COMPUTING

Blade Services Software on ATCA-9405

Programmer's Reference

P/N: 6806800R18H

July 2017

ARTESYN™

EMBEDDED TECHNOLOGIES

Trademarks

Notice

Limited and Restricted Rights Legend

Contact Address

Artesyn Embedded Technologies

Marketing Communications

2900 S. Diablo Way, Suite 190

Tempe, Arizona 85282

Contents

List of Tables

List of Figures

Overview of Contents

This manual is divided into the following chapters and appendices.

Abbreviations

This document uses the following abbreviations:

Abbreviation	Definition
AMC	Advanced Mezzanine Card
API	Application Programming Interface
AdvancedTCA	Advanced Telecommunications Computing Architecture
ATCA	Advanced Telecommunications Computing Architecture
ATIS	AdvancedTCA Interfaces Service
BBS	Basic Blade Services
BC	Base Channel
BIB	Board Information Block
BIOS	Basic Input Output System
BT	Block Transfer
CGL	Carrier Grade Linux

Abbreviation	Definition
CMC	Common Mezzanine Card
CMD	Command Line Tool
CPU	Central Processing Unit
DAT	Domain Alarm Table
DHCP	Dynamic Host Configuration Protocol
DoS	Denial of Service
ECC	Embedded Communications Computing
ER	Expected Responder
EST	Eastern Standard Time
ETH	Ethernet
EVQ	Event Queue
FC	Fabric Channel
FCU	FUF Command Line Utility
FM	Fault Management
FPGA	Field Programmable Gate Array
FRI	Firmware Recovery Image
FRU	Field Replaceable Unit
FUF	Firmware Upgrade Facility
FWH	Firmware Hub
GPIO	General Purpose Input/Output
HPI	Hardware Platform Interface
HPI-B	Hardware Platform Interface Version B
HPM	Hardware Platform Management
I/O	Input Output
IDE	Integrated Device Electronics
IP	Internet Protocol
IPM	Intelligent Platform Management
IPMB	Intelligent Platform Management Bus

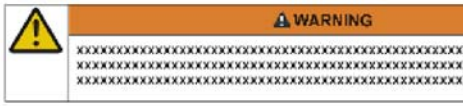
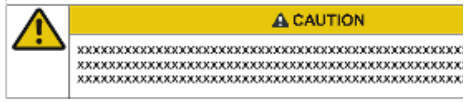
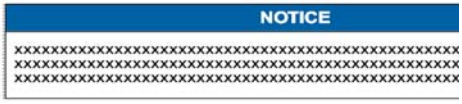
Abbreviation	Definition
IPMC	Intelligent Platform Management Controller
IPMI	Intelligent Platform Management Interface
KCS	Keyboard Control Style
LED	Light Emitting Diode
LHC	Link Health Check
LSP	Linux Support Package
LT	Lower Threshold
LUN	Logic Unit Number
MAC	Media Access Control
MIB	Management Information Base
MMC	Module Management Controller
NTP	Network Time Protocol
OEM	Original Equipment Manufacturer
OSDL	Open Source Development Labs
PC	Personal Computer
PCI	Peripheral Component Interconnect
PCIEX	PCI Express
PEM	Power Entry Module
PICMG	PCI Industrial Computers Manufacturers Group
PMC	PCI Mezzanine Card
PNE	Platform for Network Equipment
PrPMC	Processor PMC
PXE	Preboot Execution Environment
RAM	Random Access Memory
RDR	Resource Data Record
RMCP	Remote Monitoring and Control Protocol
ROM	Read Only Memory
RPM	RedHat Package Manager

Abbreviation	Definition
RTM	Rear Transition Module
SAF	Service Availability Forum
SAM	Shelf Manager and Alarm Module
SAS	Serial Attached SCSI
SATA	Serial ATA
SCSI	Small Computer System Interface
SCXB	System Controller Switching Blade
SDK	Software Development Kit
SDR	Sensor Data Record
SIP	Serial Interface Protocol
SIT	Simple Internet Transition
SMI	Serial Management Interface
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSU	Synchronization Supply Unit
TAR	Tape Archiver
TBD	To Be Defined
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TTY	Teletypewriter
UC	Update Channel
UDP	User Datagram Protocol
USB	Universal Serial Bus
UT	Upper Threshold
VLAN	Virtual Local Area Network
WDT	Watchdog Timer

Conventions

The following table describes the conventions used throughout this manual.

Notation	Description
0x00000000	Typical notation for hexadecimal numbers (digits are 0 through F), for example used for addresses and offsets
0b0000	Same for binary numbers (digits are 0 and 1)
bold	Used to emphasize a word
`Screen`	Used for on-screen output and code related elements or commands in body text
`Courier + Bold`	Used to characterize user input and to separate it from system output
Reference	Used for references and for table and figure descriptions
File > Exit	Notation for selecting a submenu
<text>	Notation for variables and keys
[text]	Notation for software buttons to click on the screen and parameter description
...	Repeated item for example node 1, node 2, ..., node 12
. . .	Omission of information from example/command that is not necessary at the time being
..	Ranges, for example: 0..4 means one of the integers 0,1,2,3, and 4 (used in registers)
|	Logical OR

Notation	Description
	Indicates a hazardous situation which, if not avoided, could result in death or serious injury
	Indicates a hazardous situation which, if not avoided, may result in minor or moderate injury
	Indicates a property damage message
	No danger encountered. Pay attention to important information

Summary of Changes

See the table below for manual revisions and changes.

Part Number	Date	Description
6806800R18H	July 2017	Added registered trademark to SRstackware.
6806800R18G	October 2016	Updated copyrights information.
6806800R18F	June 2014	Re-branded to Artesyn template. Added *Redundant Flash Installation* on page 32, *Service Processor Persistent Storage* on page 42, *frumrecset* on page 85, Chapter 5, *Network Management Utils, on page 115*, and Chapter 8, *ViewCheck, on page 151*. Updated *Software Building Blocks* on page 21, Table 2-1 on page 25, *Service Processor Linux Installation* on page 27, *Running Linux on OCTEONS* on page 47, Table 3-1 on page 53, and Chapter 4, *Hardware Platform Management, on page 65*,
6806800R18E	November 2013	Updated Table 7-2 on page 147.
6806800R18D	October 2013	Updated Table 7-2 on page 147.
6806800R18C	April 2013	Updated *Default Configuration* on page 130 and *2x40G/8x10G RTM Ports* on page 147.
6806800R18B	February 2013	Added *Debugging an SE-S application or the Linux kernel* on page 49. Updated *FUF Components* on page 53.
6806800R18A	February 2013	Initial version

1.1 Overview

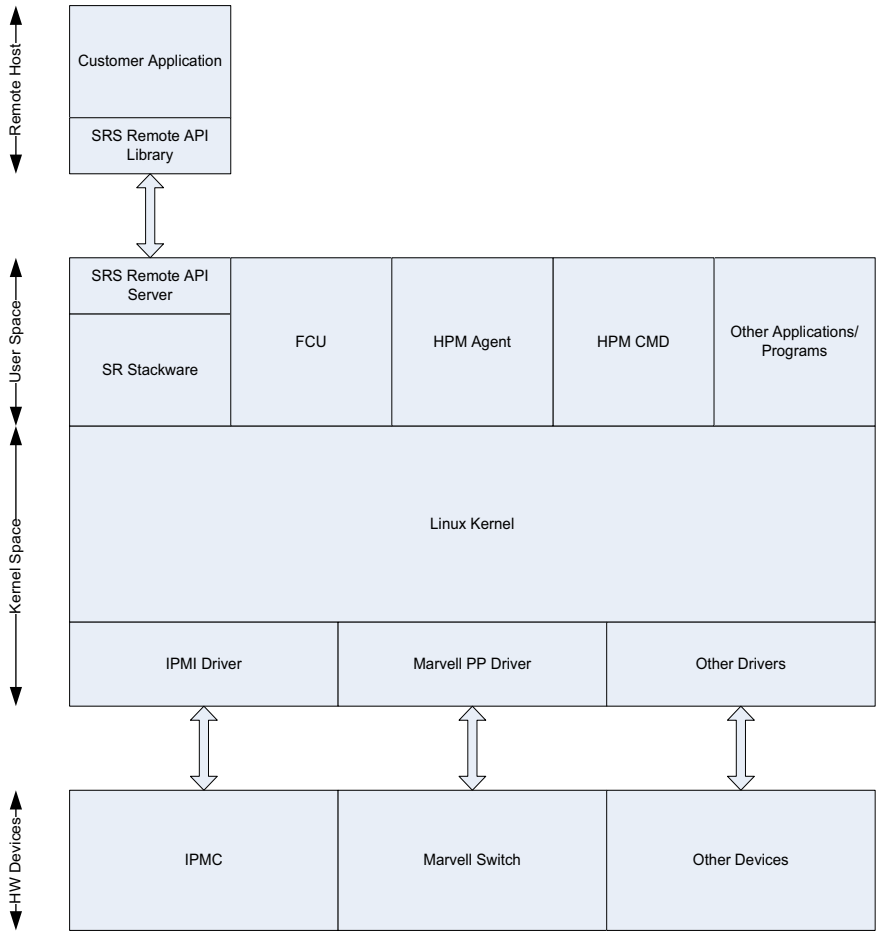
The blade services software for the ATCA-9405 includes a common set of functionality which is available for all AdvancedTCA blades and AMC modules, and a unique set of functionality which is tailored specifically to the ATCA-9405. It is designed to provide a unified software operating environment to an application and provides a standard management interface to the rest of the AdvancedTCA system environment.

1.2 Software Building Blocks

The blade services include a common set of functionality that is available for all ATCA blades and AMC modules. The blade services also include a unique set of functionality that is tailored to a particular blade or module.

Figure 1-1 illustrates the blade services software architecture.

Figure 1-1 BBS Architcture



Blade Services consists of the following software and services:

- Firmware Upgrade Facility
  The Firmware Upgrade Facility (FUF) provides a uniform way to upgrade firmware on Artesyn blades and AMC modules, regardless on which location the firmware is stored. FUF upgrades several firmware components such as FPGA, IPMC or u-boot. The FUF currently consists of a Firmware Upgrade Command Line Utility (FCU), flash device drivers, and specially prepared firmware recovery image files.
- Linux
  Blade services is based on Wind River Linux edition 4.1.
- Hardware Platform Management
  The Hardware Platform Management (HPM) in ATCA systems is based on Intelligent Platform Management Interface (IPMI) specification. IPMI commands can be complex and cumbersome. Using a certain set of commands, HPM facilitates the blade or module-level hardware management.
- SRstackware®
  SRstackware is a switching and routing stackware which supports basic switch configuration along with Layer 2 and Layer 3 protocols support.
- SNMP Agent
  As each blade services blade/module is individually managed, the default installation script installs and initializes the Net-SNMP agent.
- Network Management Utility (NMU)
  The NMU tool is used to query SFP/SFP+ and QSFP information that are connected to ARTM9405.

2.1 Overview

This section describes how to install the blade services software for the ATCA-9405. Artesyn provides software images, including software updates, to its licensed customers. To obtain the latest blade services release, please contact your local sales representative.

2.1.1 Package Information

Most of the blade services software is packaged with the Red Hat Package Manager (RPM) and is installed as a part of the standard installation. In general, you will not need to install or upgrade an individual package. All blade service software with the exception of SR Stackware are installed at the `/opt/bladeservices` location.

The following table lists the blade services software RPMs that are delivered:

Table 2-1 RPM Packages

RPM Package Name	File Name	Description
bbs-fcu	`bbs-fcu-atca9405-<version>-pne41.rpm`	Firmware Upgrade tool
bbs-hpmagentcmd	`bbs-hpmagentcmd-atca9405-<version>-pne41.rpm`	Hardware Platform Management tool
mvppdrv	`mvppdrv-<version>.ppc_e500v2-wrspne4-linux.rpm`	Marvell Packet Processor driver
mvpplib	`mvpplib-<version>.ppc_e500v2-wrspne4-linux.rpm`	Marvell Packet Processor shared library
mvppsw	`mvppsw-<version>.ppc_e500v2-wrspne4-linux.rpm`	Marvell Packet Processor application
srstackware-9405	`srstackware-9405-<version>.powerpc.rpm`	SR Stackware
srstackware-config-9405	`srstackware-config-9405-<version>.powerpc.rpm`	SR Stackware Configuration
octremoteutils	`octremoteutils-<version>.ppc_e500v2-wrspne4-linux.rpm`	Octeon remote tools
octpstored	`octpstored-<version>.ppc_e500v2-wrspne4-linux.rpm`	Octeon persistent storage daemon

Table 2-1 RPM Packages

RPM Package Name	File Name	Description
nmucmd	`bbs-nmucmd-atca9405-<version>-pne41.rpm`	NMU command tool
bbs-supervisor	`bbs-supervisor-atca9405-<version>-pne41.rpm`	Supervisor application
syslcu.atca9405	`syslcu.atca9405-<version>.noarch.rpm`	Log collection utility
diagnostics	`diagnostics-<version>.windriver4.1.linux.atca9405sp.rpm`	ViewCheck Diagnostics Core
orion-license.atca9405.windriver-pne4.1	`orion-license.atca9405.windriver-pne4.1-<version>.ppc.rpm`	Orion License for ViewCheck
ssf_main_rel.atca9405 ssf_csim_rel.atca9405 ssf_diagnosticsTLS_rel.atca9405	`ssf_<svc>.atca9405-pne4.1-<version>.ppc.rpm`	BSF services for ViewCheck
atca9405_cn68xx-vmlinux_initramfs	`atca9405_cn68xx-vmlinux_initramfs-<version>.ppc_e500v2-wrspne4-linux.rpm`	Octeon SDK Linux kernel with embedded initramfs

Some other Blade Service Software tools and files are not packaged, but inherent part of the root filesystem.

2.1.2 Installation Overview

The installation process boots a custom ramdisk kernel from the network. Current blade services release does not support installation from CD.

The blade must have access to a TFTP server to retrieve the files.

2.2 Service Processor Linux Installation

This chapter describes the installation of WindRiver Linux 4.x including Blade Services Software on the onboard flash disk.

Generally, ATCA-9405 blade is delivered with a pre-installed Linux flash installation. In this case, you can skip the flash partitioning and directly install the new Linux release.

NOTICE

Installing a new Linux release deletes the existing installation completely. It is recommended to make a back up of existing installation.

The following table lists the installation files provided with the release:

Table 2-2 Linux Installation Files

Installation Target	Installation filename	Target filename
Kernel	`atca9405_p2020-default_kernel_image-WR4.1.0.0_standard_<version>`	uImage
Device-tree-blob	`atca9405_p2020-p2020rdb.dtb-WR4.1.0.0_standard_<version>`	atca940x-sp.dtb
Root filesystem	`atca9405_p2020-standard-glibc_std-dist-<version>.tar.bz2`	extracted root filesystem

The flashinstall script installs the files on the flash disk. It is located in the '`/opt/bladeservices/bin` directory. The script can download the installation files from a TFTP server hosting these files or use local files. In the latter case, the user needs to manually copy the files before the installation.

NOTICE

After the installation, the content of the root filesystem is same as the ramdisk.

The flashinstall script has the following usage:

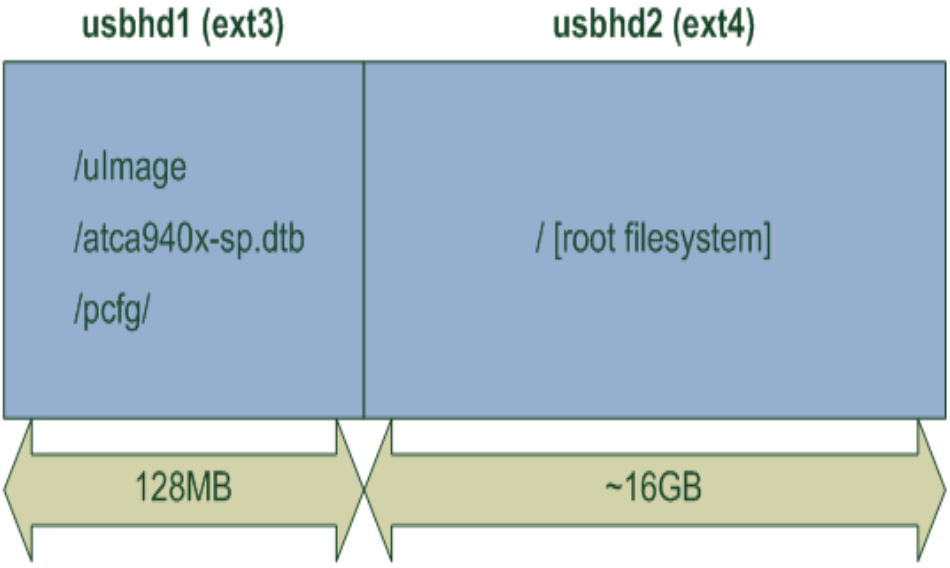
```
Usage: flashinstall [OPTION]... TARGET
Installs the ATCA-9405 Linux on the on-board flash disk.

TARGET must be one of the following:
  kernel
  dtb
  rootfs

Options:
  -i IPADDR    IP address of TFTP server
  -t FILE      location of the installation file on the TFTP server
               or local location if '-i' option is not used
               The default installation filenames are:
                  kernel: atca9405_p2020-default_kernel_image
WR4.1.0.0_standard
                  dtb:    atca9405_p2020-p2020rdb.dtb
WR4.1.0.0_standard
                  rootfs: atca9405_p2020-standard-glibc_std
dist.tar.bz2
  -b BANK_NUM  use bank BANK_NUM for installation
               Supported bank numbers are: '1 | 2' [default=1]
               A bank number >1 requires redundant partitions
  -l LOGFILE   output install steps into logfile
  -d DEVICE    use device DEVICE [default=/dev/usbhd]
  -V           print version information and exit
  -h           display this help and exit
```

2.2.1 FLash Partitioning

Onboard flash disk uses two primary partitions. The first partition is for the kernel image and device-tree-blob (DTB) files. It uses the `ext3` filesystem type and it can be handled by u-boot. The second partition is used for the root filesystem and uses the ext4 filesystem type.

The following figure shows the partition layout:

Figure 2-1 Flash Partition



The following command restores the partition layout of the onboard flash disk. Please note that all data on the disk will be lost. As already mentioned in the introduction this step is optional if Linux is already installed on the flash disk.

```
root@ATCA-9405-12-13:/root> flashprep
WARNING: All data on device '/dev/usbhd' will be lost!
Continue (y/n) y
*) removing any existing partitions on device '/dev/usbhd' [OK]
*) creating new primary partitions '/dev/usbhd1', '/dev/usbhd2'
[OK]
*) creating an ext3 fs on '/dev/usbhd1' [OK]
*) tuning ext3 fs on '/dev/usbhd1' [OK]
*) creating an ext4 fs on '/dev/usbhd2' [OK]
*) tuning ext4 fs on '/dev/usbhd2' [OK]
```

2.2.2 Kernel Upgrade

You can upgrade kernel with the flashinstall script using the kernel target option. A kernel upgrade can be performed even if the system is currently running from flash disk.

The following example downloads the kernel image file located in the `/tftpboot/9405` directory on the TFTP server with the IP address 172.16.166.55 and installs it on the flash disk.

```
root@ATCA-9405-12-13:/root> flashinstall -i 172.16.166.55 -t
9405/atca9405_p2020-default_kernel_image-WR4.1.0.0_standard -l
kernel-inst.log kernel
Installation started - this may take a while.
Please do not abort this script!

*) mounting '/dev/usbhd1' on '/mnt.flashinstall.usbhd1' [OK]
*) downloading '9405/atca9405_p2020-default_kernel_image-
WR4.1.0.0_standard' from tftp server '172.16.166.55' to
'/mnt.flashinstall.tmpfs/atca9405_p2020-default_kernel_image-
WR4.1.0.0_standard'
*) installing '/mnt.flashinstall.tmpfs/atca9405_p2020-
default_kernel_image-WR4.1.0.0_standard' on
'/mnt.flashinstall.usbhd1/uImage' [OK]
*) unmounting '/mnt.flashinstall.usbhd1' [OK]

Installation completed successfully.
```

2.2.3 Device-tree-blob Upgrade

You can upgrade Device-tree-blob (DTB) with the flashinstall script using the DTB target option. A DTB upgrade can be performed even if the system is currently running from flash disk.

The following example downloads the DTB file located in the `/tftpboot/9405` directory on the TFTP server with the IP address 172.16.166.55 and installs it on the flash disk.

```
root@ATCA-9405-12-13:/root> flashinstall -i 172.16.166.55 -t
9405/atca9405_p2020-p2020rdb.dtb-WR4.1.0.0_standard -l dtb-
inst.log dtb
Installation started - this may take a while.
```

```
Please do not abort this script!

*) mounting '/dev/usbhd1' on '/mnt.flashinstall.usbhd1' [OK]
*) downloading '9405/atca9405_p2020-p2020rdb.dtb-
WR4.1.0.0_standard' from tftp server '172.16.166.55' to
'/mnt.flashinstall.tmpfs/atca9405_p2020-p2020rdb.dtb-
WR4.1.0.0_standard' [OK]
*) installing '/mnt.flashinstall.tmpfs/atca9405_p2020-
p2020rdb.dtb-WR4.1.0.0_standard' on
'/mnt.flashinstall.usbhd1/atca940x-sp.dtb' [OK]
*) unmounting '/mnt.flashinstall.usbhd1' [OK]

Installation completed successfully.
```

2.2.4 Root filesystem Upgrade

You can upgrade root filesystem with the flashinstall script using the rootfs target option. A root filesystem upgrade cannot be performed if the system is currently running from flash disk. You can use Ramdisk or NFS boot for upgrading the root filesystem.

The following example downloads the rootfs tarball located in the `/tftpboot/9405` directory on the TFTP server with the IP address 172.16.166.55 and extracts the content to the flash disk.

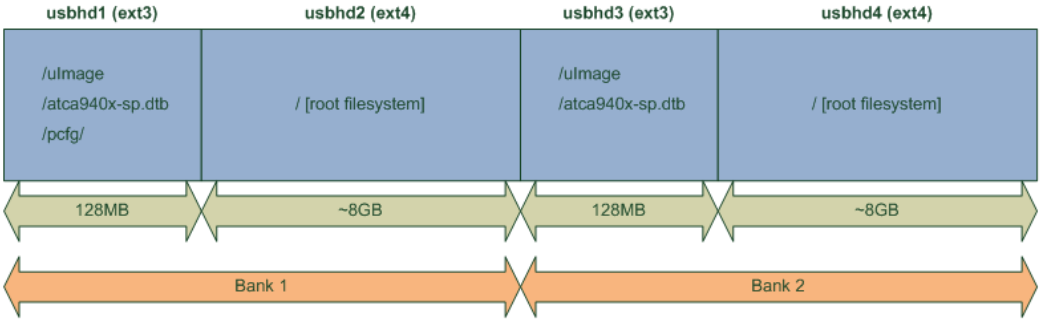
```
root@ATCA-9405-12-13:/root> flashinstall -i 172.16.166.55 -t
9405/atca9405_p2020-standard-glibc_std-dist.tar.bz2 -l rootfs-
inst.log rootfs
Installation started - this may take a while.
Please do not abort this script!

*) mounting '/dev/usbhd2' on '/mnt.flashinstall.usbhd2' [OK]
*) downloading '9405/atca9405_p2020-standard-glibc_std-
dist.tar.bz2' from tftp server '172.16.166.55' to
'/mnt.flashinstall.tmpfs/atca9405_p2020-standard-glibc_std-
dist.tar.bz2' [OK]
*) removing all files under '/mnt.flashinstall.usbhd2' [OK]
*) extracting '/mnt.flashinstall.tmpfs/atca9405_p2020-standard-
```

```
glibc_std-dist.tar.bz2' to '/mnt.flashinstall.usbhd2' [OK]
*) unmounting '/mnt.flashinstall.usbhd2' [OK]

Installation completed successfully.
```

2.2.5 Redundant Flash Installation

Optionally, the on-board flash can be used in redundant mode. In this mode the first 2 partitions are duplicated, therefore the disk uses 4 primary partitions. The first two partitions are summarized as Bank 1 and the last two partitions as Bank 2.The `flashinstall` tool has an option to specify which bank to be used for installation. The Figure 2-2 below shows the partition layout.

Figure 2-2 Partition Layout



For creating the redundant partitions, the `flashprep` tool should be used with the `-r` option. Note that all data on the disk will be lost. The following output shows the creation of the redundant partitions:

```
root@ATCA-9405-12-13:/root> flashprep -r
WARNING: All data on device '/dev/usbhd' will be lost!
Continue (y/n) y

*) unmounting /pcfg [OK]
*) removing any existing partitions on device '/dev/usbhd' [OK]
*) creating new primary partitions '/dev/usbhd1' - '/dev/usbhd4'
[OK]
*) creating an ext3 fs on '/dev/usbhd1' [OK]
```

```
*) tuning ext3 fs on '/dev/usbhd1' [OK]
*) creating an ext4 fs on '/dev/usbhd2' [OK]
*) tuning ext4 fs on '/dev/usbhd2' [OK]
*) creating an ext3 fs on '/dev/usbhd3' [OK]
*) tuning ext3 fs on '/dev/usbhd3' [OK]
*) creating an ext4 fs on '/dev/usbhd4' [OK]
*) tuning ext4 fs on '/dev/usbhd4' [OK]
```

Now, the `flashinstall` tool can be used to install kernel, device-tree blob and root filesystem to the desired bank number specified with the `-b` option. If no bank number is specified, the `flashinstall` tool uses bank number 1 to be compatible with the non-redundant installation mode. Note that the kernel, device-tree blob and root filesystem target of one software release needs to be installed on the same bank. Two different software releases can be installed on the two banks, but it is not supported to mix the different targets (kernel, dtb, rootfs) of two different software releases among two banks.

After installation, the `flashbootsel` tool should be used to specify the bank to be booted using the u-boot `run usbboot` command (see *Booting the onboard Flash Disk* on page 42). If `flashbootsel` is called without any option, it prints the currently selected bank number. Otherwise, if it is called with the `-b` option the bank number can be switched. The following output shows the current bank number and a switch to bank number 2.

```
root@ATCA-9405-12-13:/root> flashbootsel
Currently selected bank: 1 (usbdev=0:1, bdev=sda2)

root@ATCA-9405-12-13:/root> flashbootsel -b 2
Old selected bank: 1 (usbdev=0:1, bdev=sda2)
New selected bank: 2 (usbdev=0:3, bdev=sda4)
```

2.3 Booting Ramdisk on Management Processor

You can boot Linux on the service processor directly from RAM. It requires a TFTP server from which u-boot can load the necessary files. The network connection to the TFTP server is necessary only to download the required files.

Booting from RAM requires the following files from the release. These files must be stored on the TFTP server.

Table 2-3 Ramdisk Kernel

Component Type	Filename
Kernel	`atca9405_p2020-default_kernel_image-WR4.1.0.0_standard_<version>`
Device-tree-blob	`atca9405_p2020-p2020rdb.dtb-WR4.1.0.0_standard_<version>`
Ramdisk (initrd)	`atca9405_p2020-initrd-<version>.gz.uboot`

NOTICE

The ramboot provides no persistency. Any modification to the root filesystem is lost after reboot. This boot mechanism is for instance recommended to upgrade or install Linux on the flash disk.

2.3.1 Using Default Pre-configured u-boot Settings

By default, u-boot environment provides pre-configured settings for ramboot. The following u-boot environment settings should be checked and modified if necessary:

```
setenv ipaddr <target_ip_address>
setenv serverip <tftp_server_ip_address>
setenv netmask <netmask>
setenv ramdiskfile <ramdisk_file>
setenv bootfile <kernel_file>
setenv fdtfile <device-tree-blob_file>
```

To boot the Linux

- Execute the run ramboot command.
  This command loads the files from the TFTP server to the local memory and boots the Linux.

An example for booting the 1.0.7 release using the ramboot method is as follows:

```
ATCA-940X-12[SP]> setenv ipaddr 172.16.11.113
ATCA-940X-12[SP]> setenv serverip 172.16.254.254
ATCA-940X-12[SP]> setenv netmask 255.255.0.0
ATCA-940X-12[SP]> setenv ramdiskfile 9405/emr_atca9405_p2020-
initrd-1.0.7.gz.uboot
ATCA-940X-12[SP]> setenv bootfile 9405/emr_atca9405_p2020-
default_kernel_image
WR4.1.0.0_standard_EMRSN_1.0.7
ATCA-940X-12[SP]> setenv fdtfile 9405/emr_atca9405_p2020-
p2020rdb.dtb
WR4.1.0.0_standard_EMRSN_1.0.4
ATCA-940X-12[SP]>
ATCA-940X-12[SP]> run ramboot
Speed: 1000, full duplex
Using eth0 device
TFTP from server 172.16.254.254; our IP address is 172.16.11.113
Filename '9405/emr_atca9405_p2020-initrd-1.0.7.gz.uboot'.
Load address: 0x2000000
Loading: ##################
         91 MB received

         183 MB received
         ################################
done
Bytes transferred = 230766685 (dc1385d hex)
Speed: 1000, full duplex
Using eth0 device
TFTP from server 172.16.254.254; our IP address is 172.16.11.113
Filename '9405/emr_atca9405_p2020-default_kernel_image
WR4.1.0.0_standard_EMRSN_1.0.7'.
Load address: 0x1000000
Loading: ##################################################
done
Bytes transferred = 3262007 (31c637 hex)
Speed: 1000, full duplex
Using eth0 device
TFTP from server 172.16.254.254; our IP address is 172.16.11.113
Filename '9405/emr_atca9405_p2020-p2020rdb.dtb-
WR4.1.0.0_standard_EMRSN_1.0.4'.
```

```
Load address: 0xc00000
Loading: #################################################
done
Bytes transferred = 9745 (2611 hex)
WARNING: adjusting available memory to 30000000
## Booting kernel from Legacy Image at 01000000 ...
   Image Name:   Linux-2.6.34.8-WR4.1.0.0_standar
   Created:      2012-08-21  12:12:49 UTC
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    3261943 Bytes = 3.1 MiB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Loading init Ramdisk from Legacy Image at 02000000 ...
   Image Name:   Initrd-9405_p2020-1.0.7
   Created:      2012-08-21  12:27:44 UTC
   Image Type:   PowerPC Linux RAMDisk Image (gzip compressed)
   Data Size:    230766621 Bytes = 220.1 MiB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 00c00000
   Booting using the fdt blob at 0xc00000
   Uncompressing Kernel Image ... OK
   Loading Ramdisk to 223ec000, end 2ffff81d ... OK
   Loading Device Tree to 03ffa000, end 03fff610 ... OK
Using P2020 RDB machine description
Memory CAM mapping: 256/256/256 Mb, residual: 1280Mb
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Linux version 2.6.34.8-WR4.1.0.0_standard_EMRSN_1.0.7
(aas114@tesla) (gcc version 4.4.1 (Wind River Linux Sourcery G++
4.4a-323) ) #1 SMP PREEMPT Tue Aug 21 14:12:25 CEST 2012
Found initrd at 0xe23ec000:0xeffff81d
...
```

2.3.2 Using manual u-boot settings

If you want to specify your own ramboot settings, the following settings should be configured. These settings are required for the tftpboot and boot u-boot command.

```
setenv ipaddr <target_ip_address>
setenv serverip <tftp_server_ip_address>
setenv netmask <netmask>
setenv bootargs root=/dev/ram0 rw ramdisk_size=800000
console=ttyS0,9600
```

To boot the Linux manually

- Run the following command:
  ```
  tftpboot 1000000 <kernel_file>
  tftpboot c00000 <device-tree-blob_file>
  tftpboot 2000000 <ramdisk_file>
  bootm 1000000 2000000 c00000
  ```

  This command loads the files from the TFTP server to the local memory and boots the Linux.

2.4 Booting NFS-mounted Root File System on Management Processor

You can also boot Linux on the ATCA-9405 using NFS root filesystem. It requires a permanent connection to an external NFS server.

Like ramboot, this does not need any local storage. In addition, all files are persistent. Its primary intention is for development purpose.

Booting an NFS root filesystem requires the following files:

Table 2-4 NFS Root File System

Component Type	Filename
Kernel	`atca9405_p2020-default_kernel_image-WR4.1.0.0_standard_<version>`
Device-tree-blob	`atca9405_p2020-p2020rdb.dtb-WR4.1.0.0_standard_<version>`
Dist tarball	`atca9405_p2020-standard-glibc_std-dist-<version>.tar.bz2`

All files must be available on the NFS server. The dist tarball needs to be extracted on the NFS server with root permissions. Note that each ATCA-9405 blade needs its own nfs root filesystem.

Following is an example of file system structure on the NFS server:

```
+-/export/
  |
  +-nfsroot/
    |
    +-atca-9405/
      |
      +-<version>/
        |
        +-atca9405_p2020-default_kernel_image-
WR4.1.0.0_standard_<version>
        |
        +-atca9405_p2020-p2020rdb.dtb-WR4.1.0.0_standard_<version>
        |
        +-rootfs/
          |
          +-bin/
```

```
+-sbin/
+-tmp/
+-usr/
+-...
```

You can make your filesystem structure as above using the following command:

```
mkdir -p /export/nfsroot/atca-9405/<version>
cd /export/nfsroot/atca-9405/<version>
cp <release_dir>/atca9405_p2020-default_kernel_image
WR4.1.0.0_standard_<version> .
cp <release_dir>/atca9405_p2020-p2020rdb.dtb
WR4.1.0.0_standard_<version> .
mkdir rootfs
cd rootfs
sudo tar xfj <release_dir>/atca9405_p2020-standard-glibc_std-dist-
<version>.tar.bz2
```

For tftpboot and bootm u-boot commands, set the following u-boot environment:

```
setenv ipaddr <target_ip_address>
setenv serverip <nfs_server_ip_address>
setenv netmask <netmask>
setenv bootargs root=/dev/nfs rw console=ttyS0,9600
nfsroot=<nfs_server_ip>:/<nfs_server_rootfs_path>
ip=<target_ip>:<nfs_server_ip>:<gateway_ip>:<netmask>::eth0:
```

To boot the Linux manually

- Run the following command:
  ```
  nfs 1000000
  <nfs_server_ip>:/<nfs_server_rootfs_path>/<kernel_file>
  nfs c00000 <nfs_server_ip>:/<nfs_server_rootfs_path>/<device-
  tree-blob_file>
  bootm 1000000 - c00000
  ```

An example for booting the 1.0.7 release using the nfs root filesystem method is as follows:

```
ATCA-940X-12[SP]> setenv ipaddr 172.16.11.113
ATCA-940X-12[SP]> setenv serverip 172.16.254.254
ATCA-940X-12[SP]> setenv netmask 255.255.0.0
ATCA-940X-12[SP]> setenv bootargs root=/dev/nfs rw
console=ttyS0,9600
nfsroot=172.16.254.254:/export/nfsroot/atca-9405/1.0.7/rootfs
ip=172.16.11.113:172.16.254.254:172.16.254.254:255.255.0.0::eth0:
ATCA-940X-12[SP]>
ATCA-940X-12[SP]> nfs 1000000 172.16.254.254:/export/nfsroot/atca
9405/1.0.7/emr_atca9405_p2020-default_kernel_image
WR4.1.0.0_standard_EMRSN_1.0.7
Speed: 1000, full duplex
Using eth0 device
File transfer via NFS from server 172.16.254.254; our IP address is
172.16.11.113
Filename '/export/nfsroot/atca-9405/1.0.7/emr_atca9405_p2020-
default_kernel_image
WR4.1.0.0_standard_EMRSN_1.0.7'.
Load address: 0x1000000
Loading:
#################################################################

done
Bytes transferred = 3262007 (31c637 hex)
ATCA-940X-12[SP]> nfs c00000 172.16.254.254:/export/nfsroot/atca
9405/1.0.7/emr_atca9405_p2020-p2020rdb.dtb-
WR4.1.0.0_standard_EMRSN_1.0.4
Speed: 1000, full duplex
Using eth0 device
File transfer via NFS from server 172.16.254.254; our IP address is
172.16.11.113
Filename '/export/nfsroot/atca-9405/1.0.7/emr_atca9405_p2020-
p2020rdb.dtb
WR4.1.0.0_standard_EMRSN_1.0.4'.
Load address: 0xc00000
Loading: ##
done
Bytes transferred = 9745 (2611 hex)
ATCA-940X-12[SP]> bootm 1000000 - c00000
```

```
WARNING: adjusting available memory to 30000000
## Booting kernel from Legacy Image at 01000000 ...
   Image Name:   Linux-2.6.34.8-WR4.1.0.0_standar
   Created:      2012-08-21  12:12:49 UTC
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    3261943 Bytes = 3.1 MiB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 00c00000
   Booting using the fdt blob at 0xc00000
   Uncompressing Kernel Image ... OK
   Loading Device Tree to 03ffa000, end 03fff610 ... OK
Using P2020 RDB machine description
Memory CAM mapping: 256/256/256 Mb, residual: 1280Mb
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Linux version 2.6.34.8-WR4.1.0.0_standard_EMRSN_1.0.7
(aas114@tesla) (gcc version 4.4.1 (Wind River Linux Sourcery G++
4.4a-323) ) #1 SMP PREEMPT Tue Aug 21 14:12:25 CEST 2012
CPU maps initialized for 1 thread per core
bootconsole [udbg0] enabled
setup_arch: bootmem
mpc85xx_rdb_setup_arch()
Found FSL PCI host bridge at 0x0000000fffe0a000. Firmware bus
number: 0->9
PCI host bridge /pcie@ffe0a000  ranges:
 MEM 0x0000000c00000000..0x0000000c3fffffff -> 0x0000000080000000
  IO 0x0000000fffc00000..0x0000000fffc3ffff -> 0x00000000ffc00000
/pcie@ffe0a000: PCICSRBAR @ 0xfff00000
MPC85xx RDB board from Freescale Semiconductor
arch: exit
Zone PFN ranges:
  DMA      0x00000000 -> 0x00030000
  Normal   empty
  HighMem  0x00030000 -> 0x00080000
Movable zone start PFN for each node
early_node_map[1] active PFN ranges
    0: 0x00000000 -> 0x00080000
```

```
MMU: Allocated 1088 bytes of context maps for 255 contexts
PERCPU: Embedded 8 pages/cpu @c16e3000 s10740 r8192 d13836 u65536
pcpu-alloc: s10740 r8192 d13836 u65536 alloc=16*4096
pcpu-alloc: [0] 0 [0] 1
Built 1 zonelists in Zone order, mobility grouping on.  Total pages:
520192
Kernel command line: root=/dev/nfs rw console=ttyS0,9600
nfsroot=172.16.254.254:/export/nfsroot/atca-9405/1.0.7/rootfs
ip=172.16.11.113:172.16.254.254:172.16.254.254:255.255.0.0::eth0:
...
```

2.5 Booting the onboard Flash Disk

If Linux is installed properly on the onboard flash disk, you can boot Linux from the flash disk.

To boot the Linux from flash disk

- Run the `run usbboot` command

This boot mechanism requires no network connection. All files in the root filesystem are persistent. Make sure that always you shutdown the OS gracefully otherwise files or the whole filesystem may get damaged.

2.6 Service Processor Persistent Storage

The Service Processor has support for storing files persistent. The main purpose is to store configuration files persistent and blade specific independent of the current location of the rootfs.

The persistent files are stored in the `/pcfg` directory which is physically located on the first partition of the onboard flash disk. The `/pcfg` directory exists only once even if redundancy flash installation is used.

The advantages of the SP Persistent Storage support are:

- Blade specific and persistent specified config files even if RAM boot is used.
- Blade specific specified config files if a NFS root filesystem is used

- Persistency of specified config files if the on-board flash disk is updated
- Use of the same persistent config files on both banks of a redundant on-board flash disk installation

The files to be persistent are specified in the `pmap config` file located under:
`/pcfg/opt/bladeservices/etc/pmap`

The `pmap config` file specifies the file to be persistent and the location of the persistent file. For the exact syntax, inspect the file. The `pmap config` file itself is persistent so that it can be changed by the user.

Currently the following files are persistent by default however this may change in later releases:
`/opt/bladeservices/etc/mvpp.cfg`
`/etc/opt/srstackware/config/srs.cfg`

At startup the `pmap config` file is read and the specified files are replaced by symbolic links pointing to the target file within the `/pcfg` directory.

2.6.1 User Changes

If a user wants to add persistent config files, following steps are recommended:

1. Edit the `/pcfg/opt/bladeservices/etc/pmap config` file and add a line with the new config file to be persistent
2. Call the `mkpcfg` command or reboot
   The `mkpcfg` command saves the original config file as `<filename>.ORG` so it is always possible to manually restore or do comparison with the original config file.

2.6.2 Disabling Persistent Storage

If the persistent storage mechanism should not be used at all, then it can be turned off at boot time. The `pcfg` kernel command line option controls this. It can be set with the `u-boot bootargs` environment variable.

The following settings are supported:

- `pcfg=no` to disable persistent storage and restore original configuration
- `no pcfg` setting or another value than 'no' to use persistent storage support

NOTICE

Disabling persistent storage only takes care of the files within the original persistent config map. Therefore any user changes need to be restored manually.

If the onboard flash disk is re-partitioned, for example, by using the flashprep tool, then all persistent files are destroyed.

2.7 OCTEON Management

The ATCA-9405 has two cn68xx Octeon Processors managed via PCIe from the P2020 Management Processor.

We provide the following software components built with the latest Octeon-SDK 2.3 (with ATCA-9405 patch applied) from Cavium to manage and control the Packet Processors:

- Octeon remote utils
- Octeon u-boot bootloader
- Octeon Linux with initramfs

2.7.1 Octeon remote utils

The Octeon remote utils are provided as octremoteutils RPM package. The package has the following content:

- Compiled oct-remote binaries from Octeon-SDK with ATCA-9405 patch applied
- Oct script as a wrapper for the oct-remote binaries
- u-boot environment files

The sources of the oct-remote binaries are derived from the following path of the Octeon-SDK:

```
$OCTEON_ROOT/host/remote-utils/
```

Different remote connections are provided however for ATCA-9405 only PCI is currently supported. Note that the Octeon processors have no boot flash. Therefore oct-remote-boot is always necessary to boot the Octeon. The oct-remote-reset tool only resets the Octeon but does not load the bootloader.

The most important oct-remote binaries are listed in the following table:

Table 2-5 Octeon Remote Binaries

Name	Description	Notes
`oct-remote-boot`	Boots the Octeon and loads bootloader into Octeon	
`oct-remote-bootcmd`	Send bootloader commands to Octeon	Octeon need to stay in u-boot
`oct-remote-load`	Load a file into Octeon memory	Octeon need to stay in u-boot
`oct-remote-console`	Redirect the Octeon console	SE-S apps need cvmx pci console support
`oct-remote-csr`	Read/Write Octeon Config Status Register	

For more information and details, refer to the remote controlling html page of the Octeon-SDK documentation: $OCTEON_ROOT/docs/html/octeon_remote.html

2.7.2 Octeon remote wrapper

The oct script is the wrapper for the oct-remote binaries. It takes care of setting the necessary environment settings for the actual oct-remote binary. Furthermore it takes care of powering the Octeon, handling the I2C bus, checking the DRAM banks and generating the u-boot environment file. Therefore it is highly recommended to always call the oct wrapper script instead of the oct-remote executable directly.

The oct wrapper, oct <cpu_num (1|2)> <command-suffix> [<command-args>] has the following usage:

- cpu_num specifies which Octeon Processor to be addressed
- Command-suffix is the name of the tool to be executed without the "oct-remote-" prefix
- Command-args is a optional list of arguments that are passed 1:1 to the actual oct-remote executable

As an example to read the CSR register LMC0_FADR of Octeon 1, you should call:

```
oct 1 csr LMC0_FADR
```

2.7.3 Octeon u-boot environment files

The Octeon u-boot environment is hosted on the Management processor. It is supplied as a temporary file during the boot command. The temporary file is auto generated and can be configured by the two different environment files:

- /opt/bladeservices/octeon/octeon-cmn.env
- /opt/bladeservices/octeon/octeon<cpu_num>.env

The first environment file specifies the settings common for both Octeons while the second file only specifies settings for a particular Octeon. A very common use case for modifying the environment file would be changing the baudrate of the serial interface.

2.7.4 Booting the Octeon

This release provides a pre-compiled Octeon u-boot bootloader that can be directly used. It is compiled with the Octeon-SDK with ATCA-9405 patch applied.

The sources of the u-boot bootloader are derived from the following path of the Octeon-SDK:

```
$OCTEON_ROOT/bootloader/u-boot/
```

You can found the pre-compiled u-boot, boot-loader file in the following location:
`/opt/bladeservices/rom/u-boot-octeon_atca9405-<version>.bin`

To boot the Octeon

- Run the `oct <cpu_num> boot <bootloader_file>` command.

The oct wrapper internally calls the octtool to power on the Octeon.

As an example to boot the Octeon 1 with the provided bootloader call: `oct 1 boot /opt/bladeservices/rom/u-boot-octeon_atca9405-<version>.bin`

2.7.5 Running Linux on OCTEONS

The pre-installed RPM package atca9405_cn68xx-vmlinux_initramfs contains the Octeon Linux with initramfs (busybox based). It is compiled with the Octeon-SDK with ATCA-9405 patch applied.

The sources of Linux with initramfs are derived from the following path of the Octeon-SDK including the Octeon-Linux package:

`$OCTEON_ROOT/linux/`

You can found the pre-compiled Linux with initramfs in the following location:

```
/opt/bladeservices/octeon/vmlinux
```

The `/opt/bladeservices/octeon/runlinux` script can be used to start Linux on the Octeon. It first boots the Octeon, loads the Linux image to the Octeon and then start Linux on 4 cores. Please note that the console output is redirected to PCI. You can either use the oct console util to access the PCI console on the Management Processor or modify the script to use the serial console (by changing the 'console=' Linux bootarg).

The syntax of the runlinux script is:

```
/opt/bladeservices/octeon/runlinux <cpu_num>
```

The Octeon Ethernet driver responsible for the RXAUI/DXAUI interfaces are not compiled directly in the kernel but available as a module. It needs to be loaded manually since either a Simple Executive application controls the physical interfaces or a user might want to specify additional module parameters.

The follwing command loads the Octeon Ethernet driver with default module parameters:

```
modprobe octeon-ethernet
```

2.7.6 Running an Example of SE-S Application

The following commands are necessary to run an SE-S application on the Octeon Processor:

```
   oct <cpu_num> boot /opt/bladeservices/rom/u-boot-
octeon_atca9405-<version>.bin
  oct <cpu_num> load 0 <se-s file>
  oct <cpu_num> bootcmd "bootoct 0 coremask=<coremask> endbootargs"
```

An example coremask for running a SE-S application on all 32 cores is: `0xffffffff`

2.7.7 Powering the Octeons with octtool

The octtool can be used to power up or down an Octeon processor. It is located in the '/opt/bladeservices/bin' directory. When an Octeon processor is powered up, it is added to the Linux PCI subsystem as a new PCI device on an own bus. If it is powered down, it gets removed from the Linux PCI subsystem.

The following shows the lspci output of the Octeon PCI device if both processors are powered.

```
03:00.0 MIPS: Cavium Networks Device 0091 (rev 01)

04:00.0 MIPS: Cavium Networks Device 0091 (rev 01)
```

The bus number, 03 is always reserved for the first Octeon Processor while the bus number, 04 is for the second one regardless which one is powered first.

The octtool has the following usage:

```
root@ATCA-9405-12-13:/root> octtool -h

octtool v1.0

usage: octtool [--octeon=<1|2>] [command]

commands:
  -h|--help   show this help
  -s|--status show status of Octeon(s)
  -a|--add    add Octeon(s)
  -r|--remove remove Octeon(s)

if octeon is omitted then both Octeons are processed.
```

As an example the following command powers up only the first Octeon processor:

```
root@ATCA-9405-12-13:/root> octtool --octeon=1 -a
Adding Octeon 1 ... done.
Octeon 1: added
```

The following example powers down the first Octeon Processor:

```
root@ATCA-9405-12-13:/root> octtool --octeon=1 -r
Removing Octeon 1 ... done.
Octeon 1: removed
```

2.7.8 Debugging an SE-S application or the Linux kernel

The P2020 Management Processor rootfs contains a remote debugger to debug an Octeon SE-S application or Linux kernel over PCI.

Use the following commands to start debugging:

1. Start gdb by using the oct wrapper script (as you already do when using other oct-remote tools):
   `oct <cpu_num> gdb [options]`
2. Specify the boot command:
   `(Core#0-gdb) set pci-bootcmd oct <cpu_num> boot <bootloader_file>`

3. Connect to target, boot, and start application:
   ```
   (Core#0-gdb) target octeonpci bootoct 0 coremask=<coremask>
   ```

3.1 Introduction

Firmware for the ATCA-9405 is available in the following formats depending on the actual firmware component:

- HPM.1 image files
- FRI files
- Raw binary files

All components except the terminal server support the HPM.1 upgrade mechanism. HPM.1 is a PICMG standard to upgrade IPMCs or other components on behalf of the IPMC.

FRI files are also supported for some components. If a component can be upgraded via HPM.1 and FRI files then the FRI file should be preferred because the upgrade is faster.

Care has to be taken when a raw binary file is used to manually upgrade a component. In this case, no validation happens and it is possible to damage the component.

In general, FCU should be used when upgrading a firmware component. At the moment, FCU does not support all components and a manual upgrade is necessary for some components.

3.2 Backup Concept

Some firmware images on the ATCA-9405 are stored redundant on the respective storage devices. The following sections explain two different backup concepts used in ATCA-9405 in order to provide a safe upgrade mechanism.

3.2.1 Active/Backup Image Mechanism

Firmware on the ATCA-9405 blade supporting the Active/Backup image mechanism is stored in an active and backup bank on the storage device.

While upgrading the firmware, only the firmware in the backup bank is updated. After the upgrade, the new firmware is checked and if it is functional and valid, it will be switched to the new updated firmware.

This upgrade mechanism prevents having an incorrect firmware on both banks and allows the next firmware upgrade to be done easily.

The following firmware components use Active/Backup mechanism:

- IPMC Firmware
- MMC Firmware

3.2.2 Golden Image Mechanism

Firmware on the ATCA-9405 blade supporting the Golden image mechanism is stored in a golden and working bank on the storage device. In contrast to the Active/Backup image mechanism, only the Image in the working bank can be upgraded. The image in the golden bank is read-only and only used if the working image is broken. After the upgrade or after power-up (depends on the component), the working image is checked if it is valid. If the working image is valid then it is used. Otherwise if the image is broken, the golden image is used. In this case, you can try to upgrade the working image again.

Like the Active/Backup image mechanism, this mechanism also prevents having a bad firmware on both banks and allows the next firmware upgrade to be done easily.

The following firmware components use Golden Image Mechanism:

- Glue Logic FPGA
- ARTM Logic FPGA

3.3 Firmware Upgrade Facility

The Firmware Upgrade Facility (FUF) provides a uniform way to upgrade firmware on Artesyn hub blades, node blades, and AMC modules. It consists of a Firmware Upgrade Command-line Utility (FCU), flash device drivers, and specially prepared firmware recovery image files.

3.3.1 Firmware Recovery Image Files

FCU supports specially prepared firmware recovery image (FRI) files as well as firmware images in the HPM.1 format. HPM.1 is a PICMG standard to upgrade IPMCs.

By default, the image files for the current hardware configurations are loaded as part of the blade services software in `opt/bladeservices/rom`.

3.3.2 FUF Components

The following table lists the ATCA-9405 firmware components that can be handled by the Firmware Upgrade Facility and shows whether it is currently supported or not:

Table 3-1 FUF Components

Component	FCU Device Name	Type	Currently Supported	Upgrade Image Filename
IPMC Firmware	IPMI F/W	HPM.1	yes	atca9405-ipmc-fw-<version>.hpm
IPMC Bootloader	IPMI B/L	HPM.1	yes	atca9405-ipmc-boot-<version>.hpm
IPMC FRU Information	IPMI F/I	HPM.1	yes	atca9405-ipmc-fru-<version>.hpm
ARTM MMC Firmware	RTM IPMI F/W	HPM.1	yes	artm9405-mmc-fw-<version>.hpm
ARTM MMC Bootloader	RTM IPMI B/L	HPM.1	yes	artm9405-mmc-boot-<version>.hpm
ARTM MMC FRU Information	RTM IPMI F/I	HPM.1	yes	artm9405-mmc-fru-<version>.hpm
Glue Logic FPGA	PYLD FPGA	HPM.1	yes	atca9405-fpga-<version>.hpm
	atca-9405-fpga	FRI (Direct via SPI)	no	
ARTM Logic FPGA	RTM PYLD FPG	HPM.1	yes	artm9405-fpga-<version>.hpm
Switch EEPROM	PYLD MV EE	HPM.1	no	
SP Bootloader (u-boot)	PYLD F/W	HPM.1	no	
	atca-9405-p2020-cpu	FRI (Direct via SPI)	yes	atca940x-uboot-p2020-<version>.fri

3.3.3 FCU—Firmware Upgrade Command-Line Utility

The Firmware Upgrade Command-line Utility (FCU) allows you to:

- Query the current versions of firmware installed on a blade and determine which firmware devices are active.
- Verify that a specified upgrade image is sound and compatible with the current hardware.
- Upgrade a firmware image.

- Mark a device to be used as the boot source on the next reset.
- Switch between the active and stand-by IPMC firmware banks, without causing payload reset.
- Show the version of a firmware image file and compare it with the version of the currently installed firmware.

By default, the FCU binary executable is installed in `opt/bladeservices/bin`. This directory is specified in the PATH environment variable.

Some FCU operations require specially prepared upgrade files. For more details, refer *FUF Components* on page 53.

Synopsis

```
root@ATCA-9405-12-13:/root> fcu -h
USAGE:
    fcu [operations] [operands]

OPERATIONS:
    --query
    -q       Set to perform a query operation
    --upgrade
    -u       Upgrade the unused version of firmware.
             This operation requires the --file flag.
    --help
    -h       Displays this help message
    --show
    -s      Display information about the target which is included
in the given upgrade file.
             This operation requires the --file flag.
    --verify
    -v       Set to perform a verification of an upgrade file.
             This operation does NOT install the upgrade image.
             This operation requires the --file flag to be set.
    --mark
    -m       Mark the specified bank as next to boot.
             This operation depends on the --bank and
             the --device flag.
             NOTE : this operation will not cause a
```

```
                power cycle of the blade
    --compare
    -c          Compare the operation firmware with the
                image specified by the --file flag.
    --activate
    -r          Activate the specified bank.
                This operation depends on the --bank and
                the --device flag.
                NOTE : this operation could cause a
               power cycle of the blade depending on the capabilities
of the device
    --version
                Displays the version of this utility.

OPERANDS:
    --device=<device name>
    -d          Device to perform operation on.
    --file=<filename>
    -f          Filename of the firmware file
    --bank=<bankletter>
    -b          Bank-letter for mark/compare command
    --level=[0-7]
                severity level of logging. 7 logs everything.;
                default is 5
    --log=ARG
                file name for logging.

VALID FLAG COMBINATIONS:
--query
--query --device=<device name>
--upgrade --file=<filename>
--help
--show --file=<filename>
--verify  --file=<filename>
--verify --upgrade --file=<fileName>
--mark --bank=<bankletter> --device=<device-id>

--compare --file=<filename> --bank=0
```

```
--activate --bank=<bankletter> --device=<device-id>
--level=7
--log=/tmp/fcu.log
--version
```

Detailed Parameter Description

Some FCU parameters are described in more detail below:

- Parameter `--activate`
  The `--activate` Parameter can only be used for HPM.1 components. The banks of these components have the states operational, rollback, or deferred. Using this, option sets the state of the addressed bank to "operational".
  It depends on the HPM.1 component whether the activation request is executed immediately or deferred. If activation would require a payload reset then the activation request is deferred until next reset or power-up.
  If the `--upgrade` operation is used then the newly upgraded bank is activated automatically.
- Parameter `--mark`
  The `--mark` Parameter can be used for FRI components having redundant banks. Its intention is to just mark which bank is active at next boot.
  If the `--upgrade` operation is used then the newly upgraded bank is mark automatically.

The FCU verify and upgrade operations require specially prepared FRI files or HPM files; see *Firmware Recovery Image Files* on page 52.

3.3.4 Upgrading Firmware Image

This section describes recommended procedures for upgrading firmware devices.

3.3.4.1 IPMC and ARTM MMC Upgrade

The IPMC Upgrade conforms to the HPM.1 standard. It uses the Active/Backup Image Upgrade Mechansim. After the upgrade, IPMC checks for new firmware. If the new firmware is valid and functional, then IPMC switches to it. This happens without causing any payload reset.

The IPMC HPM.1 and the MMC HPM.1 image are provided as containers including the IPMC firmware. Depending on the version, even the IPMC bootloader or the FRU Information are also provided.

Upgrading the IPMC

The following is an example for upgrading the IPMC using FCU. It uses the HPM.1 IPMC image version 2.0.6 which includes the IPMC firmware and bootloader:

```
root@ATCA-9405-12-13:/opt/bladeservices/rom> fcu -uf atca9405-
ipmc-all-2.00.00000006.hpm
********************[[[[[REPORT BEGIN]]]]]********************
Operation: Upgrade
preparation stage started
capabilities of target and image header successfully compared
properties of component IPMI F/W retrieved
properties of component IPMI B/L retrieved
properties of component IPMI B/L retrieved
properties of component IPMI F/W retrieved
preparation stage successfully finished
  preparing upload
upload stage of IPMI B/L started
  initializing upload
  uploading ...100 %
  finishing upload
upload stage finished
upload stage of IPMI F/W started
  initializing upload
  uploading ...100 %
  finishing upload
upload stage finished
activation stage of new firmware started
  query self test result
activation stage successfully finished
Result   : Success
********************[[[[[ REPORT END ]]]]]********************
root@ATCA-9405-12-13:/opt/bladeservices/rom>
```

Upgrading ARTM MMC:

The following is an example for upgrading the ARTM MMC using FCU. It uses the HPM.1 MMC image version 2.0.7 which includes the ARTM MMC firmware and bootloader:

```
root@ATCA-9405-12-13:/opt/bladeservices/rom> fcu -uf artm9405-mmc-
all-2.00.00000007.hpm
********************[[[[[REPORT BEGIN]]]]]********************
Operation: Upgrade
preparation stage started
capabilities of target and image header successfully compared
properties of component RTM IPMI F/W retrieved
properties of component RTM IPMI B/L retrieved
properties of component RTM IPMI B/L retrieved
properties of component RTM IPMI F/W retrieved
preparation stage successfully finished
  preparing upload
upload stage of RTM IPMI B/L started
  initializing upload
  uploading ...100 %
  finishing upload
upload stage finished
upload stage of RTM IPMI F/W started
  initializing upload
  uploading ...100 %
  finishing upload
upload stage finished
activation stage of new firmware started
  target doesn't support self test
activation stage successfully finished
Result   : Success
********************[[[[[ REPORT END ]]]]]********************
root@ATCA-9405-12-13:/opt/bladeservices/rom>
```

3.3.4.2 Glue Logic FPGA and RTM Logic FPGA Upgrade

The FPGA upgrade for front board and RTM can be done using HPM.1 or FRI files. It uses the Golden Image upgrade mechansim. After the upgrade, a full power cycle is necessary to activate the newly upgraded FPGA firmware.

The following shows an example for upgrading the Glue Logic FPGA with FCU. It uses the HPM.1 FPGA image version 26 (0x1A):

```
root@ATCA-9405-12-13:/opt/bladeservices/rom> fcu -uf
glue_001A_working.bin.hpm
********************[[[[[REPORT BEGIN]]]]]********************
Operation: Upgrade
preparation stage started
capabilities of target and image header successfully compared
properties of component PYLD FPGA retrieved
properties of component PYLD FPGA retrieved
preparation stage successfully finished
  preparing upload
upload stage of PYLD FPGA started
  initializing upload
  uploading ...100 %
  finishing upload
upload stage finished
activation stage of new firmware started
  target doesn't support self test
activation stage successfully finished
Result   : Success
********************[[[[[ REPORT END ]]]]]********************
root@ATCA-9405-12-13:/opt/bladeservices/rom>
```

The following is an example for upgrading the RTM Logic FPGA with FCU. It uses the HPM.1 ARTM FPGA image version 20 (0x14):

```
root@ATCA-9405-12-13:/opt/bladeservices/rom> fcu -uf
artm_14_working.bin.hpm
********************[[[[[REPORT BEGIN]]]]]********************
Operation: Upgrade
preparation stage started
capabilities of target and image header successfully compared
properties of component RTM PYLD FPG retrieved
properties of component RTM PYLD FPG retrieved
preparation stage successfully finished
  preparing upload
upload stage of RTM PYLD FPG started
  initializing upload
  uploading ...100 %
  finishing upload
```

```
upload stage finished
activation stage of new firmware started
  target doesn't support self test
activation stage successfully finished
Result   : Success
********************[[[[[ REPORT END ]]]]]********************
root@ATCA-9405-12-13:/opt/bladeservices/rom>
```

3.4 Manual Firmware Upgrade

Manual firmware upgrade means upgrading a component without using FCU. The provided HPM.1 image files can be used along with every HPM.1 supported tool and the upgrade can also be done remotely.

The Service Processor u-boot provides a command to upgrade the following components by using binary files:

- Terminal Server
- SP u-boot
- Glue Logic FPGA

As already indicated in the introduction upgrading using raw binary files is not recommended and should only be done for components not supported by FCU.

3.4.1 Upgrading Firmware Images using SP U-Boot

Terminal Server

The manual firmware upgrade for the Terminal server needs to be done under u-boot. Make sure to have the u-boot prompt and necessary IP settings for TFTP transfer are set correctly. Furthermore, you must not use the terminal server at the moment.

The following steps upgrade the terminal server:

1. Transfer the terminal server image file to the blade:
   `ATCA-940X-<lslot>[SP]> tftp <image_file>`
2. Start the upgrade process:
   `ATCA-940X-<lslot>[SP]> firmware update ts`

Following is the example to upgrade the terminal server to version 1.3.30 :

```
ATCA-940X-12[SP]> tftp 9405/atca940x-ts-1.1.30.binary
Speed: 1000, full duplex
Using eth0 device
TFTP from server 172.16.254.254; our IP address is 172.16.11.113
Filename '9405/atca940x-ts-1.1.30.binary'.
Load address: 0x1000000
Loading: ##################################################
done
Bytes transferred = 43128 (a878 hex)
ATCA-940X-12[SP]> firmware update ts
Valid firmware image at 0x01000000, size 0x0000a878.
Updating firmware ...
Enabling EzPort on TS ...
Erasing flash ... done.
Writing: ##########################################
Restarting TS ...
Update done.
ATCA-940X-12[SP]>
```

SP U-Boot Upgrade

The manual firmware upgrade for the Service Processor bootloader u-boot is done under u-boot itself. Make sure to have the u-boot prompt and necessary IP settings for TFTP transfer are set correctly.

The following steps to upgrade the SP u-boot:

1. Transfer the SP u-boot image file to the blade:
   `ATCA-940X-<lslot>[SP]> tftp <image_file>`
2. Start the upgrade process.
   `ATCA-940X-<lslot>[SP]> firmware update cpu`

Following is an example to upgrade the SP u-boot from version 1.0.3 to 1.2.0 :

```
ATCA-940X-12[SP]> tftp 9405/atca940x-uboot-p2020-1.2.0.bin
Speed: 1000, full duplex
Using eth0 device
TFTP from server 172.16.254.254; our IP address is 172.16.11.113
```

```
Filename '9405/atca940x-uboot-p2020-1.2.0.bin'.
Load address: 0x1000000
Loading: ##################################################
done
Bytes transferred = 590848 (90400 hex)
ATCA-940X-12[SP]> firmware update cpu
Valid firmware image at 0x01000000, size 0x00090400.
Updating standby image from version 1.0.3 to version 1.2.0
Updating firmware ...
Erasing flash ... done.
Writing image ... done.
Activating image ... done.
Enabling FAILSAFE function ... done.
New version will be booted at next reset.
Update done.
ATCA-940X-12[SP]>
```

Glue Logic FPGA Upgrade

The Glue Logic FPGA upgrade can be done under u-boot. Make sure to have the u-boot prompt and necessary IP settings for TFTP transfer are set correctly.

1. Transfer the Glue Logic FPGA image file to the blade:
   `ATCA-940X-<lslot>[SP]> tftp <image_file>`
2. Start the upgrade process:
   `ATCA-940X-<lslot>[SP]> firmware update fpga`

Following is an example to upgrade the Glue Logic FPGA from version 25 (0x19) to 26 (0x1A) :

```
ATCA-940X-12[SP]> tftp 9405/glue_001A_working.bin
Speed: 1000, full duplex
Using eth0 device
TFTP from server 172.16.254.254; our IP address is 172.16.11.113
Filename '9405/glue_001A_working.bin'.
Load address: 0x1000000
Loading: ##################################################
done
Bytes transferred = 774157 (bd00d hex)
ATCA-940X-12[SP]> firmware update fpga
Valid firmware image at 0x01000000, size 0x000bd00d.
```

```
Updating FPGA image from version 00000019 to version 0000001A
Updating firmware ...
Update done.
ATCA-940X-12[SP]>
```

4.1 Overview

Hardware management in ATCA systems is based on the Intelligent Platform Management Interface (IPMI) specification. IPMI commands can be complex and cumbersome. To facilitate blade-level management, Artesyn provides the Hardware Platform Management (HPM) package that provides a set of commands that are based on IPMI commands but which are easier to use than the IPMI command itself. An HPM command can encapsulate a sequence of IPMI commands, for example reading the FRU Inventory data. An HPM command can be the unifier for OEM IPMI commands that are different on different blade types, for example reading the CPU boot bank. For a catalogue of supported IPMI commands of the blade, refer to the respective IPMI manual.

The HPM package consists of:

- HPM daemon, `hpmagentd`
- Command line utility, `hpmcmd`
- Script framework for managing shutdown, reboot, and local ekeying events

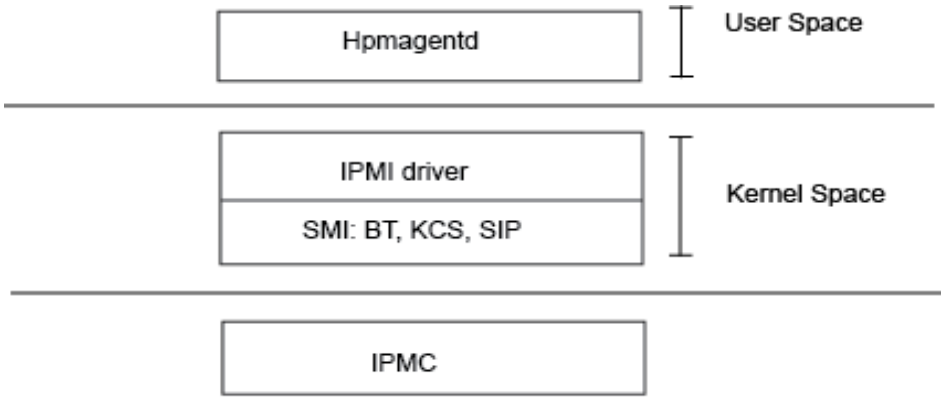
The HPM daemon is responsible to wait for events from the IPMC to perform a graceful shutdown/reboot of the operating system and to react when the link state of a channel's port is changed.

The utility `hpmcmd` displays the response of commands on the console in a human-readable format. HPM commands include:

- Retrieving and modifying FRU data
- Reading and controlling status of IPMI-controlled LEDs
- Communicating local slot location information
- Retrieving the event messages from the SEL of the IPMC

The hpmagentd and hpmcmdmake use of OpenIPMI driver to talk to the local IPMC. The following figure shows the software levels that are involved in the HPM architecture:

Figure 4-1 Software Levels of the HPM Architecture



BT: Block Transfer Interface
SIP: Serial Interface Protocol
SMI: System Maintainence Interface
KCS: Keyboard Control Style

The System Management Interface (SMI) driver provides the low level interface for talking to the IPMC and could be a KCS driver or Block Transfer (BT) driver or other. If you need more information about the software aspects of the blade IPM controller, refer to the respective IPMI manual.

4.2 hpmagentd—HPM Agent Daemon

4.2.1 Description

The hpmagentd is the service to process events from the local IPMC. For any incoming event, it calls the respective script which is part of the hpmagentcmd package. Event data is passed to a script by command line arguments. You can modify a script to fulfill your requirements. The following events are handled by the daemon:

Graceful Shutdown - When the IPMC receives an FRU activation request to deactivate an FRU, then it redirects the command to the `hpmagentd` through the IPMI driver. The `hpmagentd` invokes the shutdown script which is located in `/opt/bladeservices/bin/hpmshutdown`. By default within the script `shutdown -h now` is called which initiates a shutdown of the Linux immediately. Note that the IPMC powers down the processor in any case after a certain time. You may adjust this time with the `Graceful Shutdown Timeout` parameter of the IPMC, which can be set with a `SetSystemBootOption` IPMI command.

Graceful Reboot - On receiving an FRU control request to gracefully reboot the payload, the IPMC sends the command to the `hpmagentd`. The daemon invokes the reboot script which is available at `/opt/bladeservices/bin/hpmreboot`.

Ekeying Events - When the IPMC modifies the link state of a port, then it notifies the `hpmagentd` about that change. If the link goes down, then the script `hpmekeydown` is invoked otherwise the daemon calls `hpmekeyup`. Both scripts are placed in `/opt/bladeservices/bin`. By default these scripts are empty and do not do anything. To identify which port was changed, the `hpmagentd` passes an argument to an `hpmekeyup` script in the format: First the interface is specified: BC for Base Channel, FC for Fabric Channel, UC for Update Channel, and AMC for AMC Channel. The channel number and the port numbers of the channel are specified in the below example.

For example, the ports 1,2,3,4 of the channel 1 of a base interface is changed, then the argument looks like: "BC1.1,2,3,4".

4.2.2 Deployment

By default, the HPM daemon is installed in `/opt/bladeservices/bin`. With the `hpmagentd` binary, the scripts `hpmreboot`, `hpmshutdown`, `hpmekeyup`, and `hpmekeydown` are stored in that directory. Additionally, there are init script hpm to start and stop the daemon and the script hpmvar which exports some important variables to `/etc/default/hpmvars` to describe the board.

Synopsis

```
hpmagentd [options]
```

Parameters

```
 --log =<file name>
```

You may specify a log file for the daemon with this option. If you do not use it, then the `hpmagentd` logs to the syslog.

`-l  --level`

Specifies the level of message logging, where level is one of the standard syslog levels.

Log Level	Description
0	Emergency
1	Alert
2	Critical
3	Error
4	Warning
5	Notice (default)
6	Information
7	Debug

`-v --version`

Displays the version of the daemon

`-L --disable-led`

Disable the LED management

`-r --reboot-script=<script>`

Use the specified Blade Reboot script. Default is `/opt/bladeservices/bin/hpmreboot`

`-s --shutdown-script=<script>`

Use the specified Blade Shutdown script. Default is `/opt/bladeservices/bin/hpmshutdown`

`-u --ekey-up=<script>`

Called when a port is enabled. Default is `/opt/bladeservices/bin/hpmekeyup`

`-d --ekey-up=<script>`

Called when a port is disabled. Default is `/opt/bladeservices/bin/hpmekeydown`

`-h  --help`

Displays the help message

`-i --dont-daemonize`

Run interactively

4.2.3 hpm - Init.d Script

The `hpm init.d` script allows to start, stop, and restart the `hpmagentd`. It can be linked to a run level to automatically start the daemon at Linux boot time and to stop it when Linux shuts down.

Synopsis

```
hpm { start | stop | restart | status }
```

Parameters

`Start` - Starts the `hpmagentd`

`Stop`- Stops the `hpmagentd`

`Restarts` - Stops and Starts the `hpmagentd` again

`Status` - Reports if the `hpmagentd` is currently running or not

4.3 hpmcmd—HPM Command Utility

4.3.1 Overview

The HPM command utility talks directly to the IPMC through the IPMI driver which is part of the Linux kernel. It takes care of translating the user-friendly commands into the elaborated IPMI commands that the IPMC is able to understand. Those IPMI commands are transferred to the local IPMC. The HPM command utility can be started in interactive mode, where a prompt is displayed and the user enters commands; or it can process a single command.

By default, the `hpmcmd` binary is installed in `/opt/bladeservices/bin` directory. hpmcmd prints log messages to the syslog. By default, the log level is set to the severity error. You can modify the log severity by setting the environment variable HPMCMD_LOG_LEVEL. For example to set severity debug: export HPMCMD_LOG_LEVEL=7. For a list of log severities, refer to the table in *Deployment* on page 67.

Synopsis

```
hpmcmd [options]
```

Parameters

`-c` Processes a single command

`--help -h` Displays this help message

`-v` verbose mode for some commands

Some commands like `fruinfoget`, print more details if this options is given. Commands which do not support the verbose option ignores it.

`-t` Sends the command to a remote target.

For more information, refer *Target Addressing with hpmcmd* on page 71. If this option is not given, then the command goes to the local IPMC.

`-p` Changes the prompt when `hpmcmd` runs in interactive mode.

`-f` file option used by some hpmcmds

4.3.2 Target Addressing with hpmcmd

Using the `-t` option, you can send commands to other IPMCs or MMCs which participate on an IPMB. It has the following syntax: `-t <IPMB address>[: MMC address]`. The addresses must be set in hex format.

To send the command to an AMC attached on this blade use:

```
-t0:72 or -tlc:c0
```

To send the command to another IPMC type:

```
-t 92
```

To send the command to an MMC, which is attached on another blade in the shelf specify:

```
-t 82:72
```

4.3.3 Command Overview

The following table lists all commands from the `hpmcmd` program available on ATCA-9405. You can display this list and a short command description using the help command (see section *help* on page 89). A detailed description of the commands is given in section *Supported Commands* on page 73.

Table 4-1 Command Overview

Command	Description
bootbankget	Gets the bootbank to boot from
bootbankset	Sets the bootbank to boot from
bootparamerase	Erases boot parameter value
bootparameterrestore	Restores boot parameter section from the default
bootparamget	Gets boot parameter value
bootparamset	Sets a boot parameter value
chinfo	Retrieves channel information
cmd	Executes any IPMI command
deviceid	Gets the Device Id

Table 4-1 Command Overview (continued)

Command	Description
frudata	Allows to get FRU info in hex numbers
fruinfoget	Gets string fields from the FRU
fruinfoset	Sets string fields of the FRU
fruinv	Allows to get the FRU size and addressable units
frumrecset	Sets OEM multi-record of the FRU
fruread	Allows to read x number of bytes from the FRU
fruwrite	Allows to write x number of bytes from the FRU
fwprogevent	Sends a Firmware Progress Sensor Event
help	Gets list of commands
ipmbaddress	Gets the IPMB address
ipmcstatus	Gets the IPMC status
Lancfgget	Get LAN configuration parameter
Lancfgset	Set LAN configuration parameter
ledget	Gets the state of a specific FRU LED
ledprop	Get the LED properties for this FRU
ledset	Controls the state of a specific FRU LED
loglevelget	Gets the `hpmagentd` log level
macaddress	List the MAC addresses
partnumber	Gets the board part number
physlotnumber	Gets the board physical slot number
portget	Shows the current state E-Key governed intfs
portset	Enables/Disables ports in a channel
posttypeget	Gets the posttype to run at boot
posttypeset	Sets the posttype to run at boot
sdr	Shows the SDR records
sdr_dump	Shows the SDR records in raw format
sdrinfo	Shows SDR information

Table 4-1 Command Overview (continued)

Command	Description
sel	Shows the SEL records
selclear	Erases all contents from the SEL
selinfo	Shows SEL information
sendamc	Sends an IPMI request to an MMC behind a remote IPMC
sendcmd	Sends an IPMI request to an IPMB address IPMC
serialoutputget	Determines which serial output source goes to a particular serial port connector
serialoutputset	Select the serial output source of the serial port connector
shelfaddress	Gets the Shelf Address String
shelfslots	Prints the number of slots in the shelf
shelftype	Gets the Shelf Type from the Shelf FRU (Board Product Name)
slotmap	Prints the slotmap of the shelf
slotnumber	Shows the board slot number
solcfgget	Get SOL configuration parameter
solcfgset	Set SOL configuration parameter
version	Shows the `hpmcmd` version
watchdog	Controls Payload WDT functionality

4.3.4 Supported Commands

This section lists the supported commands. All commands are case insensitive. The examples illustrate the use of `hpmcmd` in single-command mode (-c). If you start `hpmcmd` without the `-c` option (that is, interactive mode), you simply enter these commands at the HPM command prompt.

4.3.4.1 bootbankget

Description

This command retrieves the boot bank which is currently marked as active for the CPU specified by `payload_cpu_selector`.

Firmware for the CPU on Artesyn ATCA blades is stored in redundant, persistent memory devices. This allows the firmware image in one bank to serve as a backup for the other bank. During normal operation, the CPU on a blade determines which bank to boot from based on a GPIO signal controlled by the IPMC. This bank is considered the active boot device.

Because you can change the “active” device with the hpmcmd `bootbankset` command, active status does not necessarily indicate which device was used on the last boot. It simply represents which device is set to be used on the next boot.

Synopsis

```
bootbankget <payload_cpu_selector>
```

Parameters

`payload_cpu_selector`

Is an integer between 0 and the number of CPU devices supported on the blade.

Example

```
hpmcmd -c bootbankget 0
```

BANK1

4.3.4.2 bootbankset

Description

This command sets the boot bank for a particular CPU from which the blade is supposed to boot.

Synopsis

```
bootbankset <payload_cpu_selector> <newBootBank>
```

Parameters

`payload_cpu_selector`

Is an integer between 0 and the number of CPU devices supported on the blade.

`newBootBank`

Can be set to `BANK1`, `BANK2`...

Example

```
hpmcmd -c bootbankset 0 BANK 1
```

4.3.4.3 bootparamerase

Description

This command erases the boot parameter.

Synopsis

```
bootparamerase section [name] [-t ipmbAddr[:mmcAddr]]
```

Parameters

`section`

Can have value as `USER`, `DEFAULT`, `TEST`, or `OS_PARAM`.

`name`

Specifies name of the parameter.

`t`

Sends the command to `ipmbAddr:mmcAddr`.

4.3.4.4 bootparamget

Description

This command gets the boot parameter value.

Synopsis

```
bootparamget section [name] [-t ipmbAddr[:mmcAddr]]
```

Parameters

`section`

Can have value as `USER, DEFAULT, TEST`, or `OS_PARAM`.

`name`

Specifies name of the parameter.

`t`

Sends the command to `ipmbAddr:mmcAddr`.

4.3.4.5 bootparamset

Description

This command sets the boot parameter value.

Synopsis

```
bootparamset section name=value [-t ipmbAddr[:mmcAddr]]
```

Parameters

`section`

Can have value as `USER, TEST`, or `OS_PARAM`.

`name`

Specifies name of the parameter.

`t`

Sends the command to `ipmbAddr:mmcAddr`.

4.3.4.6 chinfo

Description

Retrieves information about an IPMI Channel.

Synopsis

`Chinfo <channel>`

Parameters

`Channel`-Channel number

Example

`hpmcmd -c chinfo 1`

```
Channel Medium Type   : Asynch. Serial/Modem (RS-232)
Channel Protocol Type : TMode
Session Support       : session-less
Active Session Count  : 0
Protocol Vendor ID    : 00400A
```

`hpmcmd -c chinfo 0`

```
Channel Medium Type   : IPMB (I2C)
Channel Protocol Type : IPMB-1.0
Session Support       : session-less
Active Session Count  : 0
Protocol Vendor ID    : 001BF2
```

`hpmcmd -c chinfo 5`

```
Channel Medium Type   : OEM
Channel Protocol Type : OEM
Session Support       : session-less

Active Session Count  : 0
Protocol Vendor ID    : 0065CD
```

`hpmcmd -c chinfo 4`

```
Channel Medium Type   : System Interface (KCS, SMIC, or BT)
Channel Protocol Type : KCS
Session Support       : session-less
Active Session Count  : 0
Protocol Vendor ID    : 001BF2
```

4.3.4.7 cmd

Description

This command allows you to enter commands understood by the IPMC. Commands are entered as a sequence of hexadecimal numbers as defined in the *IPMI 1.5 Specification*.

Synopsis

```
cmd <ipmi address> <netfn cmd> <cmd data>
```

Parameters

`ipmi command`

The ipmi command specifies the sequence of hexadecimal bytes as entered using the ipmicmd tool from the OpenIPMI library. The ipmi command can have value, such as:

```
0f 00 XX ZZ W1 W2 ... Wn
```

In this example:

XX specifies netfunct in hexadecimal.

ZZ specifies the command number, as stated in the IPMI/PICMG spec.

W1 to Wn specifies the data bytes according to the command supports.

`ipmi address`

The IPMI address specifies the IPMC that receives the command, it can be the local IPMC or another IPMC on the IPMB. The IPMI address for the local IPMC consists of <f LUN>, where f is the BMC channel number. The IPMI address for a remote IPMC consists of <0 SA LUN>, where SA is the slave address.

`netfn cmd`

Identifies the command type.

`cmd data`

Specifies the message data associated with the command.

Example

GetDeviceId command to the local IMPC:

```
hpmcmd -c cmd f 0 6 1
```

GetDeviceId command to the remote IPMC on address 9a:

```
hpmcmd -c cmd 0 9a 0 6 1
```

4.3.4.8 deviceid

Description

This command retrieves the raw IPMI Get Device ID response and decodes the IPMI message.

Synopsis

```
deviceid -t [ipmbAddr[:mmcAddr]]
```

Parameters

`-t`

Sends the command to `ipmbAddre:mmcAddr`.

Example

```
hpmcmd -c deviceid

DEVICEID INFORMATION

--------------------

Device Id           =  0x00
Device Revision     =  0x00
Device Mode         =  Normal Operation; Supports Device SDR
Firmware Revision   =  2.00
```

```
IPMI Version        =  1.5
Device Support      =  IPMB Evnt Gen; FRU; SEL; Sensor;
Manufacturer ID     =  0x000065CD
Product ID          =  0x002B
Auxiliary Revision  =  0x00000013
```

4.3.4.9 frudata

Description

This command dumps the content of the FRU data in hexadecimal format.

Synopsis

```
frudata <fruid> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`

Is 0 for CPU.

`-t`

Sends the command to `ipmbAddr:mmcAddr`.

Example

```
hpmcmd -c frudata 0
75 2d 69 6e 66 6f 2e 69 6e 66 c1 00 00 00 00 15
01 07 19 c7 45 4d 45 52 53 4f 4e ce 41 54 43 41
2d 39 34 30 35 2d 33 32 47 42 cb 30 31 30 36 38
37 31 48 31 34 44 c0 c7 45 30 46 39 45 38 43 c0
c0 c1 00 00 00 00 00 5f c0 02 84 dd dd cd 65 00
01 01 0e 03 06 01 ec 9e cd 08 17 a2 11 06 01 ec
9e cd 08 17 a3 11 06 01 ec 9e cd 08 17 a4 11 06
```

```
01 ec 9e cd 08 17 a5 11 06 01 ec 9e cd 08 17 a6
11 06 01 ec 9e cd 08 17 a7 11 06 01 ec 9e cd 08
17 a8 11 06 01 ec 9e cd 08 17 a9 11 06 01 ec 9e
cd 08 17 aa 11 06 01 ec 9e cd 08 17 ab 11 06 01
ec 9e cd 08 17 ac 11 06 01 ec 9e cd 08 17 ad 11
06 01 ec 9e cd 08 17 ae 05 06 01 ec 9e cd 08 17
af c0 02 09 bd 78 cd 65 00 10 00 00 00 00 01 c0
02 82 a8 14 5a 31 00 14 00 01 90 0e 40 9a 0d 3f
5d bb b9 4a 87 2f 36 aa 37 6e 01 11 00 00 02 11
00 00 41 2f 40 00 41 2f 10 00 41 2f 30 00 41 2f
00 00 41 23 30 00 41 23 00 00 41 21 30 00 41 21
00 00 41 2f 13 00 41 2f 03 00 41 23 03 00 41 21
03 00 42 2f 40 00 42 2f 10 00 42 2f 30 00 42 2f
00 00 42 23 30 00 42 23 00 00 42 21 30 00 42 21
00 00 42 2f 13 00 42 2f 03 00 42 23 03 00 42 21
03 00 81 01 0f 00 c0 02 08 57 df 5a 31 00 1a 00
02 01 01 c0 02 0c 97 9b 5a 31 00 17 00 32 00 05
01 72 1e ff c0 02 0a d6 5e 5a 31 00 18 00 81 01
00 05 00 c0 82 10 59 55 5a 31 00 19 00 00 00 01
e5 ff ff 00 21 00 00 fe 00 00 00 00 00 00 00 00
```

.....

4.3.4.10 fruinfoget

Description

This command retrieves information from the specified FRU.

Synopsis

```
fruinfoget <fruid> [field] [-v] [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`

Is 0 for CPU.

`field`

Is one of the following data fields. If no field is specified, it retrieves the whole fruinfo for that FRU.

Field	Description
bmanufacturer	Board area manufacturer
bproductname	Board area product name
bserialnumber	Board area serial number
bpartnumber	Board area part number
pmanufacturer	Product area manufacturer
pproductname	Product area product name
ppartnumber	Product area part number
pversion	Product area version
pserialnumber	Product area serial number
passettag	Product area asset tag

`-v`

Verbose mode to get point-to-point connectivity information when no specific field is requested.

`-t`

Sends the command to `ipmbAddr:mmcAddr`.

Example

```
hpmcmd -c fruinfoget 0

Common Header:
    Format Version = 1

Board Info Area:
    Version   = 1
    Language Code            = 25
    Mfg Date/Time            = May 29 09:53:00 2013 (9156113 minutes
                               since 1996)
    Board Manufacturer       = EMERSON
    Board Product Name       = PCA,ATCA-9405-1200
    Board Serial Number      = E0F9E8C
    Board Part Number        = 0106871H14D
    FRU Programmer File ID   = fru-info.inf

Product Info Area:
    Version   = 1
    Language Code            = 25
    Manufacturer Name        = EMERSON
    Product Name             = ATCA-9405-32GB
    Product Part / Model#    = 0106871H14D
    Product Version          =
    Product Serial Number    = E0F9E8C
    Asset Tag                =
    FRU Programmer File ID   =

Multi Record Area:

    OEM MAC Addresses Record (ID=0x01)
        Version = 1

    Emerson Unknown Record (ID=0x10)
```

```
PICMG Board Point-to-Point Connectivity Record (ID=0x14)
    Version = 0

AMC Carrier Information Table Record (ID=0x1a)
    Version = 0

AMC Carrier Activation and Current Management Record (ID=0x17)
    Version = 0

AMC Carrier Point-to-Point Connectivity Record (ID=0x18)
    Version = 0

AMC Point-to-Point Connectivity Record (ID=0x19)
    Version = 0
```

4.3.4.11 fruinv

Description

This command retrieves the FRU size and the addressable unit for the specified FRU.

Synopsis

```
fruinv <fruid> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`

Is 0 for CPU.

`-t`

Sends the command to `ipmbAddr:mmcAddr`.

Example

```
hpmcmd -c fruinv 0

FruSize = 1024
```

```
Accessed Units = Bytes
```

4.3.4.12 frumrecset

Description

This command provides a way to write OEM multirecords to the FRU Inventroy Info. The format of the multirecord is defined by a XML file. This file is also to display the OEM data with the hpmcmd fruinfoget.

Synopsis

```
<multirecord version="1.0" name="" record_type="" manufacturer=""
record_id="" record_format="">
   <field type="" name="">DATA</field>
   ...
   <field type="" name="">DATA</field>
</multirecord>
```

Parameters

`version`

version of this XML format

`record_type`

A value which is equal to `0xc0` must be taken to identify the record as a multi record.

`manufacturer`

A value of a 3 bytes to identify the manufacturer. You can define an integer value like 0x123456 or the name of the manufacturer. `hpmcmd` know the IANA ID of some manufacturers.

`record_id`

This ID identifies the record together with the manufacturer ID. This field is one byte value.

`record_format`

Version of the data representation of the record. One byte value only.

NOTICE

The above attributes are part of multicord header information.

Parameters

`type`

With this attribute, you specify how the fields data is stored in the record. Currently, only 8BIT_ASCII is supported.

`name`

A name representing the information of the fields data. The name is not stored in the record but it is used to display the record data.

NOTICE

The above attributes are part of field element inside the multicord.

The DATA of the field must be placed in the text section of the field element. The maximum length of the data is 63 bytes, which is limited by the IPMI type/length byte format. For more detailed information about this format, refer to the IPMI FRU Inventory specification.

The maximum length of the record is 250 bytes. So you have to verify that the DATA length of all fields doesn't exceed this value.

Example

Using the example XML file, skspruce-example.xml

```
hpmcmd -c frumrecset 0 -f skspruce-example.xml -b /pcfg
```

You can see the new record with hpmcmd -c fruinfoget 0 -v

-b /pcfg writes the recovery file to the persistent filesystem area. You may restore the original FRU info with hpmcmd -c fruinfoset 0 -f /pcfg/fru-info-<recovery file>.

If you do not want to enter the values into the XML file, you may also generate the XML file with a script.

4.3.4.13 fruread

Description

This command gets nBytes of fruid from the startAddress of the specified FRU.

Synopsis

```
fruread <fruid> <startAddress> <nBytes> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`

Is 0 for CPU.

`startAddress`

Is the starting address for reading the fruid.

`nBytes`

Number of bytes to read in decimal.

`-t`

Sends the command to `ipmbAddr:mmcAddr`.

Example

```
hpmcmd -c fruread 0 0 280
01 00 00 01 0a 11 00 e3 01 09 19 11 b6 8b c7 45
4d 45 52 53 4f 4e d2 50 43 41 2c 41 54 43 41 2d
39 34 30 35 2d 31 32 30 30 c7 45 30 46 39 45 38
```

```
43 cb 30 31 30 36 38 37 31 48 31 34 44 cc 66 72
75 2d 69 6e 66 6f 2e 69 6e 66 c1 00 00 00 00 15
01 07 19 c7 45 4d 45 52 53 4f 4e ce 41 54 43 41
2d 39 34 30 35 2d 33 32 47 42 cb 30 31 30 36 38
37 31 48 31 34 44 c0 c7 45 30 46 39 45 38 43 c0
c0 c1 00 00 00 00 00 5f c0 02 84 dd dd cd 65 00
01 01 0e 03 06 01 ec 9e cd 08 17 a2 11 06 01 ec
9e cd 08 17 a3 11 06 01 ec 9e cd 08 17 a4 11 06
01 ec 9e cd 08 17 a5 11 06 01 ec 9e cd 08 17 a6
11 06 01 ec 9e cd 08 17 a7 11 06 01 ec 9e cd 08
17 a8 11 06 01 ec 9e cd 08 17 a9 11 06 01 ec 9e
cd 08 17 aa 11 06 01 ec 9e cd 08 17 ab 11 06 01
ec 9e cd 08 17 ac 11 06 01 ec 9e cd 08 17 ad 11
06 01 ec 9e cd 08 17 ae 05 06 01 ec 9e cd 08 17
af c0 02 09 bd 78 cd 65
```

4.3.4.14 fruwrite

Description

This command allows to write hexadecimal byte values to `fruid` starting at `startAddr`.

Synopsis

```
fruwrite <fruid> <startAddress> <hexval1> [hexval2] [...]
[hexval16] [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`

Is 0 for CPU.

`startAddress`

Starting address for writing.

`hexval1 .. hexvalN`

Is the hexadecimal value to write.

`-t`

Sends the command to `ipmbAddr:mmcAddr`.

4.3.4.15 fwprogevent

Description

This command sends a Firmware Progress Sensor Event to the Shelf Manager SEL. Refer IPMI specifications for details on values.

Synopsis

```
fwprogevent <data1> <data2> <data3>
```

Parameters

`data1`

Stores hexadecimal value as; 00 for Error, 01for Hang, and 02 for Progress.

`data2`

Stores hexadecimal value as; 00-0D for Error, 00-19 for Hang or Progress.

`data3`

Stores hexadecimal value as; FF unless an OEM data2 is specified.

4.3.4.16 help

Description

This command lists the available commands from the hpmcmd program with a brief explanation about the command.

Synopsis

```
help
```

4.3.4.17 ipmbaddress

Description

This command retrieves the blade IPMB address.

Synopsis

```
ipmbaddress
```

```
Example:
```

```
hpmcmd -c ipmbaddress
```

```
Ipmbaddress is 0x92
```

4.3.4.18 ipmcstatus

Description

This command retrieves the status of given IPMC.

Synopsis

```
ipmcstatus [-t ipmbAddr]
```

Parameters

`-t`

Specifies the target with `ipmbAddr`.

Example

```
hpmcmd -c ipmcstatus
```

```
IPMC Mode                =  NORMAL
Payload Control          =  Enabled
IPMC Outstanding Events  =   None
```

4.3.4.19 Lancfgget

Description

Get LAN configuration parameter

Synopsis

```
lancfgget <channel> [param]
```

Parameters

`channel` - channel number

`param` -

```
auth-type-support|
auth-type-enables
ip-addr
ip-addr-src
mac-addr
subnet-mask
ipv4-header-params
primary-rmcp-port
secondary-rmcp-port
bmc-generated-arp-control
gratuitous-arp-interval
default-gateway-addr
default-gateway-mac-addr
backup-gateway-addr
backup-gateway-mac-addr
community-string
num-destinations
destination-type
destination-addr
vlan-id
```

```
vlan-prio
rmcp-cipher-support
rmcp-ciphers
rmcp-priv-levels
dst-addr-vlan-tags
```

Example

```
hpmcmd -c lancfgget 5
```

```
IP Address        : 211.21.168.192
Subnet Mask       : 0.255.255.255
Default Gateway   : 0.0.0.0
MAC Address       : ec:9e:cd:02:d1:a4
```

4.3.4.20 Lancfgset

Description

Set LAN configuration parameter

Synopsis

```
lancfgset <channel> <param> <value>
```

Parameters

`channel` - channel number

`param` - ip-addr
subnet-mask
default-gateway-addr

`value` - IP address in string format

Example

```
hpmcmd -c lancfgset 4 ip-addr 192.168.24.10
```

4.3.4.21 ledget

Description

This command gets information about a specified LED controlled by the IPMC.

Synopsis

```
ledget <fruid> <led> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`

Is 0 for the main blade, 5 for the RTM.

`led`

Is BLUE for the hot swap LED or LEDN for FRU LED<n>. <n> is a number between 1 and the maximum FRU LEDs supported by the blade.

`-t`

Sends the command to `ipmbAddr:mmcAddr`.

Example

```
hpmcmd -c ledget 0 led1

Current State = OVERRIDE

State Function/(ms) Duration(ms)      Color
---------------------------------------------
Override         Off       Always        Red
```

4.3.4.22 ledprop

Description

This command displays the FRU LED properties under IPMC control.

Synopsis

```
ledprop <fruid>
```

Parameters

`fruid`

Is 0 for the main board and 5 for the RTM.

Example

```
hpmcmd -c ledprop 0

FRU LEDs under IPMC control:
    LED0 = Blue, Default: Blue
    LED1 = Amber,Red, Default: Red
    LED2 = Green,Red,Blue, Default: Green
    LED3 = Amber, Default: Amber
```

4.3.4.23 ledset

Description

This command controls the override state of a specific FRU LED.

Synopsis

```
ledset <fruid> <led> <operation> [offms] [onms] [color] [-t
ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`

Is 0 for the main blade and 5 for the rear transition module.

`led`

Is BLUE for the hot swap LED or LEDN for FRU LED<n>. <n> is a number between 0 and the maximum FRU LEDs supported by the blade.

`operation`

ON = enable override state and turn LED on.
OFF = enable override state and turn LED off.BLINK = enable override state and blink LED; off_duration and on_duration specify the blink duration; the default on and off duration is 300 ms.|
LOCAL = cancel override state and restore LED control to the IPMC, that is, local state.
TEST = run lamp test for specified on_duration, then restore prior state. The default duration is 5000ms.

`offms`

Specifies OFF duration in milliseconds. It can have value from 10ms to 2500ms in the 10ms increments. It is valid only if the operation is BLINK.

`onms`

Specifies ON duration in milliseconds. It can have value from 10ms to 2500ms in the 10ms increments. It is valid only if the operation is BLINK.

`color`

LED0 = BLUE
LED1 = RED or AMBER
LED2 = GREEN (if supported by IPMC)
LED3 = AMBER (if supported by IPMC)

`-t ipmbAddr`

Sends the command to `ipmbAddr:mmcAddr`.

Example

```
hpmcmd -c ledset 0 led1 on
```

4.3.4.24 loglevelget

Description

This command retrieves the current `hpmagentd` log level. The log level of `hpmcmd` can be set with the environment variable `HPMCMD_LOG_LEVEL`, for example export `HPMCMD_LOG_LEVEL=7` to set debuglevel. All log messages are sent to the syslog.

LogLevels:

0 Emergency
1 Alert
2 Critical
3 Error
4 Warning
5 Notice
6 Information
7 Debug

Synopsis

```
loglevelget
```

Example

```
hpmcmd -c loglevelget
```

5

4.3.4.25 macaddress

Description

This command retrieves a list of available MAC addresses.

Synopsis

```
macaddress [fruid]
```

Parameter

`Fruid`

Is 0 for CPU

Example

```
hpmcmd -c macaddress 0
Front or Rear Panel Interface MAC Addr :  ec:9e:cd:08:17:a2
Multi-Type Interface       MAC Addr :  ec:9e:cd:08:17:a3
Multi-Type Interface       MAC Addr :  ec:9e:cd:08:17:a4
Multi-Type Interface       MAC Addr :  ec:9e:cd:08:17:a5
Multi-Type Interface       MAC Addr :  ec:9e:cd:08:17:a6
```

```
Multi-Type Interface       MAC Addr :  ec:9e:cd:08:17:a7
Multi-Type Interface       MAC Addr :  ec:9e:cd:08:17:a8
Multi-Type Interface       MAC Addr :  ec:9e:cd:08:17:a9
Multi-Type Interface       MAC Addr :  ec:9e:cd:08:17:aa
Multi-Type Interface       MAC Addr :  ec:9e:cd:08:17:ab
Multi-Type Interface       MAC Addr :  ec:9e:cd:08:17:ac
Multi-Type Interface       MAC Addr :  ec:9e:cd:08:17:ad
Multi-Type Interface       MAC Addr :  ec:9e:cd:08:17:ae
Serial over Lan Interface  MAC Addr :  ec:9e:cd:08:17:af
```

4.3.4.26 partnumber

Description

This command retrieves the part number (FRU 0) of the main blade.

Synopsis

```
partnumber
```

Example

```
hpmcmd -c partnumber
```

4.3.4.27 physlotnumber

Description

This command retrieves the physical slot number in which the blade is plugged in.

Synopsis

```
physlotnumber
```

4.3.4.28 portget

Description

This command shows the current state of interfaces governed by e-keying. If no channel is specified, portget returns data for all channels in the specified interface. If neither interface nor channel are specified, portget will return data for all interfaces.

Synopsis

```
portget [interface] [channel] [devid]
```

Parameters

`interface`

Valid values are:

```
BASE | FABRIC | UPDATE | AMC
```

`channel`

It is a number from 1 to the maximum number of channels supported for the interface. Node blades usually support 2 Base and 2 Fabric channels, and switch blades support 16 Base, 15 Fabric, and 1 Update channels.

`devid`

For AMC only: it is an on-Carrier device ID that identifies the on-Carrier device to which the desired channel is connected.

Example

```
hpmcmd -c portget
```

```
  STATE   INTERFACE   CHANNEL   LINKTYPE   LINKEXT   GROUP   PORTS
-------------------------------------------------------------------
 ENABLED      BASE       1         BASE       0       0     0
 ENABLED      BASE       2         BASE       0       0     0
 ENABLED    FABRIC       1        ETHER       1       0     0,1,2,3
DISABLED    FABRIC       1     RESERVED       1       0     0,1,2,3
DISABLED    FABRIC       1     RESERVED       0       0     0,1,2,3
```

```
DISABLED   FABRIC      1    RESERVED      0       0    0,1
 ENABLED   FABRIC      2       ETHER      1       0    0,1,2,3
DISABLED   FABRIC      2    RESERVED      1       0    0,1,2,3
DISABLED   FABRIC      2    RESERVED      0       0    0,1,2,3
DISABLED   FABRIC      2    RESERVED      0       0    0,1
DISABLED   UPDATE      1         OEM      0       0    0
 ENABLED  AMC 00h      0  PCIEXPRESS      0       0    0
```

4.3.4.29 portset

Description

This command enables and disables ports in a channel. The following table lists the valid values for each parameter.

Synopsis

```
portset <intf> <chan> <grpid> <type> <typeX> <ports> <oper> [devid]
[-t ipmbAddr[:mmcAddr]]
```

Parameters

`intf`

Valid values are:

```
BASE | FABRIC | UPDATE | AMC
```

`chan`

It is a number from 1 to the maximum number of channels supported for the interface. Node blades usually support 2 Base and 2 Fabric channels, and switch blades support 16 Base, 15 Fabric, and 1 Update channels.

`grpid`

Specifies the group id. Always 0 according to the current shelf FRU information.

`type`

Valid values are:

```
BASE | ETHER | EXPRESS | INFINI | STAR | OEM
```

`typeX`

Valid values are:
`0` (for 1000Base-BX)
`1` (for 10GBase-BX4)
`2` (for FC-PI)

`ports`

A sequence of ports to act on.
For base and update channels, port is always 0.
For fabric channels, port can specify up to 4 ports as specified in PICMG 3.1:
Option 1: 0 (for port 0)
Option 2: 01 (for ports 0,1)
Option 3: 0123 (for ports 0,1,2,3)
Option 7: 3 (for port 3)

`oper`

Valid values are `DISABLE` or `ENABLE`.

`devid`

For AMC only: it is an on-Carrier device ID that identifies the on-Carrier device to which the desired channel is connected.

`-t ipmbAddr`

Sends the command to `ipmbAddr:mmcAddr`.

Example

```
hpmcmd -c portset base 1 0 base 0 0 enable
```

4.3.4.30 posttypeget

Description

This command retrieves the postType to which the board is currently set to run at boot time, for the specified CPU.

Synopsis

```
posttypeget <payload_cpu_selector>
```

Parameters

`payload_cpu_selector`

The specified CPU is set to postType to run. Is 0 for CPU.

Example

```
hpmcmd -c posttypeget 0
```

```
LONG
```

4.3.4.31 posttypeset

Description

This command sets the postType to which the board is currently set to run at boot time, for the specified CPU.

Synopsis

```
posttypeset <payload_cpu_selector> <newPostType>
```

Parameters

`payload_cpu_selector`

It is an integer between 0 and number of CPU devices supported per board.

`newPostType`

Valid values are: `SHORT | LONG`.

4.3.4.32 sdr

Description

This command shows the SDR records.

Synopsis

```
sdr
```

Example

```
hpmcmd -c sdr

recID 1: management controller device locator record
    I2C slave addr:  4D
    Channel number:  00
    Power state:     06
    Global init:     0C
    Capabilities:    2D
    Entity Id:       PICMG front board
    Entity instance: 60
    OEM:             00
    Id string:       ATCA-9405

recID 2: full sensor record
    owner is IPMB 9A sensor num 00 on lun 00 channel 00
    logical entity: PICMG front board - instance 60
    Hot Swap Carrier : FRU hot swap : sensor-specific discrete

recID 3: full sensor record
    owner is IPMB 9A sensor num 01 on lun 00 channel 00
    logical entity: RTM - instance 60
    HS_ARTM : FRU hot swap : sensor-specific discrete

.
.
.
.
recID 88: full sensor record
    owner is IPMB 9A sensor num 53 on lun 00 channel 00
    logical entity: RTM - instance 60
    Ver Change RTM : 2B : sensor-specific discrete
```

4.3.4.33 sdr_dump

Description

This command shows the SDR records in binary and hexadecimal format.

Synopsis

```
sdr_dump
```

Example

```
hpmcmd -c sdr_dump

SDR Records:

01 00 51 01 39 20 00 10 14 61 7f 69 02 01 04 22 "..Q.9 ...a.i..."" 
04 22 12 12 00 04 00 00 33 00 00 00 00 c0 07 cd "."......3......Í" 
d0 ca ff 00 00 d8 00 00 c2 00 01 01 00 00 00 ce "..............Î" 
53 42 43 20 2b 31 2e 30 35 56 20 56 74 74 "SBC +1.05V Vtt"

.
.
.
61 67 65 20 45 "age E"
```

4.3.4.34 sdrinfo

Description

This command shows the SDR information.

Synopsis

```
sdrinfo
```

Example

```
hpmcmd -c sdrinfo
```

```
LUN 0 has 084 sensors; dynamic sensor population
LUN 1 has 000 sensors
LUN 2 has 000 sensors
LUN 3 has 000 sensors
```

4.3.4.35 sel

Description

Show the SEL records

Synopsis

`Sel`

Example

`hpmcmd -c sel`

```
0x0023: Event: at: Sep 21 14:19:12 2011; from:(0x8c,0,0);
sensor:(0xda,31);
event:(0x6f,asserted): a0 00 01
0x0024: Event: at: Sep 21 14:19:12 2011; from:(0x8c,0,0);
sensor:(0xda,31);
event:(0x6f,asserted): a0 81 00
0x0025: Event: at: Sep 21 14:19:12 2011; from:(0x8c,0,0);
sensor:(0xda,31);
event:(0x6f,asserted): a0 00 01
0x0026: Event: at: Sep 21 14:19:12 2011; from:(0x8c,0,0);
sensor:(0xda,31);
event:(0x6f,asserted): a0 00 01
0x0027: Event: at: Sep 21 14:19:12 2011; from:(0x8c,0,0);
sensor:(0xda,31);
event:(0x6f,asserted): a0 41 00
0x0028: Event: at: Sep 21 14:19:12 2011; from:(0x8c,0,0);
sensor:(0xda,31);
event:(0x6f,asserted): a0 81 00
0x0029: Event: at: Sep 21 14:19:12 2011; from:(0x8c,0,0);
sensor:(0xda,31);
```

```
event:(0x6f,asserted): a0 18 00
0x002A: Event: at: Sep 21 14:19:13 2011; from:(0x8c,0,0);
sensor:(0xda,31);
event:(0x6f,asserted): a0 81 00
```

4.3.4.36 selclear

Description

Erases all contents of the SEL.

Synopsis

`Selclear`

Example

`hpmcmd -c selclear`

4.3.4.37 selinfo

Description

Show the SEL information.

Synopsis

`Sel`

Example

`hpmcmd -c selinfo`

```
SEL version: 1.5
Number of log entries: 1023
Free space: 0 bytes
Events have been dropped due to lack of space in the SEL
Last addition timestamp: Nov 15 08:40:40 2011
Last erase timestamp: Jan  1 00:00:00 1970
```

4.3.4.38 sendamc

Description

This command allows to send any of the commands supported in the IPMI specifications to a remote AMC or MMC of a remote IPMC IPMB-L.

Synopsis

```
sendamc <IPMBaddress> <MMCaddress> <netfn> <cmd> <data0> ...
<datan>
```

Parameters

`IPMBaddress`

Destination IPMB address in hex digits.

`MMCaddress`

Destination MMC address in hex digits.

`netfn`

IPMI request net function in hex digits.

`cmd`

IPMI request command in hex digits.

`data0-datan`

IPMI request data bytes, if any; in hex digits.

4.3.4.39 sendcmd

Description

This command allows a user to send any of the commands supported in the IPMI specifications to a remote IPMC.

Synopsis

```
sendcmd <IPMBaddress> <netfn> <cmd> <data0> ... <dataN>
```

Parameters

`IPMBaddress`

Destination IPMB address in hex digits.

`netfn`

IPMI request net function in hex digits.

`cmd`

IPMI request command in hex digits

`data0 ... dataN`

IPMI request data bytes, if any; in hex digits.

Example

```
hpmcmd -c sendcmd 90 06 59
07 59 C1
```

4.3.4.40 serialoutputget

Description

Determines which serial output source goes to a particular serial port connector.

Synopsis

```
serialoutputget <Serial Connector Type> <Instance Number>
```

Parameters

`Serial Connector Type` -

0 Face Plate Connector
1 Backplane Connector
2 Onboard Connector
3 Onboard Device (route to another chipset)

`Instance Number` -

zero-based instance number

Example

```
hpmcmd -c serialoutputget 0 1

Serial Output Selector: 1
```

4.3.4.41 serialoutputset

Description

Select the serial port output source for a serial port connector.

Synopsis

```
serialoutputset <Serial Connector Type> <Instance Number> <Serial
Output Selector>
```

Parameters

`Serial Connector Type` -

0 Face Plate Connector
1 Backplane Connector
2 Onboard Connector
3 Onboard Device (route to another chipset)

`Instance Number` -

zero-based instance number

`Serial Output Selector` -

an integer value >= 0

4.3.4.42 shelfaddress

Description

This command retrieves the shelf address string from the shelf FRU.

Synopsis

```
shelfaddress
```

Example

```
hpmcmd -c shelfaddress
01
```

4.3.4.43 shelfslots

Description

This command retrieves the total number of blade slots in the shelf.

Synopsis

```
shelfslots
```

Example

```
hpmcmd -c shelfslots

14 slots
```

4.3.4.44 shelftype

Description

This command retrieves the shelf FRU (IPMB 20) Board Area Product Name (FRU 254).

Synopsis

shelftype

Example

```
hpmcmd -c shelftype
```

```
CHS1406
```

4.3.4.45 slotmap

Description

This command prints a slotmap table for the shelf the blade is installed in.

Synopsis

```
slotmap
```

Example

```
hpmcmd -c slotmap

------------------------------------------------------------
Physical Slot: 01 02 03 04 . 05 06 07 08 09 10 . 11 12 13 14
Logical Slot: 01 03 05 07 . 09 11 13 04 06 08 . 10 12 14 02
IPMB Address: 82 86 8A 8E . 92 96 9A 88 8C 90 . 94 98 9C 84
------------------------------------------------------------
```

4.3.4.46 slotnumber

Description

This command retrieves the logical slot number of the slot where the blade is plugged in.

Synopsis

```
slotnumber Parameters
```

Example

```
hpmcmd -c slotnumber
2
```

4.3.4.47 solcfgget

Description

Get SOL configuration parameter

Synopsis

```
solcfgget <channel> [param]
```

Parameters

`channel` - channel number

`param` -

enable
authentication
char-settings
retry
nonvolatile-bit-rate
volatile-bit-rate
payload-channel
payload-port

Example

```
hpmcmd -c solcfgget 5
```

```
Enabled                          : false
Non-Volatile Bit Rate       (kps): 9.6
Volatile Bit Rate           (kps): 9.6
Payload Port                     : 623
```

4.3.4.48 solcfgset

Description

Set SOL configuration parameter

Synopsis

```
solcfgset <channel> <param> <value>
```

where:

`channel` - channel number

`param` -

force-encryption true | false
force-authentication true | false
privilege-level user | operator | administrator | oem

char-accumulate-interval 1-1275 (ms)
char-send-threshold 0-255
retry-count 0-255
retry-interval 0-2550 (ms)
non-volatile-bitrate 9.6 | 19.2 | 38.4 | 57.6 | 115.2
volatile-bitrate 9.6 | 19.2 | 38.4 | 57.6 | 115.2
port 0-255

4.3.4.49 version

Description

This command prints the version of the hpmcmd software.

Synopsis

```
version
```

Example

```
hpmcmd -c version
3.3.1
```

4.3.4.50 watchdog

Description

This command is used to handle the payload BMC watchdog.

Synopsis

```
watchdog set <tmr_use> <tmr_action> <pre_timeout> <flags> <lsb_val>
<msb_val>
watchdog set default
```

```
watchdog get
watchdog start
watchdog stop
watchdog reset
```

Parameters

`set`

Possible values are

Value	Description
tmr_use	dont_stop stop
tmr_action	no_action hard_reset power_cycle power_down
pre_timeout	0-255
flags	clear dont_clear
lsb_val	0-255
msb_val	0-255

The Network Management Utils (NMU) provides the tool `nmucmd` that can be used for supervision and control of transceivers. It is based on the NMU API which is part of the BLSV API. In the NMU world, a transceiver is part of the hierarchical organization among switches and ports. A board may have multiple switches and a switch may have multiple ports. A transceiver is always connected to a port.

On ATCA-9405,the NMU manages the transceivers connected on the ARTM-9405. The supported transceiver types are SFP, SFP+ and QSFP devices. Most transceiver information is retrieved from the EEPROM of the transceiver. Therefore the SFP and SFP+ transceiver must be compliant to the SFF-8472 and the QFTP transceiver to the SFF-8436 specification.

The QSFP transceivers are split up to 4 ports since some information are channel specific. If the information is not channel specific then it is replicated to all 4 ports.

The following table shows the mapping of the ARTM-9405 port number and NMU port numbers:

Table 5-1 NMU porting with ARTM-9405 port numbers

NMU port numbers	ARTM-9405 port numbers	Supported transceivers
1	SFP+ 1	SFP or SFP+
2	SFP+ 2	SFP or SFP+
3	SFP+ 3	SFP or SFP+
4	SFP+ 4	SFP or SFP+
5	SFP+ 5	SFP or SFP+
6	SFP+ 6	SFP or SFP+
7	SFP+ 7	SFP or SFP+
8	SFP+ 8	SFP or SFP+
9	QSFP 1	QSFP (Channel 1)
10		QSFP (Channel 2)
11		QSFP (Channel 3)
12		QSFP (Channel 4)
13	QSFP 2	QSFP (Channel 1)
14		QSFP (Channel 2)

Table 5-1 NMU porting with ARTM-9405 port numbers

NMU port numbers	ARTM-9405 port numbers	Supported transceivers
15		QSFP (Channel 3)
16		QSFP (Channel 4)

The nmucmd tool has the following usage:

```
root@ATCA-9405-12-13:/root> nmucmd -h
USAGE:
    nmucmd [options]

    -c       process a single command
    --help
    -h       Displays this help message
    -o       print results to a file
```

It supports the following commands:

```
root@ATCA-9405-12-13:/root> nmucmd -c help
-----------------------------------------------------------------
         Command  -  Description
-----------------------------------------------------------------
            help  -  list of commands.
            show  -  Prints information about different network
entities.
             trx  -  Modify transceiver settings
         version  -  Shows the version of this tool.
```

5.1 Interactive Mode

The nmucmd tool also has an interactive mode when it is called without any command option. In addition to the commands listed above, the following commands are supported in interactive mode:

Command	Description
quit exit bye	Exit the nmucmd interactive mode
rescan	Rescan the NMU device tree

NOTICE

If a transceiver is added or removed, the rescan command is necessary to update the NMU device tree. It internally re-initializes the NMU library. Otherwise the old device tree is displayed.

5.2 show command

The show command displays information about different network entities. On ATCA-9405, it can be used to display switch and transceiver information.

The full usage for the show command can be retrieved with the following command:

```
nmucmd -c help show
```

5.2.1 Show Switch Information

The following command provides information about all switches on the board. On ATCA-9405 there is one switch device. Its NMU name is `lion`.

```
root@ATCA-9405-12-13:/root> nmucmd -c show switch
lion: Marvell 98CX8234
```

5.2.2 Show Transceiver Information

The syntax to get information about transceiver(s) on the ARTM-9405 is as follows:

```
nmucmd -c show trx [ [switch] [port] ] [property]
```

The following example shows all information about all installed transceivers.

```
root@ATCA-9405-12-13:/root> nmucmd -c show trx
Transceivers of switch lion:
  Transceiver info at port 2:
    vendor-name          : AVAGO
    vendor-part-number   : AFCT-5715PZ
    vendor-revision      : 0000
    vendor-serial-number : AC1001S02UY
    vendor-date          : 01/05/10
    connector-type       : LC
    ethernet-code        : 1000BASE-LX
    encoding             : 8B/10B Line Code
    bitrate              : 1200 MBd
    tx-state             : Enabled
    rx-state             : Lost
    tx-fault             : Ok
    internal-temperature : 43.714844 degrees Celsius
    internal-supply-voltage : 3293300 uV
    laser-bias-current      : 12810 uA
    capability-tx-fault        : supported
    capability-soft-tx-fault   : supported
    capability-tx-disable      : supported
    capability-soft-tx-disable : supported
    capability-rx-los          : supported
    capability-soft-rx-los     : supported
  Transceiver info at port 4:
    vendor-name          : FINISAR CORP.
    vendor-part-number   : FTLX8571D3BCL
    vendor-revision      : A
    vendor-serial-number : ANE02XB
    vendor-date          : 10/01/12
    connector-type       : LC
    ethernet-code        : 10G BASE-SR
```

```
    encoding             : 64b/66b Line Code
    bitrate              : 10300 MBd
    tx-state             : Enabled
    rx-state             : Lost
    tx-fault             : Ok
    internal-temperature : 37.335938 degrees Celsius
    internal-supply-voltage : 3324200 uV
    laser-bias-current      : 8046 uA
    capability-tx-fault        : supported
    capability-soft-tx-fault   : supported
    capability-tx-disable      : supported
    capability-soft-tx-disable : supported
    capability-rx-los          : supported
    capability-soft-rx-los     : supported
```

An example to show only show the connector type of the SFP+ transceiver in port 2 is as follows:

```
root@ATCA-9405-12-13:/root> nmucmd -c show trx lion 2 connector-
type
    connector-type       : LC
```

5.3 trx command

The trx command can be used to modify transceiver settings. On the ATCA-9405, the command is no longer supported because it may lead to inconsistencies with the switch software (example, SRS). Enabling or disabling the laser is implicitly done when using the SRS shutdown command.

6.1 WindRiver Linux Layer

The WindRiver Linux build system supports layers and templates. A pre-configured layer is provided for the ATCA-9405 P2020 service processor. You can use this layer and build its own kernel, device-tree-blob, initial ramdisk, root filesystem or sysroot environment.

The `atca9405_p2020-layer.tgz` is delivered.

Following is the filesystem structure of the ATCA-9405 WindRiver Linux 4 layer with the files and directories:

```
+-atca9405/
  |
  +-Makefile
  +-env-p2020-aas114.sh
  +-version-kernel
  +-version-rootfs
  |
  +-RPMS/
  +-dist/
  +-templates/
    |
    +-feature/
    | |
    | +-atca-9405-p2020-kernel/
    |   |
    |   +-linux/
    |     |
    |     +-fsl_p2020-addons/
    |     | +-000x-...patch
    |     |
    |     +-000x-...patch
    |     +-enable-export_layer.cfg
    |     +-enable-export_layer.scc
    |
    +-board/
      |
      +-atca9405_p2020/
        |
```

```
+-config.sh
+-include
|
+-linux/
| +-atca9405_p2020-standard.scc
|
+-rootfs/
  |
  +-glibc_std/
    |
    +-fs_final.sh
    +-include
    +-pkglist.add
    +-pklist.remove
    |
    +-fs/
      |
      +-etc/
      +-opt/
      +-root/
      +-usr/
```

The following table lists all pre-build files out of the ATCA-9405 P2020 layer that are delivered with all releases:

Table 6-1 WindRiver 4 Layer Files

Filename	Description
`atca9405_p2020-System.map-WR4.1.0.0_standard_<version>`	Linux kernel system map
`atca9405_p2020-default_kernel_image-WR4.1.0.0_standard_<version>`	Linux kernel image for u-boot (image includes u-boot header)
`atca9405_p2020-linux-modules-WR4.1.0.0_standard_<version>.tar.bz2`	Linux kernel modules tarball (/lib/modules directory)
`atca9405_p2020-p2020rdb.dtb-WR4.1.0.0_standard_<version>`	Device-tree-blob for u-boot

Table 6-1 WindRiver 4 Layer Files

Filename	Description
`atca9405_p2020-initrd-<version>.gz.uboot`	Linux initial ramdisk image (image includes u-boot header)
`atca9405_p2020-standard-glibc_std-dist-<version>.tar.bz2`	Linux root filesystem tarball
`atca9405_p2020-sysroot.tgz`	WindRiver Linux toolchain to build applications for ATCA-9405 P2020

The files listed in the table above are built with the following configuration:

```
Board name:      atca9405_p2020
Kernel type:     standard
Filesystem type: glibc_std
Build type:      production
```

Building requires an installed WindRiver Linux 4.1 including the powerpc toolchain. The provided top-level Makefile in the layer can be used to generate the files listed above.

Its default rule `all` calls the WindRiver Linux configure script, enters the generated build directory and calls a second make process to build everything. The Makefile expects the WIND_HOME and BUILDDIR environment setting. The env-p2020-<user>.sh bash shell script shows an example of these settings.

6.1.1 Application Development

For ATCA-9405 P2020 application and library development the provided or a self-generated toolchain should be used. The provided top-level Makefile in the layer contains the rule 'sysroot' to generate a sysroot tarball (`atca9405_p2020-sysroot.tgz`).

The tarball can be extracted anywhere on a development machine with access to an installed WindRiver Linux 4.1 including powerpc toolchain.

NOTICE

The sysroot directory does not include a standalone toolchain.

The following directories should be included in the PATH environment in order to use the correct ATCA-9405 P2020 development toolchain:

- `$WIND_HOME/wrlinux-4/layers/ldat-tools/host-tools/bin`
- `$WIND_HOME/wrlinux-4/layers/wrll-toolchain-4.4a-323/powerpc/toolchain/x86-linux2/bin`
- `<extracted_rootdir>/sysroot/atca9405_p2020-glibc_std/x86-linux2`

The prefix for the toolchain binaries (e.g. compiler, linker...) is: `powerpc-wrs-linux-gnu-ppc_e500v2-glibc_std-`

As an example the GNU C-Compiler is: `powerpc-wrs-linux-gnu-ppc_e500v2-glibc_std-gcc`

6.2 Octeon SDK Development

All provided software components for the Octeon processor are using Caviums Octeon-SDK.

You can download Octeon-SDK from the Cavium support site: https://support.cavium.com/.

This chapter gives a quick overview about how to build software for the Octeon processor. For more information, refer *Octeon-SDK Documentation*.

Artesyn provides a patch for a specific Octeon-SDK version to work on the ATCA-9405 hardware.

The following software components are currently delivered:

Table 6-2 Octeon software components

Component Type	Filename
Bootloader (u-boot)	`u-boot-octeon_atca9405-<version>.bin`
Linux including initramfs	`atca9405_cn68xx-vmlinux_initramfs-<version>-1.ppc_e500v2-wrspne4-linux.rpm`
traffic-gen SE-S example	`traffic-gen`
passthrough SE-S example	`passthrough`
ATCA-9405 Octeon-SDK patch	`atca9405-octeon-sdk-2.3.0-<version>.tar.gz`

6.2.1 ATCA-9405 Octeon-SDK Patch

The ATCA-9405 Octeon-SDK patch modifies Octeon-SDK files to be able to build software for the ATCA-9405. A README file within the delivered tarball describes the changes as well as the installation.

Note that the Octeon-SDK patch works only for a specific Octeon-SDK version. The patch filename as well as the README file provides you the details about which version of the Octeon-SDK is required.

For example, the ATCA-9405 Octeon-SDK patch, `atca9405-octeon-sdk-2.3.0-497-2.2.8.tar.gz` works only for Octeon-SDK 2.3.0 with Cavium patch, `sdk_2.3.0_update_p9` applied.

The following steps are recommended to use Octeon-SDK for the ATCA-9405:

1. Download and install the Octeon-SDK and Octeon-Linux package from Cavium support site.
2. Apply the Octeon-SDK patch from Cavium, when required.
3. Make a copy of the original unmodified Octeon-SDK directory on your local directory.
4. Change to your local Octeon-SDK directory.
5. Source the Octeon-SDK environment file:
   ```
   . env-setup OCTEON_CN68XX
   ```
6. Extract the ATCA-9405 Octeon-SDK patch.
7. Change to the directory of the extracted patch.
8. Apply the patch:
   ```
   make
   ```

Now you are ready to build u-boot, Linux or any SE application along with the Octeon-SDK . In case if you have left your current shell and want to use the Octeon-SDK again, make sure you source the Octeon-SDK environment file:

```
. env-setup OCTEON_CN68XX
```

6.2.2 Octeon U-boot

The following steps build the Octeon bootloader u-boot for the ATCA-9405 blade. Before performing these steps, make sure you have applied the ATCA-9405 Octeon-SDK patch.

1. Change to your local Octeon-SDK directory.
2. Source the Octeon-SDK environment file:
   ```
   . env-setup OCTEON_CN68XX
   ```
3. Change to the u-boot directory:
   ```
   cd bootloader/u-boot
   ```
4. Configure u-boot
   ```
   make octeon_atca9405_config
   ```
5. Build u-boot
   ```
   make
   ```

The resulting u-boot image to be used is:
`$OCTEON_ROOT/bootloader/u-boot/ u-boot-octeon_atca9405.bin`

6.2.3 Octeon Linux

The additional Octeon package Octeon-Linux from Cavium contains a version 2.6 based Linux kernel including a busybox based initramfs. The kernel also contains the Cavium Ethernet driver to use the RXAUI and DXAUI network interfaces as regular Linux network interfaces.

The following steps build the Octeon Linux including the initramfs for the ATCA-9405 blade. Before performing these steps, make sure you have applied the ATCA-9405 Octeon-SDK patch.

1. Change to your local Octeon-SDK directory.
2. Source the Octeon-SDK environment file:
   ```
   . env-setup OCTEON_CN68XX
   ```
3. Change to the Linux directory:
   ```
   cd linux
   ```
4. Build the Linux kernel with the initramfs
   ```
   make kernel
   ```

The resulting image to be used is:
`$OCTEON_ROOT/linux/kernel_2.6/linux/vmlinux`

6.2.4 Octeon Simple Executive Applications

For building an SE application (SE-S or SE-UM), the files in the `$OCTEON_ROOT/executive` directory are required. You can refer to the Makefiles in the `$OCTEON_ROOT/examples` directory as a starting point to a new application Makefile.

The following steps build the SE-S (64bit) example application passthrough:

1. Change to your local Octeon-SDK directory.
2. Source the Octeon-SDK environment file:
   ```
   . env-setup OCTEON_CN68XX
   ```
3. Change to the passthrough directory:
   ```
   cd examples/passthrough
   ```
4. Build SE-S application
   ```
   make
   ```

The resulting image to be used is:
`$OCTEON_ROOT/examples/passthrough/passthrough`

7.1 Overview

The Switching and Routing stackware (SRstackware) is a protocol suite that is installed as a part of blade services installation on the ATCA blade. SRstackware provides features to perform ATCA-9405 switch configuration and check the statistics in addition to the protocol stack support. SRstackware supports the following features and protocols:

- IEEE 802.1p and IEEE 802.1q VLAN and bridge configuration, IEEE 802.1d - Spanning Tree Protocol (STP)
- IEEE 802.1w - Rapid Spanning Tree Protocol (RSTP)
- IEEE 802.1s - Multiple Spanning Tree Protocol (MSTP)
- IEEE 802.3ad Link Aggregation Control Protocol (LACP)
- IGMP Snooping, IGMPv3 Protocol (IGMP)
- Open Shortest Path First version 2 (OSPFv2)
- Static routing
- IEEE 802.1q GVRP, GMRP, GARP protocols
- Q-in-Q or VLAN Stacking
- Virtual Router Redundancy Protocol (VRRP)
- Router Information Protocol (RIPv2)
- Router Information Protocol next generation (RIPng)
- SNMP support

Refer Appendix A, *Related Documentation, on page 155*, for the documents and sections to be referred for using the protocols and features defined in the current SRstackware release.

7.2 Configuring SRstackware

SRstackware, after installation, comes up with the default configuration. You can update the configuration using CLI.

7.2.1 Default Configuration

The ATCA-9405 default configuration creates one bridge and multiple VLANs. All the VLAN ports are associated to the bridge. You can configure additional logical bridges and VLANs, if required. The switch ports are named as `ge#` for the 1G ports and `xe#` for the 10G/40G ports. These switch ports are accessible using the ATCA-9405 CLI.

You can modify the default configuration using the ATCA-9405 CLI. The default configuration file is saved at `/etc/opt/srstackware/config/srs.cfg`. This file is persistent across reboots except if ramdisk boot is used and SRstackware picks up this file while bootup to restore its configuration. If the administrator issues `write file` command through imish it saves the configuration to this file. It is advised to save a copy of the original configuration file for a later use.

ATCA 9405 allows ports to be configured to a specific independent or group mode. This configuration is relevant for Fabric Channel ports and RTM uplink ports of the switch. The speed/bandwidth for the configured port is implicitly set based on the mode selected.

The fabric channel and QSFP interfaces can be operated in either grouped or independent mode:

- In grouped mode, the interface appears as one switch port. Grouped modes are “KX4”, “KR4” and “QSFP”.
- In independent mode, the interface appears as four ports at the switch. Independent modes are “KX”, “KR”, “SFP” and “SFP+”.

For more information, refer *SRstackware Intelligent Network Software Switch Configuration Command Reference Guide*.

By default, ATCA-9405 does not enable any protocol on the ports.

The following table lists the switch ports and associated VLANs.

SRS Port	Interface description	Description	VLANs assigned	Default Enabled
ge1	SGMII	Service Processor (SP) ETH1	11(t), 21(t), 32(t), 33(t)	YES
ge2	SGMII	PP2_ETH0 - 1G Management	21(u), 22(u), 33(u), 34(u)	YES
ge3	SGMII	PP1_ETH0 - 1G Management	21(u), 22(u), 33(u), 34(u)	YES
ge4	SGMII	Front Panel Ethernet (PP MGMT)	34(u)	YES
ge5	SGMII	TERMSRV	21(u), 22(u), 32(u)	YES
ge6	SGMII	SP_ETH2	12(t), 22(t), 32(t)	YES
ge7	SGMII	BASE1	21(u), 32(u)	YES
ge8	SGMII	BASE2	22(u), 32(u)	YES
xe1	XGMII	Packet Processor 2 XAUI0 - 20G (PP2)	11(u), 31(u)	YES
xe2	XGMII	PP2_XAUI1 - 20G	12(u), 31(u)	YES
xe3	RXAUI	PP2_RXAUI0 - 10G	33(u)	YES
xe4	RXAUI	PP2_RXAUI1 - 10G	33(u)	YES
xe5	NA	Fabric Channel 2 - port 2 (Inactive in Group Mode)	12(u)	NA
xe6	NA	Fabric Channel 2 - port 3 (Inactive in Group Mode)	12(u)	NA
xe7	XLG	Fabric Channel 2 - port 1 - 40G (Active in Group Mode)	12(u)	NO

SRS Port	Interface description	Description	VLANs assigned	Default Enabled
xe8	NA	Fabric Channel 2 - port 4 (Inactive in Group Mode)	12(u)	NA
xe9	XGMII	Packet Processor 1 XAUI1 - 20G	12(u), 31(u)	YES
xe10	XGMII	PP1_XAUI0 - 20G	11(u), 31(u)	YES
xe11	RXAUI	PP1_RXAUI0 - 10G	33(u)	YES
xe12	RXAUI	PP1_RXAUI1 - 10G	33(u)	YES
xe13	NA	Fabric Channel 1 - port 2 (Inactive in Group Mode)	11(u)	NA
xe14	NA	Fabric Channel 1 - port 3 (Inactive in Group Mode)	11(u)	NA
xe15	XLG	Fabric Channel 1 - port 1 - 40G (Active in Group Mode)	11(u)	NO
xe16	NA	Fabric Channel 1 - port 4 (Inactive in Group Mode)	11(u)	NA
xe17	XGMII	UC	31(u)	YES
xe18	NA	RTM QSFP 2 - port 3 (Inactive in Group Mode)	79(u)	NA
xe19	NA	RTM QSFP 2 - port 4 (Inactive in Group Mode)	80(u)	NA

SRS Port	Interface description	Description	VLANs assigned	Default Enabled
xe20	XLG	RTM QSFP 2 - port 1 40G (Inactive in Group Mode)	81(u)	NO
xe21	NA	RTM QSFP 2 - port 2 (Inactive in Group Mode)	82(u)	NA
xe22	RXAUI	RTM_SFPP_1	71(u)	NO
xe23	RXAUI	RTM_SFPP_2	72(u)	NO
xe24	RXAUI	RTM_SFPP_3	71(u)	NO
xe25	RXAUI	RTM_SFPP_4	71(u)	NO
xe26	RXAUI	RTM_SFPP_5	71(u)	NO
xe27	RXAUI	RTM_SFPP_6	71(u)	NO
xe28	RXAUI	RTM_SFPP_7	71(u)	NO
xe29	RXAUI	RTM_SFPP_8	71(u)	NO
xe30	NA	RTM QSFP 1 - port 1 (Inactive in Group Mode)	71(u)	NA
xe31	NA	RTM QSFP 1 - port 2 (Inactive in Group Mode)	71(u)	NA
xe32	XLG	RTM QSFP 1 - port 3 40G (Inactive in Group Mode)	71(u)	NO
xe33	NA	RTM QSFP 1 - port 4 (Inactive in Group Mode)	71(u)	NA

When a VLAN-aware bridge is created and assigned to a switch port interface; SR stackware adds the interface to the Linux kernel as a network interface. The interface is named as `vlanx.y`, where `x` specifies the bridge name and `y` specifies the VLAN id.

The following table lists the default VLANs and the port members:

VLAN Number	Usage	Members
11	Fabric Interface 1	xe1(u),xe10(u),xe13(u),xe14(u),xe15(u) and xe16(u),ge1(t)
12	Fabric Interface 2	xe2(u),xe5(u),xe6(u),xe7(u),xe8(u) and xe9(u),ge6(t)
21	Base Interface 1	ge1(t),ge2(u),ge3(u),ge5(u) and ge7(u)
22	Base Interface 2	ge2(u),ge3(u),ge5(u),ge6(t) and ge8(u)
31	Update Channel	xe1(u),xe2(u),xe9(u),xe10(u) and xe17(u)
32	Terminal server/serial redirection	ge1(t),ge5(u),ge6(t),ge7(u) and ge8(u)
33	Local Management	ge1(t),ge2(u),ge3(u),xe3(u),xe4(u),xe11(u) and xe12(u)
34	Packet Processor management interface	ge2(u),ge3(u) and ge4(u)
71 .. 78	RTM SFP+ uplinks	xe22/xe23/xe24/xe25/xe26/xe27/xe28/xe29 resp.
79 .. 82	QSFP2	xe18/xe19/xe20/xe21 resp.
83 .. 86	QSFP 1	xe30/xe31/xe32/xe33 resp.
u -- Untagged t -- Tagged		

7.2.2 Custom Configuration

You can customize the ATCA-9405 after starting the ATCA-9405 CLI using the following steps:

1. Login as `root` at the ATCA-9405.
2. Run `/opt/srstackware/bin/imish`.

A sample output of the above steps is shown below.

```
root@Slot-1_9405:/opt/srstackware/bin>./imish

ZebOS version 7.8.4 IPIRouter 01/08/13 01:56:29

Slot-1_9405>enable
```

```
Slot-1_9405#show interface
```

7.2.2.1 Enabling Protocols

You can enable protocols on switch port of ATCA-9405. By default, the protocols are disabled. For proper functionality of the bridge, you can enable any protocol and the respective interfaces, on the bridge.

In case a bridge is removed from an interface, all the default configurations associated to that interface are also removed. You need to enable the desired VLANs and protocols on those interfaces.

7.2.2.2 Sample Configurations

This section provides a quick reference to the sequence of CLI commands for enabling protocols on the RTM ports.

- To check if the link at ge1 interface is up or down:

```
root@Slot-1_9405:/opt/srstackware/bin>./imish

ZebOS version 7.8.4 IPIRouter 01/08/13 01:56:29

Slot-7_9405>en
Slot-7_9405#show interface ge1

Interface ge1

  Hardware is Ethernet, address is 00f7.2214.0003 (bia 00f7.2214.0003)

  Description: BC1

  index 5001 metric 1 mtu 28836 duplex-full arp ageing timeout 0

  <UP,BROADCAST,RUNNING,MULTICAST>

  VRF Binding: Not bound

  Bandwidth 1g

   input packets 012480, bytes 0799356, dropped 00, multicast packets
044515

    output packets 032106, bytes 02054932, multicast packets 032106
broadcast packets 066
```

In this sample output, RUNNING in the following line indicates that the interface link is up.

```
<UP,BROADCAST,RUNNING,MULTICAST>
```

In case the link is down the sample output contains the following line.

```
 <UP,BROADCAST,MULTICAST>
```

- To show all the interfaces (switch ports):
  `root@Slot-1_9405:/opt/srstackware/bin>`**`./imish`**

```
ZebOS version 7.8.4 IPIRouter 01/08/13 01:56:29

Slot-1_9405>en
Slot-1_9405#show interface
```

  You can use include and begin options with this command to modify the output and display it.

- To configure a bridge with STP enabled on it:
  `root@Slot-1_9405:/opt/srstackware/bin>`**`./imish`**

```
ZebOS version 7.8.4 IPIRouter 01/08/13 01:56:29

Slot-1_9405>en
Slot-1_9405#conf t

Enter configuration commands, one per line.  End with CNTL/Z.

Slot-1_9405(config)#bridge 1 protocol ieee
Slot-1_9405(config)#
```

- To configure a VLAN and associate it to a bridge:
  `root@Slot-1_9405:/opt/srstackware/bin>`**`./imish`**

```
ZebOS version 7.8.4 IPIRouter 01/08/13 01:56:29

Slot-1_9405>en
Slot-1_9405#conf t

Enter configuration commands, one per line.  End with CNTL/Z.

Slot-1_9405(config)#bridge 1 protocol ieee vlan-bridge
Slot-1_9405(config)#vlan database
```

```
Slot-1_9405(config-vlan)#vlan 23 bridge 1 name VLAN23 state
enable
Slot-1_9405(config-vlan)#
```

- To configure the VLAN 23 (mentioned in the above sample) to a switch port:

```
root@Slot-1_9405:/opt/srstackware/bin>./imish

ZebOS version 7.8.4 IPIRouter 01/08/13 01:56:29

Slot-1_9405>en
Slot-1_9405#
Slot-1_9405#conf t

Enter configuration commands, one per line.  End with CNTL/Z.

Slot-1_9405(config)#interface ge1
Slot-1_9405(config-if)#switchport mode access
Slot-1_9405(config-if)#switchport access vlan 23
```

- To enable STP on the RTM port 1:
  To configure an L2 protocol, such as STP, RSTP, or MSTP on a switch port:
  Bridge level configuration
  - Create a bridge, if required. By default, the bridges are created as a part of the default configuration. In case, the interface has VLAN configuration, tag it with `vlan-bridge` (as mentioned in the code below).
  - Enable the L2 protocol, such as STP, RSTP, or MSTP on the bridge.

  Interface level configuration
  - Associate the interface with the bridge, it automatically enables the corresponding protocol of the bridge.
  - Create the VLANs.

```
root@Slot-1_9405:/opt/srstackware/bin>./imish

ZebOS version 7.8.4 IPIRouter 01/08/13 01:56:29

Slot-1_9405>enable
Slot-1_9405#
Slot-1_9405#configure terminal

Enter configuration commands, one per line.  End with CNTL/Z.
```

```
/*** Associate STP to bridge-1 as part of Step (2) ***/
Slot-1_9405(config)#bridge 1 protocol ieee vlan-bridge

/* Go to interface mode */
Slot-1_9405(config)#interface xe22

/* Remove bridge so that STP can be enabled  */
Slot-1_9405(config-if)# no bridge-group 1

/* Enables STP on the interface as part of Step (3) */
Slot-1_9405(config-if)#bridge-group 1

/* Re-configure the necessary VLANs; because VLANs disappear if
bridge is removed */
Slot-1_9405(config-if)#switchport mode hybrid

/* As a part of Step (4) */
Slot-1_9405(config-if)#switchport hybrid vlan 93

Slot-1_9405(config-if)#switchport mode hybrid acceptable-frame-
type all
Slot-1_9405(config-if)#switchport hybrid allowed vlan add 93
egress-tagged disable
/* Enable the interface */
Slot-1_9405(config-if)#no shutdown

Slot-1_9405(config-if)#exit
```

- To enable RSTP on the base RTM port 1:

  Configuring RSTP is similar to configuring STP as mentioned in the above section, *To enable STP on the base RTM port 1*. To configure RSTP replace `ieee` with `rstp`, as in the following code.

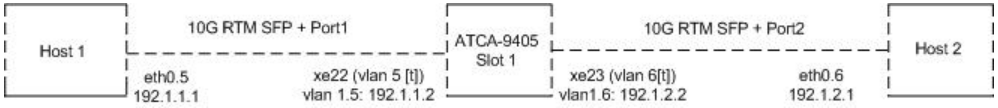
```
/* Associate RSTP to bridge-1 */
Slot-1_9405(config)# bridge 1 protocol rstp vlan-bridge
```

- To enable MSTP on the RTM port 2:

```
root@Slot-1_9405:/opt/srstackware/bin>./imish

ZebOS version 7.8.4 IPIRouter 01/08/13 01:56:29
```

```
Slot-1_9405>en
Slot-1_9405#
Slot-1_9405#conf t

Enter configuration commands, one per line.  End with CNTL/Z.

Slot-1_9405(config)#bridge 1 protocol mstp vlan-bridge
Slot-1_9405(config)#interface xe23
SLOT-1_9405(config-if)# no bridge-group 1
SLOT-1_9405(config-if)# bridge-group 1
Slot-1_9405(config-if)#no shutdown
Slot-1_9405(config-if)#
```

- To enable static routing through VLAN interfaces within the following topology.



On Host 1:

```
vconfig add eth0 5

ifconfig eth0.5 192.1.1.1/24

route add -net 192.1.2.0/24 gw 192.1.1.2 dev eth0.5
```

On ATCA-9405 Slot 1:
Create VLAN 5 and 6, with intervlan routing enabled, and associate it to Bridge 1.

```
/** If static routing through VLANs has to be done through ports of two
fabric chip sets, then these VLANs should be added on hi-gig using "vlan
<vlan-id> hi-gig". **/

Slot-1_9405>en

Slot-1_9405#conf t

Enter configuration commands, one per line.  End with CNTL/Z.

Slot-1_9405(config)#interface xe22
```

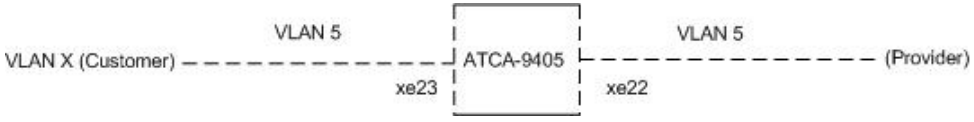
```
Slot-1_9405(config-if)#bridge-group 1 spanning-tree disable
Slot-1_9405(config-if)#switchport mode trunk
Slot-1_9405(config-if)#switchport trunk allowed vlan add 5
Slot-1_9405(config-if)#no shutdown
Slot-1_9405(config-if)#exit
Slot-1_9405(config)#interface vlan1.5
Slot-1_9405(config-if)#ip address 192.1.1.2/24
Slot-1_9405(config-if)#no shutdown
Slot-1_9405(config-if)#exit

Slot-1_9405(config)#interface xe23
Slot-1_9405(config-if)#bridge-group 1 spanning-tree disable
Slot-1_9405(config-if)#switchport mode trunk
Slot-1_9405(config-if)#switchport trunk allowed vlan add 6
Slot-1_9405(config-if)#no shutdown
Slot-1_9405(config-if)#exit
Slot-1_9405(config)#interface vlan1.6
Slot-1_9405(config-if)#ip address 192.1.2.2/24
Slot-1_9405(config-if)#no shutdown
Slot-1_9405(config-if)#exit
Slot-1_9405(config)#ip forwarding
```

On Host 2:

```
vconfig add eth0 6
ifconfig eth0.6 192.1.2.1/24
```

```
route add -net 192.1.1.0/24 gw 192.1.2.2 dev eth0.6
```

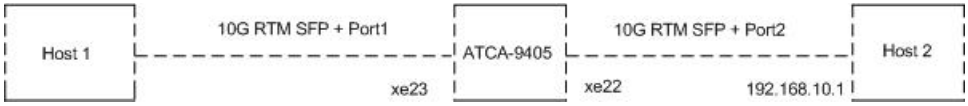
- To enable Q-in-Q (VLAN stacking) on the ATCA-9405 with the following topology.




```
Slot-1_9405>en
Slot-1_9405#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Slot-1_9405(config)#bridge 1 protocol ieee vlan-bridge
Slot-1_9405(config)#vlan database
Slot-1_9405(config-vlan)#vlan 5 bridge 1 state enable
Slot-1_9405(config-vlan)#exit

Slot-1_9405(config)#interface xe22
Slot-1_9405(config-if)#no shutdown
Slot-1_9405(config-if)#bridge-group 1
Slot-1_9405(config-if)#switchport mode access
Slot-1_9405(config-if)#switchport vlan-stacking customer-edge-port
Slot-1_9405(config-if)#switchport access vlan 5
Slot-1_9405(config-if)#exit

Slot-1_9405(config)#interface xe23
Slot-1_9405(config-if)#no shutdown
Slot-1_9405(config-if)#bridge-group 1
```

```
Slot-1_9405(config-if)#switchport mode trunk
Slot-1_9405(config-if)#switchport mode trunk allowed vlan add 5
Slot-1_9405(config-if)#switchport vlan-stacking provider-port
Slot-1_9405(config-if)#exit
```

- To enable VLAN Classification based on Source IP network with the following topology.




```
Slot-1_9405>en
Slot-1_9405#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Slot-1_9405(config)#bridge 1 protocol ieee vlan-bridge
Slot-1_9405(config)#vlan database
Slot-1_9405(config-vlan)#vlan 5 bridge 1 state enable
Slot-1_9405(config-vlan)#exit

Slot-1_9405(config)# vlan classifier rule 2 ipv4 192.168.10.0/24 vlan 5
Slot-1_9405(config)# vlan classifier group 10 add rule 2

Slot-1_9405(config)#interface xe22
Slot-1_9405(config-if)#bridge-group 2
Slot-1_9405(config-if)#switchport mode trunk
Slot-1_9405(config-if)#switchport trunk allowed vlan add 5
```

```
Slot-1_9405(config-if)# vlan classifier rule 2 ipv4 192.168.10.0/24
vlan 5

Slot-1_9405(config-if)#no shutdown

Slot-1_9405(config-if)#exit
```

7.2.3 SNMP Usage Guidelines

For Bridge-based MIBs, the context parameter of the snmp commands (`snmpget/snmpset/snmpwalk`) is used to specify the `bridge_number` and the details of the corresponding bridge are processed.

The string Bridge to be specified with the context parameter is case-sensitive.

The exact string to be used in the command can be obtained by doing a `snmpwalk` on the object "`srsBridgeName`" which is a proprietary mib-object provided by ATCA-9405.

In case of 9405, by default only Bridge1 is created.

Example of a `snmp` command:

```
snmpget -v3 -u admin -n "Bridge1" -l noAuthNoPriv -a MD5 -A adminpwd123
localhost dot1dBaseBridgeAddress.0
```

Here, the option 'n' is used to specify the context. The above command will return the MAC address used by this bridge when it must be referred to in a unique fashion.

For the mibs that do not distinguish based on the bridge, needs to be run with null context (that is -n "") or the context option can be entirely removed from the `snmp` command.

The following are the mibs, among the currently supported mibs for which context needs to be specified as `Bridge_number`:

- BRIDGE MIB
- P-BRIDGE MIB

- Q-BRIDGE MIB
- RSTP MIB

The following are the mibs, currently supported, that do not need any context to be specified:

- OSPF MIB
- RIP MIB
- IF MIB
- IEEE8023-LAG-MIB
- SRS MIB (Artesyn Proprietory MIB)

7.2.4 SNMP Traps Usage Guidelines

SNMP traps and SNMP Inform requests are the two SNMP notifications supported with the SRstackware package. The following sections detail the configurations to be made at SNMP agent and manager in order to facilitate these notifications.

7.2.4.1 Configuring Agent to Send SNMP Notifications

In order to configure SRstackware to send SNMP traps/Inform requests, a shell command has been provided as part of SRstackware package.

This shell command has the following syntax:

```
#srs_trap_config <IPv4 Address> <NONE/TRAP/INFORM>
```

`NONE`: Neither SNMP traps nor Inform requests are sent to the SNMP manager running on machine with IPv4 Address.

`TRAP`: Trap messages are sent to SNMP manager running on machine with IPv4 Address.

`INFORM`: SNMP Inform requests are sent to SNMP manager running on machine with IPv4 Address.

Example usage of the command:

In order to send traps to a system with IP address 10.10.10.10, run the command at the agent as follows:

```
#srs_trap_config 10.10.10.10 TRAP
```

The keywords NONE/TRAP/INFORM are case sensitive.

7.2.4.2 Configuring SNMP Manager to Receive SNMP Notifications

The snmptrapd daemon needs to be run at the receiver to receive the SNMP notifications sent by the agent. snmptrapd needs a configuration file to be supplied with proper configuration in order to receive the traps properly. snmptrapd configuration file must have the following two lines:

```
createUser -e <Agent's Engine-ID> admin MD5 adminpwd123 DES
```

```
authUser log,execute admin
```

`Agent's Engine-ID`: The Engine-ID with which SNMP Agent is running.

To obtain the Agent's Engine-ID, issue the following command:

```
snmpget -v3 -u admin -A adminpwd123 <Agent's IP address>
snmpEngineID.0
```

The result of this command will be in Hex-format and needs to be passed as a single string in the configuration file.

Example:

```
snmpget -v3 -u admin -A adminpwd123 10.10.10.20 snmpEngineID.0
```

Output:

```
SNMP-FRAMEWORK-MIB::snmpEngineID.0 = Hex-STRING: 80 00 1F 88 04 65
6D 65 72 73 6F 6E
```

Now, snmptrapd configuration file will look as follows:

```
createUser -e 0x80001F8804656D6572736F6E admin MD5 adminpwd123 DES
```

```
authUser log,execute admin
```

snmptrapd may be run at the receiver (SNMP manager) as follows (the formatting options can be modified as you desire, refer the man pages of snmptrapd for details):

```
/usr/sbin/snmptrapd -x tcp:localhost:705 -c
"<snmptrapd_configuration_file>" -C -f -Le
```

7.3 Standards Supported

Table 7-1 Standards Supported

Protocol	Standard
STP	802.1d
RSTP	802.1w
MSTP	802.1s
GVRP	802.1Q
GMRP	802.1Q
IGMP Snooping	RFC 4541
LACP	802.3ad
OSPFv2	RFC 2328
OSPF NSSA Option	RFC 3101
OSPF Graceful Restart	RFC 3623
OSPF Restart Signaling	draft-nguyen-ospf-restart-05
OSPF Opaque LSA Option	RFC 2370
OSPF Alternative Implementation of OSPF ABR	RFC 3509
IGMPv3	RFC 3376 and draft-ietf-magma-igmp-proxy-06
RIP	RFC 2453

Table 7-1 Standards Supported (continued)

Protocol	Standard
RIPng	RFC 2080
VRRP	RFC 3768
SNMP MIBS	
Bridge-MIB	RFC 4188
P-Bridge-MIB	RFC 4363
Q-Bridge-MIB	RFC 4363
IF-MIB	RFC 2863
OSPF-MIB	RFC 1850
IEEE8023-LAG-MIB	IEEE 802.3ad MIB
RSTP-MIB	RFC 4318
RIP MIB	RFC 1724
VRRP MIB	draft-ietf-vrrp-unified-mib-06

7.4 2x40G/8x10G RTM Ports

Table 7-2 ARTM-9405 2x40G/8x10G

Description	Marvell Port No# (only for Debugging purpose)	SRS Port	Port Group
Packet Processor 2 (PP2) XAUI0 - 20G	0	xe1	1
PP2_XAUI1 - 20G	2	xe2	1
Service Processor (SP) ETH1	4	ge1	1
PP2_ETH0 - 1G Management Interface	5	ge2	1
PP2_RXAUI0 - 10G	6	xe3	1
PP2_RXAUI1 - 10G	7	xe4	1

Table 7-2 ARTM-9405 2x40G/8x10G (continued)

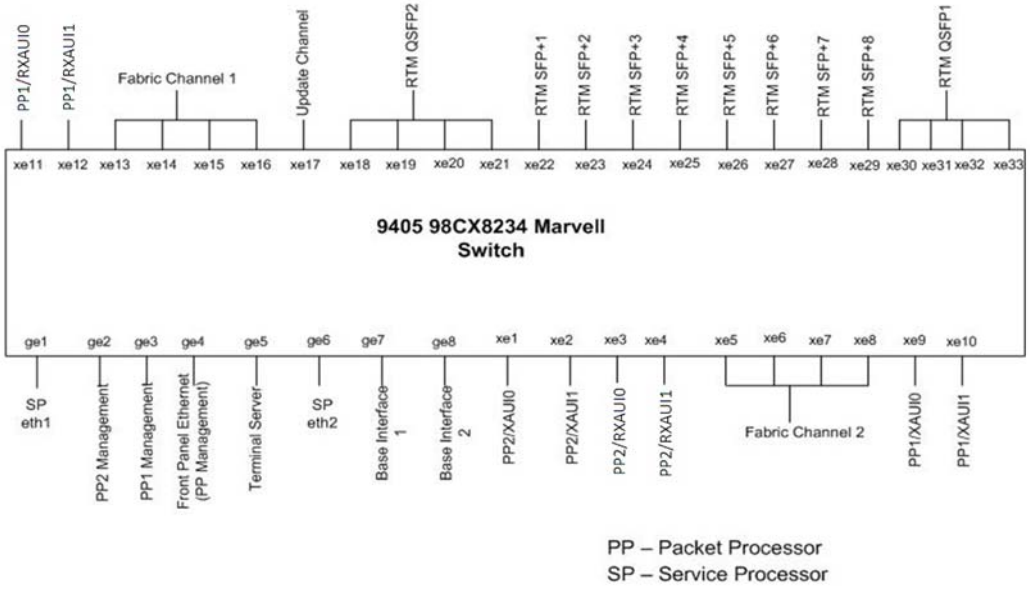
Description	Marvell Port No# (only for Debugging purpose)	SRS Port	Port Group
Fabric Channel 2 - port 2 (Inactive in Group Mode)	8	xe5	1
Fabric Channel 2 - port 3 (Inactive in Group Mode)	9	xe6	1
Fabric Channel 2 - port 1 - 40G (Active in Group Mode)	10	xe7	1
Fabric Channel 2 - port 4 (Inactive in Group Mode)	11	xe8	1
Packet Processor 1 (PP1) XAUI1 - 20G	16	xe9	2
PP1_XAUI0 - 20G	18	xe10	2
PP1_ETH0 - 1G Management Interface	20	ge3	2
PP_MGMT	21	ge4	2
PP1_RXAUI0 - 10G	22	xe11	2
PP1_RXAUI1 - 10G	23	xe12	2
Fabric Channel 1 - port 2 (Inactive in Group Mode)	24	xe13	2
Fabric Channel 1 - port 3 (Inactive in Group Mode)	25	xe14	2
Fabric Channel 1 - port 1 - 40G (Active in Group Mode)	26	xe15	2
Fabric Channel 1 - port 4 (Inactive in Group Mode)	27	xe16	2
TERMSRV	32	ge5	3
SP_ETH2	33	ge6	3
BASE1	34	ge7	3
BASE2	35	ge8	3
UC	38	xe17	3

Table 7-2 ARTM-9405 2x40G/8x10G (continued)

Description	Marvell Port No# (only for Debugging purpose)	SRS Port	Port Group
RTM QSFP 2 - port 3 (Inactive in Group Mode)	40	xe18	3
RTM QSFP 2 - port 4 (Inactive in Group Mode)	41	xe19	3
RTM QSFP 2 - port 1 40G (Inactive in Group Mode)	42	xe20	3
RTM QSFP 2 - port 2 (Inactive in Group Mode)	43	xe21	3
RTM_SFPP_1	48	xe22	4
RTM_SFPP_2	49	xe23	4
RTM_SFPP_3	50	xe24	4
RTM_SFPP_4	51	xe25	4
RTM_SFPP_5	52	xe26	4
RTM_SFPP_6	53	xe27	4
RTM_SFPP_7	54	xe28	4
RTM_SFPP_8	55	xe29	4
RTM QSFP 1 - port 1 (Inactive in Group Mode)	56	xe30	4
RTM QSFP 1 - port 2 (Inactive in Group Mode)	57	xe31	4
RTM QSFP 1 - port 3 40G (Inactive in Group Mode)	58	xe32	4
RTM QSFP 1 - port 4 (Inactive in Group Mode)	59	xe33	4

Figure 7-1, illustrates the ATCA-9405 98CX8234 Marvell Switch.

Figure 7-1 ATCA-9405 98CX8234 Marvell Switch



8.1 Overview

ViewCheck is a comprehensive software service that can be used to diagnose, manage, and monitor Artesyn ATCA blades. The diagnostic utilities of ViewCheck help in identifying, detecting, and locating hardware issues on a blade. ViewCheck also provides mechanism to monitor status of Storage devices, Ethernet counters and errors.

ViewCheck can be accessed locally using CLI and XML interfaces provided via Blade Services Framework (BSF) service.

ViewCheck can provide both:

- InService diagnostics
- Out of Service diagnostics

InService Diagnostics
In this mode, the diagnostics service can run while the blades are instantiated with customer applications and providing service.

ViewCheck can monitor key hardware parameters like Network Counters, Network errors, and in addition to watching out for kernel critical errors logged by various hardware devices and device drivers.

Out of Service Diagnostics
In this mode, a service OS is provided with various Out of Service Diagnostics tests and utilities. This Service OS has to be loaded and diagnostics utilities can be executed. To execute these tests, CLI /XML access is provided.

Both InService and Out of Service are provided as binary images. For more information on Binary images and installation procedures, refer *Installation chapter in ViewCheck on ATCA-9405 Installation and Use manual*.

For more information on commands supported for InService and OOS, refer *Commands Execution chapter in ViewCheck on ATCA-9405 Installation and Use manual*.

ViewCheck on ATCA-9405 is supported on P2020 service processor (SP) and two Cavium OCTEON II Packet Processors (PP). Procedure to install and accessing ViewCheck on SP and PP are different, refer *Installation chapter in ViewCheck on ATCA-9405 Installation and Use manual*.

8.2 ViewCheck Access Methods

This section explains the different methods to access ViewCheck services on the Artesyn ATCA-9405 blades.

You can access ViewCheck using the following interfaces.

- CLI
- XML

Using these interfaces, you can

- Initiate a diagnostic test
- Query available diagnostic tests
- Query status of a particular diagnostic test
- Start and stop monitoring
- Stop a diagnostic test
- Generate Hardware inventory

8.2.1 CLI

CLI is one of the interfaces provided to access the ViewCheck capabilities on the blade. Using CLI, you can start, stop, and query kind of primitives at this prompt. The ViewCheck CLI can be accessed via a console using SSH.

Blade Services Framework, a proprietary service of Artesyn, is used to provide the CLI access to ViewCheck service. BSF binaries are provided along with the ViewCheck binaries.

For more information on BSF RPMs and Installation procedures, refer *Installation chapter in ViewCheck on ATCA-9405 Installation and Use manual*.

8.2.2 XML

XML is also one of the primary interfaces to access ViewCheck capabilities on the blade. XML interface supports methods, classes, and event notification mechanism. Using XML, you can start, stop, query, and configure the parameters related to tests and monitors. XML interface can be accessed in the same manner as CLI and is provided by BSF.

In addition, XML notifications are generated with details:

- State changes about the diagnostic test under execution
- Pre-determined monitor crossing set Threshold value
- Occurrence of any pre-determined hardware device error/warning generated by the device driver (or) the kernel on the blade.

A.1 Artesyn Embedded Technologies - Embedded Computing Documentation

The publications listed below are referenced in this manual. You can obtain electronic copies of Artesyn Embedded Technologies - Embedded Computing publications by contacting your local Artesyn sales office. For released products, you can also visit our Web site for the latest copies of our product documentation.

1. Go to www.artesyn.com/computing/support/product/technical-documentation.php.
2. Under **FILTER OPTIONS**, click the **Document types** drop-down list box to select the type of document you are looking for.
3. In the **Search** text box, type the product or document name and click **Filter**.

Table A-1 Artesyn Embedded Technologies - Embedded Computing Publications

Document Title	Publication Number
ATCA-9405 Installation and Use	6806800M71
SRstackware Switch Configuration Command Reference	6806800P76
ATCA-9405 Datasheet	"Datasheet"
ViewCheck on ATCA-9405 Installation and Use	6806800S51

A.2 Related Specifications

For additional information, refer to the following table for related specifications. As an additional help, a source for the listed document is provided. Please note that, while these sources have been verified, the information is subject to change without notice.

Table A-2 Related Specifications

Organization	Document Title
Intel developer.intel.com/design/servers/ipmi	Platform Management FRU Information Storage Definition v1.0
Intel developer.intel.com/design/servers/ipmi	IPMI Specification V2.0
PICMG picmg.org/specifications.stm	PICMG 3.0 Revision 2.0 Advanced TCA Base Specification PICMG 3.1 Revision 1.0 Specification Ethernet/Fiber Channel
Broadcom	BCM8725 Preliminary Data Sheet (Doc. No. 8725-DS01-R, September 19, 2007)
Marvell	Prestera-DX Packet Processors Hardware Design Guide (MV-S300644-00, Rev. D, December 27, 2006) 92DX5126 Hardware Specifications (MV-S104553-00, Rev. C, March 5, 2008)
Service Availability Forum Specifications http://www.saforum.org	SAI-HPI-B.01.01 Hardware Platform Interface Specification
Service Availability Forum Specifications http://www.saforum.org	SAI-AIS-A.01.01 Application Interface Specification
Service Availability Forum Specifications http://www.saforum.org	SAI-HPI-SNMP-B.01.01
Service Availability Forum Specifications http://www.saforum.org	SAIM-HPI-B.01.01-ATCA SAF HPI-to-AdvancedTCA Mapping Specification

ARTESYN™
EMBEDDED TECHNOLOGIES