mandatory to disable Node Manager Policy Control before power Budget control could be set.

```
##### Disable Node Manager Policy Control #####
b4=0 # Global Disable Node Manager Policy control
EnDisNMPolicyControl "0x$b4 0x$domainId 0x$policyId"
if [ $error -eq 0 ];then
    Print " Global Disable Node Manager policy control" 5
    Print " " 5
fi
```

2. Send the `Set Total Power Budget` command

```
##### Set Total Power Budget #####
#
#  Target power budget in Watts that should be maintained by the
Power Budget control Service
   PowerBudget_b5=$c8
   PowerBudget_b6=$c9
#
SetTotalPowerBudget " 0x$domainId 0x$PowerBudget_b5 0x$PowerBudget_b6
"
```

3. Send the `Get Total Power Budget` command

```
GetTotalPowerBudget " 0x$domainId "
if [ $((0x$PowerBudget_b5)) -eq $((0x$b5)) ] && [
$((0x$PowerBudget_b6)) -eq $((0x$b6 )) ];then
    Print " Total Power Budget has been set" 5
    Print " " 5
else
    Print " Unable to set Power Budget" 0 31
    Print " " 0
    scriptError=1
fi
###################################
```

10.2 Removing the Total Power Budget

To remove Total Power Budget execute the following code:

```
SetTotalPowerBudget " 0x$domainId 0x00 0x00 " # Removing Power Budget
GetTotalPowerBudget " 0x$domainId "
if [ 0 -eq $((0x$b5)) ] && [ 0 -eq $((0x$b6)) ];then
    Print " Total Power Budget has been removed" 5
    Print " " 5
else
    Print " Unable to remove Total Power Budget" 0 31
    Print " " 0
    scriptError=1
fi
```

11 MaxAllowedCpuStates.sh

This script shows how to set the current P-State/T-State.

11.1 Setting the Max Allowed CPU States:

To set the Max allowed CPU States perform the following steps:

1. Send the `Disable Node Manager Policy Control` command.

   It is mandatory to disable Node Manager Policy Control before Max Allowed CPU States could be set

```
#### Disable Node Manager Policy Control ##
b4=0 #  Global Disable Node Manager policy control
EnDisNMPolicyControl "0x$b4 0x$domainId 0x$policyId"
if [ $error -eq 0 ];then
    Print " Global Disable Node Manager policy control" 5
    Print " " 5
fi
```

2. Send the `Set Total Power Budget` command with a second and a third parameter set to zero.

   It is also mandatory to disable the Total Power Budget Control before the Max Allowed CPU States could be set.

```
##### Set Total Power Budget #####
#
#  Target power budget in Watts that should be maintained by the
Power Budget control Service
   PowerBudget_b5=0 # Issuing SetTotalPowerBudget command with target
budget set to 0 disables the Power Budget Control
   PowerBudget_b6=0 # and allows BMC to control P-States directly
#
SetTotalPowerBudget "0x$domainId 0x$PowerBudget_b5 0x$PowerBudget_b6"
if [ $error -eq 0 ];then
    Print " Power Budget Control has been disabled" 5
    Print " " 5
fi
```

3. Send the `Set Max Allowed CPU P-State/T-State` command

```
##### Set Max Allowed CPU P-State/T-State #####
#
#  P-State number to be set
   PStateNum=$(( 0x$b5 - 1 )) # decimal value
#  T-State number to be set
   TStateNum=$(( 0x$b6 - 1 )) # decimal value
#
SetMaxAllowedCPUPStateTState "0x$domainId $PStateNum $TStateNum"
GetMaxAllowedCPUPStateTState "0x$domainId"
if [ $PStateNum -eq $((0x$b5)) ] && [  $TStateNum -eq $((0x$b6))
];then
```

```
    Print " Max Allowed CPU P-State/T-State has been set " 5
    Print " " 5
else
    Print " Unable to set Max Allowed CPU P-State/T-State " 0 31
    Print " " 0
    scriptError=1
fi
```

12 NumberOfCpuStates.sh

This script shows how to get the number of supported P-States/T-States.

12.1 Getting the Number of CPU States

To get the Number of CPU States send the `GetNumber Of P-States/T-States` command:

```
GetNumberOfPStatesTStates "0x$domainId"
```

13 HostCpuData.sh

This script shows how to acquire and change the CPU configuration data, stored in ME.

13.1 Setting the Host CPU Data

To set THE Host CPU Data send the `set Host CPU Data` command:

```
##### Set Host CPU Data ####
#
#  Host CPU data
   b5=80 # End of POST notification
#  Number of P-States
   b6=$NumOfPStates
#  Number o T - States
   b7=$NumOfTStates
#  Number of installed CPUs/socket
   b8=1
#  Processor Discovery Data for the lowest number processor in LSByte-
first order
#       Turbo power current Limit MSR 1ACh for the lowest number
processor passed by BIOS
        b9=0
        b10=0
        b11=0
        b12=0
        b13=0
        b14=0
        b15=0
        b16=0
#       Platform info MSR 0CEh for the lowest number processor passed by
BIOS
        b17=0
        b18=0
        b19=0
        b20=0
        b21=0
        b22=0
        b23=0
        b24=0
#  Icc_tdc reading from PECI for the lowest number processor
        b25=0 # SPS Firmware should querry Icc_tdc using 'OEM get
Reading' with type "Icc_tdc reading from PECI"

SetHostCPUdata "0x$domainId 0x$b5 0x$b6 0x$b7 0x$b8 0x$b9 0x$b10 0x$b11
0x$b12 0x$b13 0x$b14 0x$b15 0x$b16 0x$b17 0x$b18 0x$b19 0x$b20 0x$b21
0x$b22 0x$b23 0x$b24 0x$b25"
```

14 PsuConfiguration.sh

This script shows how to change the PSU configuration.

14.1 Setting the PSU Configuration

To set the PSU Configuration send the `Set PSU Configuration` command:

```
##### Set PSU Configuration #####
#
# PMBUS PSU address 1 | Mode
  sb5=b0 # the PSU is installed or may be attached in the future; address
is not used
# PMBUS PSU address 2 | Mode
  sb6=2
# PMBUS PSU address 3 | Mode
  sb7=4
# PMBUS PSU address 4 | Mode
  sb8=3
# PMBUS PSU address 5 | Mode
  sb9=8
# PMBUS PSU address 6 | Mode
  sb10=10
# PMBUS PSU address 7 | Mode
  sb11=14
# PMBUS PSU address 8 | Mode
  sb12=0
SetPSUConfiguration "0x$domainId 0x$sb5 0x$sb6 0x$sb7 0x$sb8 0x$sb9
0x$sb10 0x$sb11 0x$sb12"
```

15 NmBasicTests.sh

Sequentially runs all the other scripts. The script will be terminated if an error occurs during execution of any of the script.

16 TestScenario1.sh

This is an example of a basic test scenario that demonstrates how the Node Manager limits power consumption. After seeing a “Start CPU load and press enter” message run your CPU loading tool.

16.1 Scenario description

The script performs the following tasks:

1. Test environment initialization

   The task is performed to disable all limiting activities before performing the test.

2. Waiting for load

   The user should start the cpu and memory load and press enter.

3. Resetting and Acquiring the statistics

   The task is performed to obtain the information about the average power consumption before the limiting is enabled.

4. Creating and enabling the policy

   The task is performed to start the power consumption limiting. The power limit for policy is calculated by the following equation:

   temp=$(($(($AveragePower1)) - 15))

   AveragePower1 – power obtained form statistics in task 3.

   The power limit specified for the policy is 15 watts less than the average power consumption obtained in task 3.

5. Resetting and acquiring the statistics

   The task is performed to obtain the information about the current power consumption.

6. Comparing the power consumption

   The task is performed to compare the amount of the power consumption before and after enabling of the power consumption limiting. Power limit should be reduced by about 15 watts

17 LoopTest.sh

The script executes the chosen script a specified number of times.

17.1 Running the LoopTest.sh script

1. Syntax:

   ```
   ./LoopTest.sh $numOfTests $logFile $command $debugLevel
   ```

Parameters Description:

- `numOfTests` – the number of `$command` invokes
- `logFile` – a name of the log file
- `command – a name of the script to invoke`
- `debuglevel` – specifies a debug level of the invoked command

2. Example of LoopTest.sh script usage:

   In the following example the `NMVersion.sh` script is executed 10 times with a debug level set to 6. The log from the `NmVersion.sh` script execution is written to the `TestResults file.`

   ```
   ./LoopTest.sh 10 ./TestResults ./NmVersion.sh 6
   ```

18 SendCommand.sh

The script allows to execute the chosen function from the SPSLib library, directly from the Command Line Terminal.

18.1 Running the SendCommand.sh script

3. Syntax:

```
./SendCommand.sh FunctionName DebugLevel Parameters
```

Parameters Description:

- `FunctionName` – the name of the function to invoke
- `DebugLevel` – the debug level to set
- `Parameters` – string that include the arguments passed to the chosen function

4. Example of SendCommand.sh script usage:

   In the following example the `GetNMCapabilities` function from the SPSLib library is executed directly from the Linux Command Line Terminal with a debug level set to 10 and parameters “0x01 0x00 0x10 “

   Type into the command line:

```
./SendCommand.sh GetNMCapabilities 10 “0x00 0x10”
```

   You can achieve the same result typing into the command line the following

```
FunctionName=GetNMCapabilities
DebugLevel=10
Parameters=’”0x00 0x10”’

eval ./SendCommand.sh $FunctionName $DebugLevel $Parameters
```