Out of Band procedures
for
bullx R gen 3 series

BIOS and BMC IPMI firmware updates

frederic.temporelli@bull.net
Tel : +33-4-76-29-75-82

Content

1 Changes 3
2 Purpose of this document 4
3 Ressources 5
3.1 Required packages 5
3.2 lUpdate Syntax 5
3.3 SMCIPMITool syntax 6
3.4 sum syntax 7
3.5 Rules common to all the OOB tools 9
4 BMC firmware update 10
5 BMC and OOB feature 11
5.1 OOB activation key 11
5.2 Check if OOB is enabled 11
5.2.1 Check if OOB is enabled – “sum” command 11
5.2.2 Check if OOB is enabled – “SMCIPMITool” command 12
5.3 Collect the BMC MAC addresses for the activation key 12
5.4 BMC activation and keys file 13
6 BIOS operations 14
6.1 Check the BIOS image 14
6.1.1 Check the BIOS image with sum 14
6.1.2 Check the BIOS image with SMCIPMITool 14
6.2 Flash the BIOS 15
6.2.1 Flash the BIOS with “sum” command 15
6.2.2 Flash the BIOS with “SMCIPMITool” command 16
7 Parallel updates with OOB utilities 17
7.1 Parallel BMC reset 17
7.2 Parallel BMC IPMI firmware update 17
7.3 Parallel BIOS update 18

1 Changes

Rev 1.2 :

- document has been renamed (previous name: OOB Tools - Usage - Rev 1.1.docx)
- smcibu is now replaced by sum
- parallelized commands based on xargs
- new page template
- improved examples (scripts, commands, syntax)

Rev 1.3 :

- considerations about « -reboot » flag when upgrading the BIOS
- updated tools (sum, SMCIPMITool)
- table of contents

2 Purpose of this document

"Out of Band" (OOB) feature is the ability to remotely manage some components of a remote system components, from the management node, through the Baseboard Management Controler (BMC), without having to run any remote command on the managed system.

This document is describing the commands to be run from the management node for upgrading the BMC and the BIOS of the remote bullx R gen3 ("-E3") systems with the Out of Band feature. These operations can be done even if the remote system is down, and if the system is not fully installed (no processor or processors not compatibles with the current BIOS level, no memory, broken BIOS, …). The only restriction is to have at least one power cable plugged and the BMC already configured for being available from the network (Network cable installed, static or DHCP configuration)

3 Ressources

3.1 Required packages

You need the following packages :

- **lUpdate (release 1.21 of later) :** IPMI Firmware Utility
- **sum (release 1.2.0 or later) :** Supermicro Update Manager
- **SMCIPMITool (release 2.6.5 or later) :** Supermicro IPMI Tool (JRE Bundle).

Note: SMCIPMITool 2.6.2.1 is provided with sum 1.2.0

*Optional: for parallel processing, you may use the **xargs** command. For RedHat 6 systems, **xargs** is provided in the **findutils** package.*

3.2 lUpdate Syntax

```
**************************************************************************
* ATEN Technology, Inc.                                                  *
**************************************************************************
* FUNCTION   :  IPMI FIRMWARE UPDATE UTILITY                             *
* VERSION    :  1.21                                                     *
* BUILD DATE :  Dec  3 2012                                              *
* USAGE      :                                                           *
*              (1)Update FIRMWARE : lUpdate -f filename.bin [OPTION]     *
*              (2)Dump FIRMWARE   : lUpdate -d filename                  *
*              (3)Restore CONFIG  : lUpdate -c -f filename.bin           *
*              (4)Backup CONFIG   : lUpdate -c -d filename.bin           *
**************************************************************************
* OPTION                                                                 *
*   -i the IPMI channel, currently, kcs and lan are supported            *
* LAN channel specific arguments                                         *
*   -h remote BMC address and RMCP+ port, (default port is 623)          *
*   -u IPMI user name                                                    *
*   -p IPMI password correlated to IPMI user name                        *
*   -r Preserve Configuration (default is Preserve)                      *
*      n:No Preserve, reset to factory default settings                  *
*      y:Preserve, keep all of the settings                              *
*   -c IPMI configuration backup/restore                                 *
*      -f [restore.bin] Restore configurations                           *
*      -d [backup.bin] Backup configurations                             *
**************************************************************************
* EXAMPLE                                                                *
*   we like to upgrade firmware through KCS channel                      *
*   lUpdate -f fwuperade.bin -i kcs -r y                                 *
*   lUpdate -d fwdump.bin -i kcs -r y                                    *
*                                                                        *
*   we like to restore/backup IPMI config through KCS channel            *
*   lUpdate -c -f restore.bin -i kcs -r y                                *
*   lUpdate -c -d backup.bin -i kcs -r y                                 *
*                                                                        *
*   we like to upgrade firmware through LAN channel with                 *
*   - BMC IP address 10.11.12.13 port 623                                *
*   - IPMI username is usr                                               *
*   - Password for alice is pwd                                          *
*   - Preserve Configuration                                             *
*   lUpdate -f fw.bin -i lan -h 10.11.12.13 623 -u usr -p pwd -r y       *
*   lUpdate -d fwdump.bin -i lan -h 10.11.12.13 623 -u usr -p pwd -r y   *
*                                                                        *
*   we like to restore/backup IPMI config through LAN channel with       *
*   - BMC IP address 10.11.12.13 port 623                                *
*   - IPMI username is usr                                               *
*   - Password for alice is pwd                                          *
*   - Preserve Configuration                                             *
*   lUpdate -c -f fw.bin -i lan -h 10.11.12.13 623 -u usr -p pwd         *
*   lUpdate -c -d fwdump.bin -i lan -h 10.11.12.13 623 -u usr -p pwd     *
**************************************************************************
```

3.3 SMCIPMITool syntax

```
[fred@lost SMCIPMITool]$ ./SMCIPMITool

SMC IPMI Tool V2.6.5(Build 131220) - Super Micro Computer, Inc.
Usage:
      SMCIPMITool <IP> <username> <password> [commands ... ]
```

*Note : BMC IP address is required when working with **SMCIPMITool**. The command cannot be used with network name.*

SMCIPMITool is a BMC/IPMI dedicated utility, with a subset of subcommanded dedicated to BIOS management

BIOS dedicated commands subset :

```
SMCIPMITool 192.168.51.190 ADMIN ADMIN bios
Command:bios
Command(s):
ver                              Check BIOS info
image <filename>                 Check BIOS image file
update <filename> [options]      Update BIOS
setKey <ProductKey>              Activate Product Key for BIOS update
getMACs <start> <end> <netMask> <file>   Collect MAC addresses into file
setKeys <file>                   Activate product keys for BIOS update
```

3.4 sum syntax

```
[fred@lost sum]$ ./sum
Supermicro Update Manager (for UEFI BIOS) V1.2.0 (2013/12/3) Copyright (C) 2013
Super Micro Computer, Inc. All rights reserved

NAME
  sum (Supermicro Update Manager)

SYNOPSIS
  sum [OPTION]

OPTIONS
  -h  Shows this help information.
  -v  Displays the verbose output.
  -i  <BMC IP address or BMC host name>
  -l  <BMC system list file name>
  -u  <BMC user ID>
  -p  <BMC user password>
  -c  <command name> (case insensitive)

USAGE MODES
  OOB (Out-Of-Band)
            To update and configure a single system remotely through its on-board BMC, use the -i, -u and -p
options.

  In-Band
            To update and configure a single system through its local OS, ignore the -i, -u and -p options.

  Concurrent OOB
            To concurrently update and configure multiple systems through the OOB channel, ignore the -i
option. Use the -l
            option.

            System list file formats
              General OOB commands:
                Format 1: BMC_IP_or_HostName
                Format 2: BMC_IP_or_HostName Username Password
              ActivateProductKey command:
                Format 1: BMC_IP_or_HostName Product_Key
                Format 2: BMC_IP_or_HostName Username Password Product_Key
```

```
COMMANDS
  CheckOOBSupport (OOB Only)
  ActivateProductKey (OOB Only)
    --key   <product key value>
            Uses the product key to activate the managed system.
  UpdateBios
    --file  <file name>
            Updates with the given BIOS image file.
    --reboot  (Optional)
            Forces the managed system to reboot.
    --flash_smbios  (Optional)
            Overwrites SMBIOS data.
            This option is used only for specific purposes. Unless you are familiar with SMBIOS data, do not
use this option.
    --preserve_mer  (Optional)
            Preserves ME firmware region. This option is used only for specific purposes. Unless you are
familiar with ME
            firmware, do not use this option.
    --preserve nv  (Optional)
            Preserves NVRAM. This option is used only for specific purposes. Unless you are familiar with
BIOS NVRAM, do not
            use this option.
  GetBiosInfo
    --file  <file name> (Optional)
            Shows the information of the given BIOS image file.
  GetDefaultBiosCfgTextFile
    --file  <file name>
            Saves the default BIOS configurations to a file.
    --overwrite  (Optional)
            Overwrites the output file.
  GetCurrentBiosCfgTextFile
    --file  <file name>
            Saves the current BIOS configurations to a file.
    --overwrite  (Optional)
            Overwrites the output file.
  ChangeBiosCfg
    --file  <file name>
            Updates with the given BIOS configuration file.
    --reboot  (Optional)
            Forces the managed system to reboot.
  LoadDefaultBiosCfg
    --reboot  (Optional)
            Forces the managed system to reboot.
  GetDmiInfo
    --file  <file name>
            Saves the current DMI information to a file.
    --overwrite  (Optional)
            Overwrites the output file.
  ChangeDmiInfo
    --file  <file name>
            Updates from the given DMI information text file.
    --reboot  (Optional)
            Forces the managed system to reboot.
```

```
EXAMPLES
  #./sum -i 192.168.34.56 -u ADMIN -p ADMIN -c CheckOOBSupport
  #./sum -i 192.168.34.56 -u ADMIN -p ADMIN -c ActivateProductKey --key 1111-1111-1111-1111-1111-1111
  #./sum -i 192.168.34.56 -u ADMIN -p ADMIN -c UpdateBios --file BIOS.rom --reboot
  #./sum -i 192.168.34.56 -u ADMIN -p ADMIN -c GetBiosInfo --file BIOS.rom
  #./sum -i 192.168.34.56 -u ADMIN -p ADMIN -c GetDefaultBiosCfgTextFile --file default.txt --overwrite
  #./sum -i 192.168.34.56 -u ADMIN -p ADMIN -c GetCurrentBiosCfgTextFile --file current.txt --overwrite
  #./sum -i 192.168.34.56 -u ADMIN -p ADMIN -c ChangeBiosCfg --file current.txt --reboot
  #./sum -i 192.168.34.56 -u ADMIN -p ADMIN -c LoadDefaultBiosCfg --reboot
  #./sum -i 192.168.34.56 -u ADMIN -p ADMIN -c GetDmiInfo --file dmi.txt --overwrite
  #./sum -i 192.168.34.56 -u ADMIN -p ADMIN -c ChangeDmiInfo --file dmi.txt --reboot
```

3.5 Rules common to all the OOB tools

- you don’t have to be *root* for using the OOB tools, but you have to give a BMC username/password with administrator privileges. In this document, we are assuming that the BMC account ADMIN (password ADMIN) has administrator privileges.

- if the PATH variable is not updated with path to OOB tools, you will have to use the full path to the commands, or you will have to go in the directory where the tool is located.

4 BMC firmware update

OOB feature is available since BMC IPMI firmware 1.87 and later. For some BMC, the firmware has to be updated prior to be able to use the OOB feature.

Linux TOOL for BMC update is **lUpdate** .

Syntax for BMC firmware update with **lUpdate**:

```
./lUpdate -f <firmware file> -i lan -h <BMC IP> -u ADMIN -p ADMIN -r n
```

Example for BMC firmware update with **lUpdate**:

This is a loop for upgrading several BMC (sequential upgrade) with "1.87 no logo" firmware.

```
#!/bin/sh
for SIP in $(seq 101 160)
do
  IP=192.168.1.${SIP}
  LOG="bmcup-${IP}.log"
  ./lUpdate -f SMC IPMI fw1.87nologo.bin -i lan -h ${IP} -u ADMIN -p ADMIN -r n 1>${LOG} 2>&1 &
  sleep 2
done
wait
```

5 BMC and OOB feature

5.1 OOB activation key

For each BMC, the OOB feature need to be activated with a product key. If the BMC OOB feature is not activated, OOB operations are not available.

5.2 Check if OOB is enabled

The **sum** or the **SMCIPMITool** commands can be used to check if the OOB feature is activated for each BMC.

5.2.1 Check if OOB is enabled – "sum" command

Syntax for the **sum** command:

```
sum –i <BMC IP or Hostname> -u <ADMIN LOGIN> -u <ADMIN PASSWORD> -c CheckOOBSupport
```

Example when using the **sum** command and the product key is not activated:

```
sum -i hwm118  -u ADMIN -p ADMIN -c CheckOOBSupport
Supermicro Update Manager (for UEFI BIOS) V1.2.0 (2013/12/3) Copyright (C) 2013
Super Micro Computer, Inc. All rights reserved

[KEY]
Product Key Activated............No

[BMC]
BMC FW Version...................03.19
BMC Supports OOB BIOS Config.....Yes
BMC Supports OOB DMI Edit........Yes

[BIOS]
BIOS Board ID....................0711
BIOS Build Date..................2013/7/5
BIOS Supports OOB BIOS Config....Yes
BIOS Supports OOB DMI Edit.......Yes
```

Example when using the **sum** command and the product key is activated:

```
sum -i r424e3a-bmc -u ADMIN -p ADMIN -c CheckOOBSupport
Supermicro Update Manager (for UEFI BIOS) V1.2.0 (2013/12/3) Copyright (C) 2013
Super Micro Computer, Inc. All rights reserved

[KEY]
Product Key Activated............Yes

[BMC]
BMC FW Version...................03.17
BMC Supports OOB BIOS Config.....Yes
BMC Supports OOB DMI Edit........Yes

[BIOS]
BIOS Board ID....................062F
BIOS Build Date..................2013/6/28
BIOS Supports OOB BIOS Config....Yes
BIOS Supports OOB DMI Edit.......No
    [download DMI file]  not supported.
```

5.2.2 Check if OOB is enabled – "SMCIPMITool" command

When using **SMCIPMITool**, the **bios ver** subcommand can be used for checking if OOB is produc key activated. If the product key is not activated, the subcommand is requesting that the product key need to be activated. If the OOB feature is activated, the command is returning the BIOS release.

Example when using the **SMCIPMITool bios ver** subcommand and the product key is not enabled:

```
SMCIPMITool 192.168.51.190 ADMIN ADMIN bios ver
The product key needs to be activated for this device
```

Example when using the **SMCIPMITool bios ver** subcommand and the product key is enabled:

```
SMCIPMITool 192.168.51.190 ADMIN ADMIN bios ver
062f BIOS Date: 6/28/2013
```

5.3 Collect the BMC MAC addresses for the activation key

If OOB is not activated, an activation key has to be requested and configured in the BMC.

For requesting such a key, the MAC address has to be provided.

The **SMCIPMITool bios getMACs** subcommand is in charge of getting the MACs address of the BMC located in a subnet.

Syntax of **SMCIPMITool bios getMACs** for getting the MAC addresses**:**

```
SMCIPMITool.jar <BMC IP> <IPMI USER> <IPMI PASSWD> bios getMACs <BMC START IP> <BMC STOP IP> <BMC SUBNET MASK> <OUTPUT FILE>
```

Example of **SMCIPMITool bios getMACs** subcommand for getting the MAC addresses:

```
SMCIPMITool.jar 10.26.101.51 ADMIN ADMIN bios getMACs 10.26.101.1 10.26.101.54 255.255.255.0  macs.txt
Finding BIOS update available Devices ...
00259074A04E ;   10.26.101.9            IPMI 2.0 SIM(WA) IPMI+KVM
0025907369C2 ;   10.26.101.37           IPMI 2.0 SIM(WA) IPMI+KVM
0025907369BA ;   10.26.101.38           IPMI 2.0 SIM(WA) IPMI+KVM
0025907368C2 ;   10.26.101.39           IPMI 2.0 SIM(WA) IPMI+KVM
0025907369BC ;   10.26.101.40           IPMI 2.0 SIM(WA) IPMI+KVM
002590CC28E4 ;   10.26.101.45           IPMI 2.0 SIM(WA) IPMI+KVM
002590CC2804 ;   10.26.101.46           IPMI 2.0 SIM(WA) IPMI+KVM
002590CC173D ;   10.26.101.47           IPMI 2.0 SIM(WA) IPMI+KVM
002590CC9778 ;   10.26.101.48           IPMI 2.0 SIM(WA) IPMI+KVM
002590CC28E5 ;   10.26.101.49           IPMI 2.0 SIM(WA) IPMI+KVM
0025906D0673 ;   10.26.101.50           IPMI 2.0 SIM(WA) IPMI+KVM
002590CC133B ;   10.26.101.51           IPMI 2.0 SIM(WA) IPMI+KVM
12 device(s) found. macs.txt was created
```

Example of output file ("macs.txt") :

```
00259074A04E;10.26.101.9
0025907369C2;10.26.101.37
0025907369BA;10.26.101.38
0025907368C2;10.26.101.39
0025907369BC;10.26.101.40
002590CC28E4;10.26.101.45
002590CC2804;10.26.101.46
002590CC173D;10.26.101.47
002590CC9778;10.26.101.48
002590CC28E5;10.26.101.49
0025906D0673;10.26.101.50
002590CC133B;10.26.101.51
```

The output file has to be provided to people in charge of OOB Keys requests.

5.4 BMC activation and keys file

A file with the keys (based on "**macs.txt**" file) is provided by the people in charge of OOB Keys.

Example of "**keys.txt**" file, with MAC and Keys

```
002590CE2F9C;192.168.1.107;2498-66A2-25CD-0CBA-5F3C-2766
002590CE2FBA;192.168.1.105;0657-D56C-CB77-E3E9-81A1-68DC
```

The **SMCIPMITool bios setKeys** subcommand is used to apply the keys (from the keys file) on all the BMC on a subnet

Syntax of the **SMCIPMITool bios setKeys** subcommand for setting the keys:

```
SMCIPMITool <BMC IP> <IPMI USER> <IPMI PASSWD> bios setKeys <KEYS FILE>
```

Example of the **SMCIPMITool bios setKeys** subcommand for setting the keys:

```
SMCIPMITool.jar 192.168.1.101 ADMIN ADMIN bios setKeys Keys.txt
```

note: even if the BMC IP addresses are given in the key file, a valid BMC IP has to be provided to the command.

6 BIOS operations

6.1 Check the BIOS image

Before flashing the BIOS, the file with the BIOS image can be verified in the way to check that the right BIOS release will be used for the BIOS upgrade. The verification of the BIOS file can be done with **sum** or with **SMCIPMITool**.

6.1.1 Check the BIOS image with sum

When the BIOS file is verified with **sum**, the command is also returning the information about the remote system to upgrade (= "*managed system*").

The returned information are the "*BIOS build date*" and the "*Board ID*".

WARNING: the managed system and the BIOS file should have the same *Board ID*. If the IDs are not the same, the BIOS image is not dedicated to the motherboard and your system should not restart after the BIOS upgrade.

Syntax of **sum** for verification of the BIOS image

```
sum -i <BMC IP or Hostname> -u <IPMI USER> -p <IPMI PASSWD> -c GetBiosInfo --file <BIOS file>
```

Example of BIOS image verification with **sum**:

```
sum -i 192.168.51.190 -u ADMIN -p ADMIN -c GetBiosInfo --file X9DRTH3.628
Supermicro Update Manager (for UEFI BIOS) V1.2.0 (2013/12/3) Copyright (C) 2013
Super Micro Computer, Inc. All rights reserved
Managed system...........192.168.51.190
    Board ID.............062F
    BIOS build date......2013/6/28
Local BIOS image file....X9DRTH3.628
    Board ID.............062F
    BIOS build date......2013/6/28
```

6.1.2 Check the BIOS image with SMCIPMITool

When the BIOS image is verified with **SMCIPMITool**, the command is returning the BIOS release date, the motherboard type (please check if the motherboard of the target system is the same model) and the BIOS size.

Syntaxe of **SMCIPMITool** for verification of the BIOS image:

```
SMCIPMITool.jar <BMC IP> <IPMI USER> <IPMI PASSWD> bios image <BIOS file>
```

Example of BIOS image verification with **SMCIPMITool**:

```
[fred@lost SMCIPMITool]$ ./SMCIPMITool 192.168.51.190 ADMIN ADMIN bios image X9DRTH3.628
Date     = 06/28/2013
MB Type  = X9DRT-HF
Size     = 16 MB
```

6.2 Flash the BIOS

BIOS can be upgraded by using **sum** or **SMCIPMITool** commands.

sum is the preferred command for BIOS upgrade.

6.2.1 Flash the BIOS with "sum" command

When flashing the BIOS with **sum** command, the syntax is :

```
sum -i <BMC IP or Hostname> -u <IPMI USER> -p <IPMI PASSWD> -c UpdateBios --file <BIOS file> [optional flags]
```

note: After BIOS upgrade, the upgraded system has to be restarted. The «--reboot» flag can be used if the upgraded system can be restarted without any restriction. But in most of the case, when an operating system is running on the upgraded system, the "—reboot" flag should not be used, and the system should be restarted from the operating system, in the way to stop all the running jobs in the right way and to flush all the I/Os that may be in progress.

For usual BIOS update, the optional flags should not be used. These flags are described in **sum** online help (see section 3.4).

Example of BIOS update when using the **sum** command:

```
sum -i 192.168.51.152 -u ADMIN -p ADMIN -c UpdateBios  --file /release/Supermicro/X9/X9DRH/X9DRH3.C27
Supermicro Update Manager (for UEFI BIOS) V1.2.0 (2013/12/3) Copyright (C) 2013
Super Micro Computer, Inc. All rights reserved
Status: Start updating BIOS for 192.168.51.152

***********************************WARNING***************************************
    Do not remove AC power from the server.
*****************************************************************************

.................................................
.
.................................................
.................................................
.................................................
.................................................
.................................................
.................................................
.............
Status: BIOS is updated for 192.168.51.152

Note: You have to reboot or power up the system for the changes to take effect
```

Notes: the updated machine was powered off, and the CPU were not installed.

6.2.2 Flash the BIOS with "SMCIPMITool" command

When flashing the BIOS with the SMCIPMIToolFlash BIOS with **SMCIPMITool** command, the syntax is:

```
SMCIPMITool.jar <BMC IP> <IPMI USER> <IPMI PASSWD> bios update <BIOS file> [optional flags]
```

sum is the preferred command for the BIOS upgrade, and **SMCIPMITool** should not be used for such usage.

Example of BIOS update when using the **SMCIPMITool** command:

```
java -jar SMCIPMITool.jar 192.168.51.190 ADMIN ADMIN bios update X9DRT201210232002.ROM -N -R -MER
Option: NVRAM overwrite
Option: ME region
Option: SMBIOS preserve
===============
BIOS Image info
===============
Date     = 10/23/2012
MB Type  = X9DRT-HF
Size     = 16 MB

=============
BIOS ROM info
=============
062f BIOS Date: 10/23/2012

====================
Uploading BIOS image
====================
Progress:|>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>| 100%
Upload Time: 2 min 27 sec(s)

=============
Updating BIOS
=============
Progress:|>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>| 100%
Update Time: 4 min 34 sec(s)
Done
Total Elapse Time: 7 min 8 sec(s)
```

7 Parallel updates with OOB utilities

A java virtual machine is required when using **SMCIPMITool** and **sum** commands (sum is internally using SMCIPMITool). Due to the heavy system load for each java virtual machine, the management node may be overloaded when multiple instances of these command are executed.

In the way to avoid such overload for parallel update, the **xargs** utility may be used in the way to restrict the number of simultaneous upgrades. Use the -P flag to restrict the number of simultaneous commands. A good start point is to get the value of th -P flag lower than or equal to the number of processors of the management node. This number of simultaneous command may also be reduced if the management node memory is restricted.

7.1 Parallel BMC reset

Before upgrading the BMC IPMI firmware, BMC cold reset is recommended, in the way to have the BMC in a good state before of the update.

Example for BMC cold reset (before BMC IPMI firmware upgrade) when using the usual ipmitool command:

```
NODES="r424e3a-bmc r424e3b-bmc r424e3c-bmc r424e3d-bmc r424f3a-bmc r424f3b-bmc r424f3c-bmc r424f3d-bmc"
echo $NODES | sed 's/ /\n/g' | xargs  -t  -I {} -P 4 ipmitool -I lanplus -H {} -U ADMIN -P ADMIN bmc reset
cold
ipmitool -I lanplus -H r424e3a-bmc -U ADMIN -P ADMIN bmc reset cold
ipmitool -I lanplus -H r424e3b-bmc -U ADMIN -P ADMIN bmc reset cold
ipmitool -I lanplus -H r424e3c-bmc -U ADMIN -P ADMIN bmc reset cold
ipmitool -I lanplus -H r424e3d-bmc -U ADMIN -P ADMIN bmc reset cold
Sent cold reset command to MC
ipmitool -I lanplus -H r424f3a-bmc -U ADMIN -P ADMIN bmc reset cold
Sent cold reset command to MC
ipmitool -I lanplus -H r424f3b-bmc -U ADMIN -P ADMIN bmc reset cold
Sent cold reset command to MC
Sent cold reset command to MC
ipmitool -I lanplus -H r424f3c-bmc -U ADMIN -P ADMIN bmc reset cold
ipmitool -I lanplus -H r424f3d-bmc -U ADMIN -P ADMIN bmc reset cold
Sent cold reset command to MC
Sent cold reset command to MC
Sent cold reset command to MC
Sent cold reset command to MC
```

*Note: in this example, parameters (BMC hostnames) are separated by blank. The blank is replaced by a New Line (\n) through the **sed** command. Max of 4 jobs running at the same time… (-P 4)*

7.2 Parallel BMC IPMI firmware update

Example for BMC firmware update:

- Create a script in charge of updating one BMC – arguments are IPMI firmware filename and BMC hostname. Here, the script is named xargslup.sh and it is taking the given parameter – firmware filename and BMC hostname - in the way to compute the BMC IP and to update this BMC with the given IPMI filen. A timestamp and the BMC hostname are printed before and after the firmware update:

```
#!/bin/sh

FIRM=$1
BMC=$2
IP=$(ping -c1 $BMC | awk '/PING/{print $3}' | sed 's/(\|)//g')
LOG="${BMC}.log"
LUPDCMD=/home/fred/Downloads/Linux1.21/lUpdate

DATE=$(date)
echo "$BMC - start => $DATE"
$LUPDCMD -f ${FIRM} -i lan -h ${IP} -u ADMIN -p ADMIN -r n 1>$LOG 2>&1
DATE=$(date)
echo "$BMC - stop => $DATE"
#!/bin/sh
```

- Start the update by giving all the BMC hostname (the script will convert hostname to IP). **xargs** will split each argument for a dedicated command (flag *-n 1*) and no more than 3 jobs (flag *-P 3*)will be running at the same time:

```
NODES="r424e3a-bmc r424e3b-bmc r424e3c-bmc r424e3d-bmc r424f3a-bmc r424f3b-bmc r424f3c-bmc r424f3d-bmc"
echo $NODES | xargs -t -n 1 -P 3 ./xargslup.sh OEM_V226_NOLOGO.BIN
```

With the command output (on the console/terminal), we can check that no more than 3 jobs are running at the same time:

```
./xargslup.sh OEM_V226_NOLOGO.BIN r424e3a-bmc
./xargslup.sh OEM_V226_NOLOGO.BIN r424e3b-bmc
./xargslup.sh OEM_V226_NOLOGO.BIN r424e3c-bmc
r424e3a-bmc - start => Mon Sep  9 11:51:16 UTC 2013
r424e3b-bmc - start => Mon Sep  9 11:51:16 UTC 2013
r424e3c-bmc - start => Mon Sep  9 11:51:16 UTC 2013
r424e3b-bmc - stop => Mon Sep  9 11:57:20 UTC 2013
./xargslup.sh OEM_V226_NOLOGO.BIN r424e3d-bmc
r424e3d-bmc - start => Mon Sep  9 11:57:20 UTC 2013
r424e3a-bmc - stop => Mon Sep  9 11:57:26 UTC 2013
./xargslup.sh OEM V226 NOLOGO.BIN r424f3a-bmc
r424f3a-bmc - start => Mon Sep  9 11:57:26 UTC 2013
r424e3c-bmc - stop => Mon Sep  9 11:57:29 UTC 2013
./xargslup.sh OEM_V226_NOLOGO.BIN r424f3b-bmc
r424f3b-bmc - start => Mon Sep  9 11:57:29 UTC 2013
r424e3d-bmc - stop => Mon Sep  9 12:03:25 UTC 2013
./xargslup.sh OEM V226 NOLOGO.BIN r424f3c-bmc
r424f3c-bmc - start => Mon Sep  9 12:03:25 UTC 2013
r424f3b-bmc - stop => Mon Sep  9 12:03:27 UTC 2013
./xargslup.sh OEM_V226_NOLOGO.BIN r424f3d-bmc
r424f3d-bmc - start => Mon Sep  9 12:03:27 UTC 2013
r424f3a-bmc - stop => Mon Sep  9 12:03:29 UTC 2013
r424f3c-bmc - stop => Mon Sep  9 12:09:09 UTC 2013
r424f3d-bmc - stop => Mon Sep  9 12:09:20 UTC 2013
```

note: the **lUpdate** command is a light command, with few impacts on the management node load. But it may be interesting to restrict the number of lUpdate running instances for following purpose:

- Light network architecture, with hub (may lead to update issue if too many update are simultaneously done)
- Restricted number of available TCP-IP ports (only some tenth of thousand ports are available for all the network tasks, which may be short for some clusters)

7.3 Parallel BIOS update

Example for BIOS update:

- Create a script in charge of updating one BIOS (BMC hostname given in arg to the script). Here, the script is named xargsbup.sh and it is taking the given parameters – the BIOS filename and the BMC hostname - in the way to update the BIOS through BMC. A timestamp and the BMC hostname are printed before and after the BIOS update:

```
#!/bin/sh

BIOS=$1
BMC=$2
LOG="${BMC}-BIOS.log"
SUMCMD=/home/fred/sum_1.0.0_Linux_x64/sum

DATE=$(date)
echo "$BMC - start => $DATE"
$SUMCMD -i ${BMC} -u ADMIN -p ADMIN -c UpdateBios --file ${BIOS}  1>$LOG 2>&1
DATE=$(date)
echo "$BMC - stop => $DATE"
```

- Start the update by giving all the BMC hostname. **xargs** will split each argument for a dedicated command (flag *-n 1*) and no more than 2 jobs (flag *-P 2*)will be running at the same time:

```
NODES="r424e3a-bmc r424e3b-bmc r424e3c-bmc r424e3d-bmc"
Echo $NODES | xargs -t -n 1 -P 2 ./xargsbup.sh R24E3X51T.rom
```

With the command output (on the console/terminal), we can check that no more than 2 jobs are running at the same time:

```
./xargsbup.sh R24E3X51T.rom r424e3a-bmc
./xargsbup.sh R24E3X51T.rom r424e3b-bmc
r424e3a-bmc - start => Mon Sep  9 12:59:27 UTC 2013
r424e3b-bmc - start => Mon Sep  9 12:59:27 UTC 2013
r424e3a-bmc - stop => Mon Sep  9 13:06:41 UTC 2013
./xargsbup.sh R24E3X51T.rom r424e3c-bmc
r424e3c-bmc - start => Mon Sep  9 13:06:41 UTC 2013
r424e3b-bmc - stop => Mon Sep  9 13:07:10 UTC 2013
./xargsbup.sh R24E3X51T.rom r424e3d-bmc
r424e3d-bmc - start => Mon Sep  9 13:07:10 UTC 2013
r424e3c-bmc - stop => Mon Sep  9 13:14:07 UTC 2013
r424e3d-bmc - stop => Mon Sep  9 13:14:12 UTC 2013
```

note: in this example, r424e3a and r424e3b are up and running, r424e3c is down, and r424e3d has no processor and is down too. We can see that BIOS is updated on all the nodes, even is processors are not installed. In all cases, BIOS update take ~7 minutes.