HW Diagnostics Specification

Requirement Specification

MBB HWP eSW

24.10.2013

ATCA and BCN platforms

Table of contents

[Summary of changes 4](#_Toc370384684)

[1 Introduction 5](#_Toc370384685)

[1.1 Scope of document 5](#_Toc370384686)

[1.2 Definitions 5](#_Toc370384687)

[2 Common Requirements 6](#_Toc370384688)

[2.1 Diagnostic Output Management 8](#_Toc370384689)

[2.2 Online HW Error Monitoring 10](#_Toc370384690)

[2.3 Offline Diagnostics 11](#_Toc370384691)

[3 Functional Areas 11](#_Toc370384692)

[3.1 Intel CPUs 11](#_Toc370384693)

[3.1.1 Diagnostic info set 11](#_Toc370384694)

[3.1.2 Online test list 12](#_Toc370384695)

[3.1.3 Offline test list 12](#_Toc370384696)

[3.2 Unit Computer 12](#_Toc370384697)

[3.2.1 Diagnostic info set 12](#_Toc370384698)

[3.2.2 Online test list 13](#_Toc370384699)

[3.2.3 Offline test list 13](#_Toc370384700)

[3.3 Storage (HDD, Flashes, etc.) 13](#_Toc370384701)

[3.3.1 Diagnostic info set 13](#_Toc370384702)

[3.3.2 Online test list 14](#_Toc370384703)

[3.3.3 Offline test list 14](#_Toc370384704)

[3.4 Line Interfaces (SDH, Ethernet, Serial Interfaces) 14](#_Toc370384705)

[3.4.1 Diagnostic info set 14](#_Toc370384706)

[3.4.2 Online test list 15](#_Toc370384707)

[3.4.3 Offline test list 15](#_Toc370384708)

[3.5 Ethernet Technology 15](#_Toc370384709)

[3.5.1 Diagnostic info set 15](#_Toc370384710)

[3.5.2 Online test list 16](#_Toc370384711)

[3.5.3 Offline test list 17](#_Toc370384712)

[3.6 Data Processing Block (DSP, Octeon CPU) 18](#_Toc370384713)

[3.6.1 Diagnostic info set 18](#_Toc370384714)

[3.6.2 Online test list 18](#_Toc370384715)

[3.6.3 Offline test list 18](#_Toc370384716)

[3.7 Memory 18](#_Toc370384717)

[3.7.1 Diagnostic info set 18](#_Toc370384718)

[3.7.2 Online test list 19](#_Toc370384719)

[3.7.3 Offline test list 19](#_Toc370384720)

[3.8 Clock / Synchronization 20](#_Toc370384721)

[3.8.1 Diagnostic info set 20](#_Toc370384722)

[3.8.2 Online test list 20](#_Toc370384723)

[3.8.3 Offline test list 21](#_Toc370384724)

[4 HW Management 22](#_Toc370384725)

[4.1 Diagnostic info set 22](#_Toc370384726)

[4.2 Online test list 22](#_Toc370384727)

[4.3 Offline test list 23](#_Toc370384728)

[5 References 24](#_Toc370384729)

[6 Glossary 25](#_Toc370384730)

# Summary of changes

|  |  |
| --- | --- |
| Scope: | HWP (ATCA/BCN) Diagnostics Specification for AB6/BCN3.5 |
| Editor: | Jari Malinen |
| Location: | Sharenet-IMS <https://sharenet-ims.inside.nokiasiemensnetworks.com/Overview/D504304861> |

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Issue | Date | Handled by | Status / Comments |
| 1.0 | 24.10.2013 | J. Malinen | First common version for AB6/BCN3.5, rebranding |
|  |  |  |  |
|  |  |  |  |

Approved by

|  |  |
| --- | --- |
| 25.10.2013 | Mikko Piironen |

# Introduction

## Scope of document

The document defines diagnostics requirements in terms of functionality and implementation for tests, monitoring and information reporting that are meant to be implemented by HW vendors for ATCA and BCN projects.

Diagnosis functions have the following goals:

* improving high availability of the plug-in unit by means of self-diagnosis and self-correction actions carried out by the unit eSW itself,
* troubleshooting by helping users to locate faults when abnormal behaviour of the plug-in unit is observed, where root cause appears to be in the plug-in unit eSW/FW or HW,
* facilitating the collection of data used for fault reporting to the plug-in unit supplier, designed to detect potential faulty components inside a HW FRU under diagnostic test or during in-service processing.
* Learning acceptable vs. non-acceptable levels of errors to define the truly faulty unit.

The term “diagnostics” is often used also in a broader sense, to cover all kinds of capabilities to detect eSW/FW/HW faults and to determine the causes of symptoms, mitigations, and solutions. These can include e.g. utilizing processor error management functionality or on-line monitoring of sensors in the HW and using HW events for informing about exceeding of threshold values in the monitored sensor data. Generally it is preferred that the HW and eSW provide good capabilities for on-line monitoring of sensors in the HW, so that possible HW faults could be detected in an early stage through HW events related to these events, without any additional diagnostic tests being invoked.

### Definitions

HW diagnostics requirements can be categorized into:

* information to be provided,
* online diagnostics,
* offline diagnostics.

**Information** contains basic data that provides the user with identification info and inventory data such as the eSW versions, configuration & test definition data, statistics data, environment variables and so on.

The FRU supplier must provide tools (commands and scripts) that provide this basic information. These must be possible to use from the local Unit Computer and from a remote entity, such as the System Manager, while the unit is providing service. In particular, with regards to the eSW versions, the supplier must provide a mechanism to retrieve each running eSW version for each FRU in the running OS e.g. from Linux.

**Online diagnostics** refers to tests that can be executed on demand while the FRU is providing service.

These include monitoring, supervision and self-recovery actions that the FRU must execute during normal operation. This is to guarantee that the unit eSW is functioning properly and to report abnormal situation or faults by means of spontaneous events (e.g. traps, SEL events).

**Offline diagnostics** refers to such tests that require setting the FRU into a special test mode, where the services provided by the normal SW are not available.

Offline tests are always intrusive, compromising availability of services. Some on-line tests can also be intrusive, and can compromise serviceability, even though they would not explicitly set the unit out of service. Offline test shall also not have any impact on any other in-service FRUs residing in the system at the time of testing.

The purpose of offline tests is the identification of latent faults and fault localization. These tests can be performed remotely via the network. Offline tests take the blade out of service and may disable all external network ports except the base port and the unit computer port at the face plate.

# Common Requirements

This chapter lists the common requirements for ATCA and BCN products.

In general, diagnostic SW shall be implemented to detect HW problems.

1. Each intelligent board (FRU) shall have comprehensive diagnostics test support provided by the HW vendor. For field replaceable board extensions not capable of running tests independently (e.g. daughter-card, DIMM, RTM), tests shall be supported by the main board.
2. The diagnostic test suite shall cover verifying FRU components/functions representing at least 80% of FRU predicted failure rate.
3. The diagnostic test suite shall contain comprehensive memory test tools, capable of testing virtually all installed memory for common failure modes. As the OS normally tends to allocate a significant share of the memory to other purposes, special test-mode measures may be needed to get around this obstacle.
4. It shall be possible to interrogate what diagnostic tests are made available in each HW component. Details of each test shall include e.g. list of sub-tests included, CLI syntax, parameters, expected maximum execution time, and whether the test is intrusive. This information needs be well documented in the accompanied test manuals.
5. ECC (Error Checking and Correction) shall be supported and used to monitor online and detect errors in offline monitoring.
6. Execution of a HW diagnostic test shall produce a comprehensive test report file in human readable format, onto onboard non-volatile media, other than boot flash.
7. It shall be possible to retrieve the report of the last diagnostics test run.
8. Every error item reported by diagnostic shall correspond to real HW failure.
9. The test report shall be unambiguous to the user, in stating whether the overall outcome is passed, warning or failed.
   1. PASSED shall mean, that no errors above acceptable level found and no replacement required.
   2. WARNING shall mean that some minor errors found, but no immediate replacement required. This shall be interpreted as a sign of high probability of getting errors in the future, early warning of what is coming.
   3. FAILED shall mean, that the target FRU is recommended to be replaced; the HW vendor’s judgement based on test findings.
10. Every diagnostic report shall contain a standard-formatted test summary footer part in its end in plain text format. The test summary footer part shall contain the following lines:

Test summary: PASSED|WARNING|FAILED

Test ID: <name of the diagnostic test case, given by the HW vendor>

Report Name: <name of the report file>

File Name: <full path name of the report file>

Sub Tests: <a comma-separated list of the sub-tests contained in the diagnostic test executed>

Failed Sub Tests: <a comma-separated list of the sub-tests that failed in the diagnostic tests executed>

Start Time: <a timestamp indicating when the diagnostic test was started; e.g. 2014-01-21 08:03:30>

End Time: <a timestamp indicating when the diagnostic test was finished; e.g. 2014-01-21 08:04:27>

Board Manufacturer: <the name of the board manufacturer>

Board Name: <the name of the FRU tested; e.g. “ACPI6-A”>

More thorough information about the progress and more detailed error status of each sub-test shall be contained in the diagnostic report body, preceding the test summary footer part.

1. Board specific diagnostic test execution shall not affect the operation of other FRUs in the system. Intrusive tests on dependent board-sets (e.g. carrier/module) are an unfortunate exception to this. Intrusive tests shall only be utilized, if that is the only practical way to provide sufficient coverage. The user shall be guided to take appropriate measures to minimize the potential impact of running such tests.
2. Board specific diagnostic test execution shall not affect the diagnostic test operation of other FRUs.
3. It shall be possible to check the HW diagnostics test readiness for starting.
4. It shall be possible to cancel (stop) a running HW diagnostics test in a safe way. No test report is required in this case.
5. It shall be possible to interrogate what diagnostic tests are made available in each FRU. Details of each test should include e.g. list of sub-tests included, CLI syntax, parameters, expected maximum execution time, and whether the test is intrusive. This information shall be well documented in the accompanying test manuals.
6. It shall be possible to retrieve the status of currently running diagnostic test. Status information shall inform about e.g. test round, percentage completed and number of tests outstanding.
7. A board test suite shall contain multiple focused tests/test-sets, from which the user can select the one/ones to be run.
8. A single run of a diagnostic test shall be engineered not to last over 5 minutes. Note, that this is considered satisfied, if the parameters of the test command (like memory range) can be used to control the duration accordingly, on user choice.
9. Boards that support RTMs shall support diagnostic tests for the RTM as well.
10. HW diagnostics “memory dump” feature should not be optional or command line parameter or API argument to exclude the “memory dump” feature when the diagnostic is triggered.
11. Diagnostic tests should not result in reset/powercycle of the board without explicit confirmation from the user.
12. Diagnostics for boards targeted to run NSN SW shall be supported in the NSN Linux environment.
13. Diagnostics tests shall be controllable using CLI, and HPI DIMI.
14. Diagnostics tests shall be controllable using SSH, SOL or other remote connection utility.
15. Diagnostics tests shall be able to execute from shell script without human intervention.
16. Implementation of DIMI-based HW Diagnostics shall follow requirements specified in “HPI FUMI - DIMI Implementation Guide” [1].
17. ATF (Automated Test Framework) based diagnostics shall be supported in selected FRUs.

## Diagnostic Output Management

In the following, some general rules regarding the handling of the Blade/FRU diagnostics info are detailed.

1. The diagnostics output (information and test results) shall always be accessible from a remote entity (e.g. the System Manager).
2. Diagnostics implementation shall support CLI as the system management interface.
3. The interface to the tester for preparing, executing and collecting the diagnostic info shall be based on DIMI standard as specified in SAI-HPI-B.03.02. All boards and AMCs containing a management processor shall support a HPI DIMI interface. For further details and requirements see “HPI FUMI - DIMI Implementation Guide” [1].
4. All the logs and test reports related to diagnostic tests shall be available also after a reboot or a power cycle.
5. Consolidated summary of test results should be displayed on terminal and as well in the report.
6. The diagnostics output shall be stored in non-volatile storage media of the FRU by the FRU Diagnosis eSW itself. E.g. if the OS running on the FRU is Linux the files can be stored under a specific directory in /var/log.
7. Each diagnostic log file shall be identified by a test identifier and a time and date in order for each file to be unique and to ensure that only a user specific command can delete/purge the logs directories.
8. In case the disk/memory space is not sufficient for storing the log (report) files an error message shall be reported to the user. This error indication shall be included in the Hi-Level log.
9. If it is not possible to warn the user then the oldest log files shall be automatically deleted from the log directory. This automatic deletion operation shall be reported in a hi-level log.
10. In general, it would preferable having all collected information data related to counters, statistics and traffic represented in XML format.
11. The log (report) file as well should have a XML structure and format. In the following, some example of XML file.

eswVersionInfo.xml:

<blade\_diagnostic>

<software\_version>

<linux>1.2.3.4.5\_buildA</linux>

<fpga>1.2.3</fpga>

<upgradable\_component>GeneralVersionInformation</upgradable\_component>

</software\_version>

</blade\_diagnostic>

trafficInfo.xml[[1]](#footnote-1):

<blade\_diagnostic>

<traffic\_counters>

<TotalPacketsReceived>12334</TotalPacketsReceived>

<TotalReceivedPacketsNotForwarded>2143</TotalReceivedPacketsNotForwarded>

<LocalTrafficFrames>43524</LocalTrafficFrames>

<PauseFramesReceived802.3x>52435</PauseFramesReceived802.3x>

<UnacceptableFrameType>63456</UnacceptableFrameType>

<MulticastTreeViableDiscards>5635</MulticastTreeViableDiscards>

<ReservedAddressDiscards>6345643</ReservedAddressDiscards>

</traffic\_counters>

</blade\_diagnostic>

1. The diagnostics information shall contain a single hi-level log that stores the event of the diagnostic module/feature for example user request, errors, warning, etc., in general simple information that shows the time/date of the event and the event type.

Example:

| <time> <date> | <type> | <event>

| 15:44:19 11/12/2010 | INFO | Start Diagnostic test Command From Host:10.22.33.45

| 15:42:19 11/12/2010 | INFO | Get Diagnostic Info from Host:10.22.33.45

| 15:40:19 11/12/2010 | WARNING | No Space Left

| 15:39:19 11/12/2010 | INFO | xxx.xml file deleted

| 15:36:19 11/12/2010 | ERROR | Diagnostic Module Crash

…

…

type : INFO, WARNING, ERROR, etc.

1. All reports shall be stored in one directory and the file name shall reflect what kind of test was executed. Online background diagnostics information (i.e. not triggered by user) shall be logged outside HW diagnostics file structure. HW event logs shall be stored separately.
2. The FRU shall support both online diagnostics and offline tests. The offline diagnostic tests shall be marked as DEGRADING in the DIMI resource descriptor record. In the CLI, offline tests shall be clearly marked as such.
3. Same diagnostic tests shall be supported via CLI and HPI DIMI.
4. The supplier shall provide an OpenHPI plugin for their DIMI interface.
5. The diagnostic CLI shall not allow for the user to start overlapping diagnostic tests or start the same test concurrently from different interfaces.
6. Error monitors should have configurable error thresholds which will result in IPMI events when triggered.
7. The board support package shall contain tools to access status data provided by board self-test / offline diagnostics.
8. The board shall support LED indicators for status visualization. Please see Platform Architecture Specification (BCN HW Platform) and LED Control in ATCA FRU documents for more details.

## Online HW Error Monitoring

1. Intelligent FRUs shall make use of the HW error monitoring and logging features supported by the underlying HW, in order to provide an extensive early HW failure detection capability. Note: HW errors here refer to misbehaviour of local HW such as bus/link errors, memory errors, other device errors, usually detected via a form of integrity check (CRC, parity, etc). From error detection, an error notification is needed, to alert System management and/or the user for further actions.
2. Different types of errors shall be signalled differently referring to warning/failed categories, e.g. correctable vs. uncorrectable flag.
3. Where built in HW monitors do not provide good functional error coverage, additional SW based background HW monitors shall be provided as a complement.
4. Monitoring for errors shall be based on generic Linux tools (like EDAC, MCElog) where available. Using such tools promotes commonality across different vendor/board/CPU environments for the integration path towards related SW platform subsystems.
5. Fatal/catastrophic (potentially all uncorrectable) HW errors shall be reported to SM via a mechanism (such as SMI/NMI triggered flag towards IPMC), that can be trusted even under the errored condition in the target (still best effort). Fatal error signalling shall be implemented in HW level.
6. The error monitors for device/error classes that have a suitable standard IPMI/HPI mapping shall be exposed in the form of standard IPMI sensors [4], and report via events as their trigger condition is met. Proprietary/OEM sensors shall be implemented upon NSN requirements.
7. Correctable HW errors shall by default be corrected on the fly, and the error rate monitored via counters with adjustable alert thresholds. Exceeding the error rate limit thus becomes a non-fatal error, to be reported to System management.
8. The default thresholds for the error monitors shall be set by the board vendors, based on their experience of normal acceptable error rate, on a non-faulty board. The defaults by convention are to be agreed with NSN.
9. The error monitors shall be remotely configurable from System Manager, for enable/disable, thresholds, and potential local recovery actions, e.g. bad memory page offlining.
10. Diagnostic tools shall not utilize the CPU resources more than 1-2% at any during the run or generating report or compressing the report.

## Offline Diagnostics

1. The set of off-line diagnostic tests shall localize the fault to the right FRU. Vendor shall provide the failure coverage. FRU here refers to the smallest field replaceable module, that contains the faulty device; e.g. DIMM, daughter card.
2. The set of tests needs to be designed both for detection coverage, and localization accuracy, utilizing the Error Management HW capabilities if possible.
3. Before starting off-line monitoring, the FRU shall read and store ‘sticky bit error information’, i.e. register bit that always stores information except in power cycle.

# Functional Areas

This chapter lists the requirements for ATCA and BCN platforms per different functional units or FRUs.

The ‘Diagnostics Info Set’ chapter defines the specific information requested for the FRU in case. Online and offline test lists define the minimum test cases requested.

## Intel CPUs

### Diagnostic info set

* Bios settings readout (both banks)
* Boot device
* BIOS version
* opROM status (enable/disable)
* ECC s/d bit error counters readout: The unit shall support SECDED ECC error correction for the DRAM memory. Errors reported (correctable/uncorrectable) shall be configurable by SW.
* SMBIOS event log readout: SMBIOS event log shall be supported
* FRU Upgradable components versions readout
* RAM HW block shall be diagnosed
* Ethernet HW block shall be diagnosed
* RAM supervision shall be supported. Supervision shall supervise single/multiple bits errors in each DIMM. RAM supervision is done by checking MCE log.
* Error correction logic enabled by HW shall be enabled
* There shall be thresholds for correctable errors when enabled by HW. Exceeding thresholds will be signalled to OS. It shall be possible for user to adjust correctable error thresholds.

### Online test list

* Integrity check on BIOS flash content in CPU startup: Bad checksum for the system BIOS shall result in automatic switching to the backup version.
* BIOS firmware copies identity check
* Read/Write test on stand-by flash bank (e.g. BIOS reflashing without disturbing normal operation!)
* Check error registers in case of error monitoring exceeding predefined thresholds.

### Offline test list

* Test multiple/single reset of the unit
* Stress memory and PCIe with data and address patterns and determine functionality with error management HW, e.g. by reading error registers and amount of retries.
* Internal CPU faults shall be reported by error management.

## Unit Computer

This chapter applies to shelf manager requirements as well.

### Diagnostic info set

* Uboot Environment variables readout: It shall be possible to get the data in diagnostics report (if needed).
* POST results readout: The Power-On Self Test (POST) performs initialization and basic tests of the processor core and unit computer block components on which the processor depends. All peripherals of the unit computer block shall be tested.
* FRU data.

### Online test list

* Integrity check on boot flash content: Boot flash shall contain checksum and version information for bootloader, environment variables and OS segments.
* OS – Kernel Integrity supervision
* Kernel
* Filesystem
* Persistent memory (Flash, EEPROM, etc.)
* Availability (check presence and checksum of memory unit)
* ECC Counters Readout
* The Shelf Manager shall also support the following:
* Master-only I2C bus monitoring
* Check that all sensors of the shelf and Shelf Manager’s own sensors on I2C bus are readable
* I2C protocol errors
* I2C/SMBus bus time out
* USB interface monitoring:
* USB interface protocol errors (CRC, packet sequence errors).
* HW redundancy interface
* Shelf Manager shall monitor the status of the redundant Shelf Manager via HRI interface for peer health/presence monitoring.

### Offline test list

* External memory (RAM) test
* ECC counter readout
* E.g. x86 memory test features
* Size of memory test shall be configurable
* Interface tests e.g. PCIe, sRIO (DSP blade only), Ethernet etc.

## Storage (HDD, Flashes, etc.)

### Diagnostic info set

* SMART Information set readout (SATA/SAS), including estimated lifespan (for SSD)
* MTD info readout for flashes.

### Online test list

* Consistency Check (RAID)
* MTD access test
* SMART error statistics information read out (e.g. number of bad blocks to determine aging).

### Offline test list

* Sectors Check (SATA/SAS)
* Integrity access check (FLASH)
* Benchmarks
* SMART provided diagnostic.

## Line Interfaces (SDH, Ethernet, Serial Interfaces)

This chapter describes end point diagnostics of line interfaces.

### Diagnostic info set

* The line interface checks shall be performed in any reset. It shall also be possible to perform the checks by a CLI command and through SNMP.
* It is acceptable that the switch is not capable of passing traffic during the diagnostics.

The FRU shall implement at least the following diagnostics:

* Verifying the validity of contents of all flash and prom memory
* Verifying LAN port functionality Firmware versions.
* The FRU shall monitor input errors in each link (CRC, alignment, bad symbol, Frame undersize, oversize etc) and collect traffic statistics.
* If the error rate exceeds configured threshold, the link shall be considered failed and recovery actions shall be started (provided that RSTP or MSTP is enabled and a suitable redundant link exists).
* The integration time shall be around one second, but if the implementation is SW-based (i.e. performed by the Unit Computer) it is acceptable to have integration time of 30...60 seconds.
* It shall be possible to enable or disable this feature. By default, this feature shall be disabled.
* High rate of runts or oversize frames must not trigger (rapid)-STP or MSTP reconfiguration.

### Online test list

* Internal EEPROM integrity check
* MAC duplication test.

### Offline test list

* SFP Tests
* Laser turn on/off check
* Check LOS presence within laser turned OFF (Loss of Signal alarm does make sense within optical fibres ON). The FRU shall support the digital diagnostic interface for the pluggable (SFP/SFP+/XFP/ QSFP/CXP) interface.
* It shall be possible access these pluggable ports individually. The LOS pin of the interface must also be routed for future debugging purposes.
* Check laser power values, benchmark with default values
* Diagnostic loopback
* Internal loopback (copper only, via SW):



* Link-up/down detection check
* link-up/down detection speed check
* Speeding negotiation check (HD,FD).

## Ethernet Technology

This chapter describes Ethernet switch related diagnostics.

### Diagnostic info set

* Switch configuration information (e.g. running-configuration, /etc.)
* SW Network Platform version (e.g. Fastpath, ZebOS, etc. version)
* HW product details
* Log information (e.g. retrieved from …/log)
* Dip-switch states (e.g. jumpers)
* Switch link states (e.g. up/down, autonegotiation, etc.)
* Switch link counters (e.g. errors, statistics, etc)
* Switch processor load
* SNMP info (e.g. traps, MIB, etc.)
* Packet size info.

### Online test list

* Traffic forwarding capabilities of the unit (e.g. duplication of traffic packets to a fixed port)
* Unit port functionalities for valid and not valid traffic:
* get port counters
* wait for 1 second
* get port counters
* calculate delta traffic flow

Also the following counters should apply this:

* Number of error/anomalous events, e.g.
* Total number of Collisions seen by MAC
* Total number of Late Collisions seen by MAC
* Number of RX ERROR events seen by the receive side MAC
* Number of frames not transmitted by MAC due to internal errors
* Number of CRC error events
* Number of Undersize packets received
* Number of fragments received
* Number of oversize packets received
* Number of jabber packets received
* Number of MAC Control Frames received with opcode different from 00-01
* Number of good Flow Control Frames received
* Number of Flow Control Frames sent

Example:

Packets Received Without Error.......... 254252 (32 Pkt/s)  
Packets Received With Error.................... 242 (232 Pkt/s)  
Broadcast Packets Received................242354252 (542 Pkt/s)  
Packets Transmitted Without Errors....... 4234252 (782 Pkt/s)  
Transmit Packet Errors......................... 423452 (454 Pkt/s)  
Collision Frames............................... 4234252 (2142 Pkt/s)

* SNMP trap[[2]](#footnote-2)
* Reach other subsystem devices[[3]](#footnote-3)
* Verify possibility for unit authentication and availability of configuration services (e.g. snmp, ssh, telnet daemons, etc.).

### Offline test list

These tests shall be performed after the reset. It shall also be possible to perform the tests by a CLI command and through SNMP. During offline tests the switch is not capable to manage traffic. Below areas to be tested:

* POST
* Internal loopback traffic test
* External loopback traffic test
* Test multiple/single reset of the unit[[4]](#footnote-4)
* Validation of the running-config   
  - Save the running-config   
  - Restore the default running- config   
  - Check that the system is working   
  - Apply the saved running-config  
  - Check that the system is working
* Switch processor load test.

## Data Processing Block (DSP, Octeon CPU)

### Diagnostic info set

* FRU data

### Online test list

* Test heartbeat messages (from unit computer to DSP)

### Offline test list

* DSP internal tests with TI POST tools
  + Included in DSP MCSDK
* DSP external memory tests
  + E.g. x86 memory test features
* DSP interface tests
  + SRIO tests (in future this might be PCIe)
    - Messages transferred with Unit computer
  + Ethernet interface tests
    - Messages transferred with Unit computer
* Octeon DRAM memory test to cover whole memory area
  + Options: iteration count, data pattern, core count 1 to all
* Octeon MIPS-core test
  + Options: core count 1 to all
* Octeon simple Interface tests:
  + UART
  + PCIe
  + Ethernet
  + FLASH
  + I2C
* Octeon Traffic generator or IP/MAC packet forwarder to test packet interfaces
  + Options: core count 1 to all.

## Memory

### Diagnostic info set

Installed configuration:

* Number of banks/elements
* Mode (single/double/triple channel)
* Memory Size

Detailed information:

* Technology
* Speed
* Vendor
* Density
* CAS Latency.

### Online test list

The unit in service should provide online diagnostics monitor parameters and healthiness of the unit computer memories checked for load, accessibility and errors.

* Persistent memory (ROM for BIOS, EEPROM, etc.)
  + Availability (check presence and checksum of memory unit).

### Offline test list

* POST (Power On Self Test about memory)
* The blade should perform a self test after power up which validates at least all relevant memory resources of the unit computer.
* The POST shall identify failures which may prevent the OS from booting. The POST progress should be reported in IPMC registers which can be read even if the boot process stops working due to a HW failure.
* Persistent memory (ROM for BIOS, EEPROM, etc) from an external unit (i.e. from either OMU/CLA or Shelf Mgr via IPMI)
* Availability (check presence of memory unit)
* Accessibility (check correctness of the entire memory unit)
* RAM (check that Unit Computer's RAM is functioning correctly)
  + ECC errors (if supported by HW)
  + Dump of the working memory (for debugging purposes)
  + Data bus test (walking 1 test pattern)
  + Address bus test (walking 0 and walking 1 test patterns)
  + SDRAM read/write test using different test patterns.

## Clock / Synchronization

clock

Figure: Example of AHUB3 HW synchronization schema

### Diagnostic info set

* Input clock status
* Input clock quality statistic (Activity, soft alarm, hard alarm, etc.)
* Time used for locking
* Internal parameters set to registers
* HW error indication
* MCG operation state statistic (lock, free-run…)
* PLL operation state statistic (lock, free-run…).

### Online test list

N/A

### Offline test list

Pre-steps:

* Set MCG role to autonomous, trigger peer to be master
* Disable CLK1, CLK2 output during diagnostic

Diag-steps:

* MCG disable a specific input clock
* MCG enable the input clock
* MCG try to lock to the input clock for several minutes.
* Input clock switch over test.

# 4 HW Management

The term “platform management” is used to refer to the monitoring and control functions that are built in to the platform hardware and primarily used for the purpose of monitoring the health of the system hardware. This typically includes monitoring elements such as system temperatures, voltages, fans, power supplies, bus errors, system physical security, etc. It includes automatic and manually driven recovery capabilities such as local or remote system resets and power on/off operations. It includes the logging of abnormal or ‘out-of-range’ conditions for later examination and alerting where the platform issues the alert without aid of run-time software. Lastly it includes inventory information that can help identify a failed hardware unit.

The Hardware Platform Management shall be based on the IPMI version 2.0 specification [4] and backward compliant with the IPMI version 1.5 specification.

Unit IPMI Controller shall be capable of issuing a Diagnostic Interrupt to the Payload according to the PICMG 3.0 [3].

## 4.1 Diagnostic info set

Examples of information that IPMI implementation shall include, but not limited to.

1. IPMI Get Device ID Command to check IPMI version and IPMC FW version on Unit Controller.
2. Get Address Info Command to check Hardware Address of unit controller.
3. IPMI Unit Controller shall supervise/monitor all sensors. This will read the sensor list equipped on Unit Controller. Using the list it can be checked that all equipped sensors are compliant to HW Design Specification of Unit Controller and that all sensor names are correct. Additionally checking that all equipped sensors shall be properly read and no alarm must be present.
4. It shall be possible to access, in read/write mode, the FRU Information Area Storage, via standard IPMI Commands.
5. IPMB-0 Link status check to get IPMB link info. It will return the IPMB-0 Sensor Number that is monitoring the associated IPMB-0 Link.
6. FRU Information Area Storage reading. With this the compliance with NSN specification can be checked.

## 4.2 Online test list

1. IPMI Unit Controller shall supervise/monitor all sensors.

## 4.3 Offline test list

1. IPMI Cold/Warm Reset Commandto force a Hard Reset of the IPMI Controller (without affecting Payload Area)
2. IPMI FRU Control - Cold Reset Option to force a Hard Reset of the Payload Area (without affecting the IPMI Controller.
3. Validation aTCA Payload Interface (UART)
4. ATCA Unit Controller Deactivation/Activation by SW

# References

1. HPI FUMI DIMI Implementation Guide\_v3.1.doc (Sharenet link: https://sharenet-ims.inside.nokiasiemensnetworks.com/livelink/livelink?func=ll&objId=401609138&objAction=Browse)
2. Benchmarks:
   1. BMIPS <http://en.wikipedia.org/wiki/BogoMips>
   2. MIPS <http://en.wikipedia.org/wiki/Instructions_per_second>
   3. FLOPS <http://en.wikipedia.org/wiki/FLOPS>
   4. <http://lbs.sourceforge.net/>
   5. Whetstone/Dhrystone
3. PICMG 3.0 AdvancedTCA Base Specification
4. IPMI 2.0 Intelligent Platform Management Interface Specification
5. HPI SAIM-HPI-B.03.02-xTCA Specification for HPI Mapping to AdvancedTCA® and MicroTCA®
6. BCNMB-A / BCNMB-B DIMI Concept Document

# Glossary

|  |  |
| --- | --- |
| Abbreviation | Explanation |
| AB | ATCA Blade |
| AMC | Advanced Mezzanine Card |
| ATCA | Advanced Telecommunications Computing Architecture |
| BCN | BOX Controller Network |
| BER | Bit Error Rate |
| BIOS | Basic Input-Output System |
| CPU | Central Processing Unit |
| DHCP | Dynamic Host Configuration Protocol |
| DIMI | Diagnostics Initiator Management Instrument (SAF) |
| DSP | Digital Signal Processor |
| FRU | Field Replaceable Unit (exchangeable HW component / board) |
| ECC | Error Correction Code |
| EEPROM | Electrically Erasable Programmable Read-Only Memory |
| FW | Firmware |
| ESW | Embedded Software |
| IPMB | Intelligent Platform Management Bus (I2C) |
| IPMC | Intelligent Platform Management Controller |
| IPMI | Intelligent Platform Management Interface |
| LMP | Local Management Processor |
| NE | Network Element (tenant, guest within FP-VI environment) |
| POST | Power On Self Test |
| RAM | Random-Access Memory |
| ROM | Read-Access Memory |
| SATA | Serial Advanced Technology Attachment |
| SEL | System Event Log |
| ssh | Secure Shell |
| SFP | Small Form-factor Pluggable transceiver |
| ShMgr | Shelf Manager |
| SMS | System Management Software |
| SNMP | Simple Network Management Protocol |
| SRIO | Serial Rapid I/O |
| TPM | Trusted Platform Module |

1. This example doesn't refer to a test that gives a boolean result (PASS (WARNING)/FAILED) but only to a collection of statistics. [↑](#footnote-ref-1)
2. The supplier shall provide a process/daemon that periodically (e.g. each 5/10 sec or 1 min) sends a “test trap“ for verifying that the trap itself is sent and received correctly (as client and server are running on the same unit). [↑](#footnote-ref-2)
3. NSN shall provide a list of the shelf elements which must be reached from the switch unit. [↑](#footnote-ref-3)
4. For instance, ADSP2-A sends a reset command to a single or to all DSP CPUs of the board. [↑](#footnote-ref-4)