ViewCheck on ATCA-F140

Installation and Use

P/N: 6806800S60E

March 2020

SMART™ Embedded Computing

Trademarks

The stylized "S" and "SMART" is a registered trademark of SMART Modular Technologies, Inc. and “SMART Embedded Computing” and the SMART Embedded Computing logo are trademarks of SMART Modular Technologies, Inc. All other names and logos referred to are trade names, trademarks, or registered trademarks of their respective owners. These materials are provided by SMART Embedded Computing as a service to its customers and may be used for informational purposes only.

Disclaimer*

SMART Embedded Computing (SMART EC) assumes no responsibility for errors or omissions in these materials. **These materials are provided "AS IS" without warranty of any kind, either expressed or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.** SMART EC further does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SMART EC shall not be liable for any special, indirect, incidental, or consequential damages, including without limitation, lost revenues or lost profits, which may result from the use of these materials. SMART EC may make changes to these materials, or to the products described therein, at any time without notice. SMART EC makes no commitment to update the information contained within these materials.

Electronic versions of this material may be read online, downloaded for personal use, or referenced in another document as a URL to a SMART EC website. The text itself may not be published commercially in print or electronic form, edited, translated, or otherwise altered without the permission of SMART EC.

It is possible that this publication may contain reference to or information about SMART EC products, programming, or services that are not available in your country. Such references or information must not be construed to mean that SMART EC intends to announce such SMART EC products, programming, or services in your country.

Limited and Restricted Rights Legend

If the documentation contained herein is supplied, directly or indirectly, to the U.S. Government, the following notice shall apply unless otherwise agreed to in writing by SMART Embedded Computing.

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data clause at DFARS 252.227-7013 (Nov. 1995) and of the Rights in Noncommercial Computer Software and Documentation clause at DFARS 252.227-7014 (Jun. 1995).

SMART Embedded Computing, Inc.

2900 S. Diablo Way, Suite 190

Tempe, Arizona 85282

USA

*For full legal terms and conditions, visit *www.smartembedded.com/ec/legal*

Table of Contents

Table of Contents

List of Tables

List of Tables

About this Manual

Overview of Contents

This guide provides detailed information about installation, configuration, and how to work with ViewCheck. This manual is divided into following chapters and appendix.

About this Manual lists all conventions and abbreviations used in this manual and outlines the revision history.

Chapter 1, Introduction on page 13 provides detailed overview and features of ViewCheck.

Chapter 2, Concepts of ViewCheck on page 15 describes the concepts of ViewCheck.

Chapter 3, Installation of ViewCheck on page 19 provides instructions to install ViewCheck.

Chapter 4, Commands Execution on page 37 describes the various tests that can be executed in ViewCheck.

Appendix A, Related Documentation on page 73 lists the relevant manuals and provides additional information.

Abbreviations

The following tables lists the abbreviations used throughout the document.

Abbreviation	Definition
ATCA	Advanced Telecom Computing Architecture
BSF	Blade Services Framework. A derivative of System Services Framework.
CLI	Command Line Interface
Client	The applications used to Access ViewCheck via the various Interfaces (CLI and XML)
HW	Hardware
INSM	In Service Monitoring. Functional module in ViewCheck framework handling the monitoring functionality of various critical parameters in the blade.
OOSD	Online Out of Service Diagnostics. Functional module in ViewCheck framework that manages Test Management requests related to Out of Service Diagnostics.
XML	Extensible Markup Language

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. `Sample of Programming used in a table (9pt)`
`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
..	Ranges, for example: 0..4 means one of the integers 0,1,2,3, and 4 (used in registers)
|	Logical OR
	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

Notation	Description
	Indicates a property damage message
	Indicates a hot surface that could result in moderate or serious injury
	Indicates an electrical situation that could result in moderate injury or death
	Indicates that when working in an ESD environment care should be taken to use proper ESD practices
	No danger encountered, pay attention to important information

Summary of Changes

Part Number	Date	Description
6806800S50A	October, 2013	Initial version
6806800S50B	April, 2014	Re-branded to Artesyn template.
6806800S50C	July, 2014	Added new commands in *Generic Commands on page 37*. Added new tests in *CPU on page 69*, *Memory on page 69*, and *NETWORK on page 70*. Added *ViewCheck TestLog Information on page 35*.
6806800S50D	October 2014	Updated *Installation of ViewCheck on page 19*.
6806800S50E	March 2020	Rebrand to SMART Embedded Computing template.

Introduction

1.1 Overview

ViewCheck™ is a comprehensive software service used to diagnose, manage, and monitor SMART Embedded Computing 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 Command Line Interface (CLI) and Extensible Markup Language (XML) interfaces provided via Blade Services Framework (BSF) service.

ViewCheck provides:

- **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, and network errors. It can also be used for watching kernel critical errors logged by various hardware devices and device drivers.

- **Out of Service Diagnostics**
  In this mode, ViewCheck can execute all Out of Service Diagnostics (OOSD) tests along with InService Diagnostics activities.

ViewCheck provides a command called `switch-mode`, using which you can switch from InService to Out of Service and vice versa.

For more information on commands supported for InService and OOSD, refer *Commands Execution on page 37*.

1.2 ViewCheck Access Methods

This section explains various methods for accessing ViewCheck services on SMART EC 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

1.2.1 CLI

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.

BSF, a proprietary service of SMART EC, 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 of ViewCheck on page 19*.

1.2.2 XML

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.

XML notifications are generated with following details:

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

Concepts of ViewCheck

2.1 Overview

This section explains terminology and keywords used extensively in ViewCheck services.

2.2 Test Identification

Unique Test identification is based on following triple key:

```
< Device Category, Test ID, Device Instance >
```

2.3 Device Category

The Device Category is an enumerated value, reused from similar enumeration already defined in HPI-B Standard specification.

It is used to express commonly used devices, such as Storage, Network, Serial, CPU, and Memory, available on all the blades, irrespective of function and architecture. This category is used in commands as one of the key fields to identify uniquely a particular test.

Device Category allows for:

- Grouping of test cases per category for display and statistical purposes
- Reuse of Test IDs across device categories

2.4 Test ID

Test ID is an integer value that uniquely identifies the actual test that can be invoked or executed on a hardware device instance, which belongs to a specific device category available on the blade. Following are the examples of tests that can be executed on the devices:

- Ping Flood test
- Network connectivity test in case of Network Device Category
- Bad Blocks test in case of Storage Device category

Each of these tests would be associated with a unique Test Identifier (Test ID). These Test IDs start with value ’0’ and increases linearly for various sub-tests in a device category.

Some tests may be applicable to all device instances in a particular device category. The combination of <Device category, Test ID, Device Instance> would be unique and provides capability to control, execute, and manage the test on a device instance in a device category. With this mechanism, same test can be simultaneously initiated or triggered on multiple device instances under that device category, thus providing parallel execution of tests.

2.5 Device Instance

Hardware devices uniquely identified and recognized by the drivers and OS on the blade are treated as device instances. A device instance can belong to a particular device category. Tests can be invoked and executed on this device instance. OS and driver support to access the device is assumed to be readily available.

For example, device instances eth0, eth1, eth2 or Base 0, Base 1, Base 2 are used to identify unique devices in networking device category. Similarly, hda1, hda2, and so on can identify unique instances of devices in the storage category. Device instances use the standard nomenclature already defined by the OS (for instance Linux) on the blade.

A diagnostics test identified with < Device Category, Test ID, device Instance> is executed on the specified device instance.

2.6 Monitor ID

ViewCheck service monitors pre-identified parameters for hardware devices. These parameters are network device counters, and errors. To periodically poll and check these parameters, the ViewCheck service uses CLI and XML configuration. For each parameter of interest, a Monitor ID is an enumerated constant that uniquely represents the monitoring entity. ViewCheck uses this value to control monitoring and while reporting events via XML on these monitors.

2.7 Error ID

Error ID is to provide identification for pre-determined errors/warnings of hardware devices generated by the device driver or the kernel on the blade. These critical and error messages are indications of abnormal behavior on part of the kernel or the hardware device on the blade. ViewCheck functionality attempts to detect all such errors and provides suitable information to external high-level software intelligence to act upon.

The list of messages that constitute these errors is not standardized by the hardware device Vendor nor the Linux Kernel Community. Error ID attempts to standardize all such messages on SMART Embedded Computing blades.

These messages would be OS and driver specific. Mostly, the same Error ID would be associated with the same category of error across blades and OSs. For more information on commands, refer to *Commands Execution on page 37*.

NOTES: Monitor ID and Error ID are used in InService Monitoring.

Installation of ViewCheck

3.1 Overview

This section explains the ViewCheck release modules and installation procedures to install and run ViewCheck service on the SMART Embedded Computing ATCA-F140 blade.

ViewCheck service is released as an RPM image. BSF RPMs are also distributed along with ViewCheck RPMs.

3.2 ViewCheck RPM Image

The ViewCheck RPM Image functionally comprises Diagnostics Framework, specific test cases, and test suites. The ViewCheck RPM always uses same OS variant and compile time environment based on the BBS release of the target blade. For ATCA-F140 blade, the ViewCheck RPM is created for PNE 3.x environment.

ViewCheck service RPM consists of:

- Diagnostics Core - Daemon
- Static Test Suite Configuration files for the Specific Blade
- Start/Stop Scripts for Diagnostics Core

Using the following command, you can install the ViewCheck RPM Image:

```
rpm -iv --nodeps diagnostics-
<RELEASE>_<BUILD>.<DIST>.<OS>.atcaf140.rpm
```

Using the following command, you can remove the ViewCheck RPM Image:

```
rpm -e diagnostics-<RELEASE>_<BUILD>.<DIST>.<OS>.atcaf140.rpm
```

The following table provides details of the files that are created on the blade once the ViewCheck RPM is installed.

Table 3-1 RPM Files

File Name	Path	Description
`diagcored`	`/opt/diagnostics/bin/`	ViewCheck Core - Daemon
`diagconfig.xml` `diaguserconf.xml`	`/opt/diagnostics/etc/diag/`	ViewCheck configuration file and user configuration file.
`diagcore`	`/opt/diagnostics/etc/init.d/`	Script to Start/Stop ViewCheck Core

Table 3-1 RPM Files (continued)

File Name	Path	Description
libdiagintf.so	/lib64/	Interface library between ViewCheck core daemon and BSF application.
<TestScripts>.sh	/opt/diagnostics/tools/diagtestscripts/	Test scripts
EmrDiag_Debug.log	/opt/diagnostics/var/log/diag/service/	ViewCheck daemon service log
diagLib_log, diagCore_log, diagResults_log, diagTestRaw_log, diagShowCmds_log	/etc/logrotate.d	Configuration files required for log rotation of service logs
Testutilities	/opt/diagnostics/tools/diagtestutils/	Utilities and Tools used by ViewCheck application

3.3 BSF Service RPM

BSF service is distributed as a package contains 3 RPMs, namely eMIND, BSFCore, and Diagnostics Transport layer service. Following are the list of RPMs:

ssf_main_rel-<BLADE>-<DIST>-<RELEASEBUILD>.<ARCH>.rpm

ssf_csim_rel-<BLADE>-<DIST>-<RELEASEBUILD>.<ARCH>.rpm

ssf_diagnosticsTLS_rel-<BLADE>-<DIST>-<RELEASEBUILD>.<ARCH>.rpm

You should install BSF RPMs in the following sequence:

1. ssf_main
2. ssf_csim
3. ssf_diagnosticsTLS

You can install the BSF using the following RPM commands:

- rpm -iv --nodeps --force ssf_main_rel-<BLADE>-<DIST>-<RELEASEBUILD>.<ARCH>.rpm
- rpm -iv --nodeps --force ssf_csim_rel-<BLADE>-<DIST>-<RELEASEBUILD>.<ARCH>.rpm
- rpm -iv --nodeps --force ssf_diagnosticsTLS_rel-<BLADE>-<DIST>-<RELEASEBUILD>.<ARCH>.rpm

After the installation, the BSF binary files are installed at the `/opt/ssf` location.

The BSF applications can be started, stopped, and restarted using below script.

```
/opt/ssf/etc/config/S99SsfBsfRun.sh {start|stop|restart}
[main|csim|diag]
```

NOTICE

In the above syntax, specifying BSF application name is optional. When application name is not specified, the action is performed on all the three applications (main, csim, and diag).

To uninstall the BSF, execute the following RPM commands:

- `rpm -e ssf_diagnosticsTLS_rel-<BLADE>-<DIST>-<RELEASEBUILD>.<ARCH>.rpm`
- `rpm -e ssf_csim_rel-<BLADE>-<DIST>-<RELEASEBUILD>.<ARCH>.rpm`
- `rpm -e ssf_main_rel-<BLADE>-<DIST>-<RELEASEBUILD>.<ARCH>.rpm`

NOTICE

Uninstall the BSF RPMs in the following sequence:

1. **ssf_diagnosticsTLS**
2. **ssf_csim**
3. **ssf_main**

3.4 Access and Execution of a Test Using CLI

You can access ViewCheck CLI using the following procedure:

1. Establish the secure shell using SSH or Putty.
2. Start the **Telnet** connection from an already established secure shell.

```
telnet localhost 11001

Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
Welcome to Emerson's SSF CLI
```

3. Type your username and password.

```
Username: Admin
Password:
Access granted
>enable
#configure terminal
BSF(config)#
BSF(config)#virExecEnv vee0
BSF(VEE-vee0)#diagnostic
BSF(diag-vee0)#
```

NOTICE

By default, the administrator user name and password are Admin.

After logging into the ViewCheck CLI, you can list all the supported commands by typing `'?'` on the CLI console.

```
BSF(diag-vee0)#?

    device-category         Configure device Category

    exit                    Exit from diagnostic
```

hw-inventory-list	Provides the Inventory of Hardware (Type, Vendor ID, Major Number, Minor Number and any associated Details) as detected by the Diagnostics Application.
purge-all-log	All log files are Zipped and stored away.
purge-log	Purge specific log file.
set-log-level	Sets the Log Level of Diagnostic Application.
show-log-info	List all Log files of Diagnostics Application.
show-systemerror-log	Show system error log.

You can enter into device category mode by giving the command device-category on CLI and view the list of commands supported only at device category level.

```
BSF(diag-vee0)#device-category ?
```

deviceCategory>	other, processor, hardDisk, memory, os, pciBus, pciExpressBus, scsiBus, sataBus, clock, firmware, cpld, fpga, networkinterface, digitalsignalprocessor, networkprocessingunit, interface, systemBus, flash, serial, i2cBus, spiBus, usbBus, ipmc, all

```
BSF(diag-vee0)# device-category networkinterface
BSF(diag-vee0-networkinterface)#?
```

exit	Exit from deviceCategory
list-device-instances	List all the possible device instances in present deviceCategory
list-tests	User can use this command to get information on the available Diagnostic tests with details like tests and sub tests associated, along with Test IDs

set-lower-threshold-info	Set the Lower Threshold value for the Monitor
set-rate	set the rate of change value for network monitors
set-poll-interval-default	Set Poll Interval to default value
set-poll-interval	Set the Poll Interval
set-threshold-default	Set Threshold to default value
set-upper-threshold-info	Set the Upper Threshold value for the Monitor
show-lower-threshold-info	Show Lower Threshold info
show-monitor-id	List all monitors for the deviceCategory
show-networkcounter-log	show network counters log
show-networkerror-log	show network errors log
show-poll-interval	Show Poll Interval
show-rate	show the rate of change value for network monitors
show-test-help	Brief help on the usage of the Specific Test referred by Test ID
show-test-result	Show test result
show-test-status	Show test status
show-upper-threshold-info	Show upper threshold info
start-monitor	Start a monitor
start-test	Start a test
stop-monitor	Start a monitor
stop-test	Stop a test

After logging into the CLI, you can start, stop, and query a test from the CLI.

You can view the details of the test by executing `show-test-help` command.

To start the test, you can run `start-test` command with `testId, deviceInstance,` and `arguments` as input to the command. The arguments can be neglected for tests that does not take any arguments as input.

After test execution, the results can be viewed by `show-test-result` command, which displays test result and a raw log generated by that test.

Using CLI, you can list all the InService diagnostics monitors in a device category. By default, all monitors start when ViewCheck application is initialized. You can start and stop any monitor using `start-monitor` and `stop-monitor` commands.

To exit from the **ViewCheck** CLI:

```
BSF(diag-vee0)#exit
BSF(VEE-vee0)#exit
BSF(config)#exit
#exit
```

3.5 Access and Execution of a Test Using XML

You can access ViewCheck XML interface similar to ViewCheck CLI.

1. Establish the secure shell using SSH or Putty.
2. Start the **Telnet** connection from an already established secure shell.

```
telnet localhost 15550

Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
```

3.5.1 Authenticate

After a connection with the XML Agent is established, the only command which should be executed is `Authenticate`. This command is for evaluating user credentials.

The `Authentication` command contains user name and password for verification. Successful authentication is indicated by the success response, otherwise an error message is returned.

You need to send the user credentials to XML Agent using the command `Authenticate`. The XML Agent forwards the authentication request to the BSF system which validates the user credentials and allows the XML client to access it.

- **Request:**

  In the below request, the user credentials `Admin` and `Admin` are created by default in the BSF. You need to provide credentials by creating them using the `CreateObject` command as shown in the section *GetClassList on page 27*.

```
<Script><Authenticate><User>Admin</User><Password>Admin</P
assword></Authenticate></Script>]]>]]>
```

- **Response:**

  The below response is received if the authentication is successful.

  ```
  <?xml version="1.0"?><Response><Success/></Response>]]>]]>
  ```

  The following response is received if authentication fails.

  ```
  <?xml
  version="1.0"?><Response><Error><ErrorCode>1</ErrorCode><T
  ype>Operation not allowed</Type><Description>Authentication
  error</Description><CustomError/></Error></Response>]]>]]>
  ```

3.5.2 Configure

This command configures the XML protocol for the indentation and the events to enable/disable state. There are two configuration options available:

IndentOutput option controls the indentation of the XML response produced by the MINDAgent. Its default value is ’0’, which means, by default, output indentation is off.

- **Request:**

  Below command request sets the indentation to 4:
  <Script><Configure><IndentOutput>4</IndentOutput></Configure></Scri
  pt>]]>]]>

- **Response:**

  ```
  <?xml version="1.0"?>
  <Response>
   <Success/>
   </Response>]]>]]>
  ```

EnableEvents option enables or disables events that reach XML interface from the BSF framework. Successful execution is indicated by the success response. Otherwise, an error message is returned.

The values for the command is true/false. True option enables the events and false option disables the events.

Request:

```
<?xmlversion ="1.0"?>
<Script>
<Configure>
<EnableEvents>true</EnableEvents>
</Configure>
</Script>]]>]]>
```

Response:

```
<?xml
version="1.0"?><Response><Success/></Response>]]>]]>
```

3.5.3 GetClassList

This command is used to retrieve all BSF classes.

- **Request:**
  ```
  <Script ><GetClassList /></Script >]]>]]>
  ```
- **Response:**
  The response message received from the XML Agent is shown below. The response contains all the BSF classes including the BSF framework classes along with the application defined classes.

  ```
  <Response>
      <Classes>
  <Class>session</Class>
  <Class>shell</Class>
  <Class>mode</Class>
  <Class>command</Class>
  <Class>user</Class>
  <Class>group</Class>
  <Class>membership</Class>
  <Class>ACManager</Class>
  <Class>shutdown</Class>
  <Class>usmuser</Class>
  <Class>traphost</Class>
  <Class>cppscript</Class>
  <Class>classlock</Class>
  <Class>globallock</Class>
  <Class>CommunityMO</Class>
  <Class>SystemMO</Class>
  <Class>SNMPAgentConfig</Class>
  <Class>EventReceiver</Class>
  <Class>EventFilter</Class>
  <Class>agent</Class>
  <Class>logsink</Class>
  <Class>logfilter</Class>
  ```

```
<Class>acl</Class>
<Class>aclclass</Class>
<Class>aclmember</Class>
<Class>aclmoid</Class>
<Class>aclclassrange</Class>
<Class>aceclass</Class>
<Class>acemember</Class>
<Class>acemoid</Class>
<Class>aceclassrange</Class>
<Class>virExecEnv</Class>
<Class>interface</Class>
<Class>service</Class>
<Class>diagnostic</Class>
<Class>deviceCategory</Class>
    </Classes>
</Response>]]>]]>
```

NOTICE

ViewCheck software uses classes, diagnostics and device category to start, stop and query the status of tests and monitors.

3.5.4 DescribeClass

`DescribeClass XML` command retrieves the description and properties of a BSF class such as class creatability, class deletability, class writability, attributes, attributes description, attribute types etc. It returns class description on success or an error.

- **Request:**
  The below XML request gets the description of class 'shelf' in the BSF system.

```
<?xmlversion ="1.0"?>
<Script version ="2">
<DescribeClass>
<Class>diagnostic</Class>
</DescribeClass>
</Script>]]>]]>
<?xml version="1.0"?>
```

- **Response:**
  The below XML response shows the different details of the class 'shelf' in the BSF system.

```
<Response>
    <ClassDesc>
<Class>diagnostic</Class>
<Id>1201</Id>
<Description>In Diagnostic mode, user can configure and
perform
Diagnostic related tests, monitoring &amp;amp;
updating.</Description>
<IsSingleton>false</IsSingleton>
<IsCreatable>true</IsCreatable>
<IsDeletable>true</IsDeletable>
<IsWritable>false</IsWritable>
<MOIDDesc>
    <Name>diagnosticMoid</Name>
    <Id>1201</Id>
    <Description/>
    <MOIDItemDesc>
<Ref>
    <ClassId>1102</ClassId>
    <MOIDId>1102</MOIDId>
</Ref>
    </MOIDItemDesc>
</MOIDDesc>
<MethodDesc>
    <Name>hwInventoryList</Name>
    <Id>10</Id>
   <Description>Provides the Inventory of Hardware (Type, Vendor
ID, Major Number, Minor Number and any associated Details) as
detected by the Diagnostics Application. </Description>
    <IsStatic>false</IsStatic>
    <ArgDesc>
<Name>outputResult</Name>
<Id>0</Id>
```

```
<Description/>
<TypeDesc>
    <Name/>
    <Description/>
</TypeDesc>
<IsInput>false</IsInput>
<IsOutput>true</IsOutput>
<IsOptional>false</IsOptional>
    </ArgDesc>
</MethodDesc>
<MethodDesc>
    <Name>reload</Name>
    <Id>20</Id>
    <Description>User can issue this command when there are
updates to the User XML .</Description>
    <IsStatic>false</IsStatic>
    <ArgDesc>
<Name>commandStatus</Name>
<Id>0</Id>
<Description/>
<TypeDesc>
    <Name/>
    <Description/>
</TypeDesc>
<IsInput>false</IsInput>
<IsOutput>true</IsOutput>
<IsOptional>true</IsOptional>
    </ArgDesc>
</MethodDesc>
<MethodDesc>
    <Name>setLogLevel</Name>
    <Id>30</Id>
    <Description>Sets the Log Level of Diagnostic
Application.</Description>
    <IsStatic>false</IsStatic>
    <ArgDesc>
```

```
<Name>logLevelValue</Name>
<Id>0</Id>
<Description>Valid log levels (3-Critical, 2-Normal, 1-
Info)</Description>
<TypeDesc>
    <Name>Integer</Name>
    <Description>Integer number in range (-2^32)/2 to (2^32)/2 -
1</Description>
</TypeDesc>
<IsInput>true</IsInput>
<IsOutput>false</IsOutput>
<IsOptional>false</IsOptional>
    </ArgDesc>
    <ArgDesc>
<Name>commandStatus</Name>
<Id>1</Id>
<Description/>
<TypeDesc>
    <Name/>
    <Description/>
</TypeDesc>
<IsInput>false</IsInput>
<IsOutput>true</IsOutput>
<IsOptional>true</IsOptional>
    </ArgDesc>
</MethodDesc>
<MethodDesc>
    <Name>showLogInfo</Name>
    <Id>40</Id>
    <Description>List all Log files of Diagnostics
Application.</Description>
    <IsStatic>false</IsStatic>
    <ArgDesc>
<Name>outputResult</Name>
<Id>0</Id>
<Description/>
```

```
<TypeDesc>
    <Name/>
    <Description/>
</TypeDesc>
<IsInput>false</IsInput>
<IsOutput>true</IsOutput>
<IsOptional>false</IsOptional>
    </ArgDesc>
</MethodDesc>
<MethodDesc>
    <Name>purgeLog</Name>
    <Id>50</Id>
    <Description>Purge specific log file.</Description>
    <IsStatic>false</IsStatic>
    <ArgDesc>
<Name>logFileName</Name>
<Id>0</Id>
<Description>log file name</Description>
<TypeDesc>
    <Name/>
    <Description>file name</Description>
</TypeDesc>
<IsInput>true</IsInput>
<IsOutput>false</IsOutput>
<IsOptional>false</IsOptional>
    </ArgDesc>
    <ArgDesc>
<Name>commandStatus</Name>
<Id>1</Id>
<Description/>
<TypeDesc>
    <Name/>
    <Description/>
</TypeDesc>
<IsInput>false</IsInput>
<IsOutput>true</IsOutput>
```

```
<IsOptional>true</IsOptional>
    </ArgDesc>
</MethodDesc>
<MethodDesc>
    <Name>purgeAllLog</Name>
    <Id>60</Id>
    <Description>All log files are Zipped and stored
away.</Description>
    <IsStatic>false</IsStatic>
    <ArgDesc>
<Name>commandStatus</Name>
<Id>0</Id>
<Description/>
<TypeDesc>
    <Name/>
    <Description/>
</TypeDesc>
<IsInput>false</IsInput>
<IsOutput>true</IsOutput>
<IsOptional>true</IsOptional>
    </ArgDesc>
</MethodDesc>
<MethodDesc>
    <Name>showSystemErrorLog</Name>
    <Id>70</Id>
    <Description>show system error log</Description>
    <IsStatic>false</IsStatic>
    <ArgDesc>
<Name>outputResult</Name>
<Id>0</Id>
<Description/>
<TypeDesc>
    <Name/>
    <Description/>
</TypeDesc>
<IsInput>false</IsInput>
```

```
<IsOutput>true</IsOutput>
<IsOptional>false</IsOptional>
    </ArgDesc>
</MethodDesc>
    </ClassDesc>
</Response>]]>]]>
```

3.5.5 InvokeMethod

`InvokeMethod` XML command calls the method of an BSF Object. BSF methods are defined with method parameters such as input, output, and input-output.

BSF Object method can be invoked with a list of input or input-output arguments. This command returns a list of output or input-output arguments. The input arguments means it is only an input and will not be displayed in the output. But in the case of input-output arguments, both input and output will be displayed in the output.

- **Request:**
  The below example invokes the method class 'diagnostics' with instance "vee0". Upon execution of this method by the class instance, response will be sent in the output/input-output arguments.

  ```
  Request
  =======
  <Script>
  <InvokeMethod>
  <Object>
  <Class>diagnostic</Class>
  <Name>vee0</Name>
  </Object>
  <Method>setLogLevel</Method>
  <Argument>
  <Name>logLevelValue</Name>
  <Value>1</Value>
  </Argument>
  </InvokeMethod>
  </Script>]]>]]>
  ```

- **Response**:

  `InvokeMethod` command returns the below response on executing the above example command.

```
<?xml version="1.0"?>
<Response>
    <Arguments>
<Argument>
    <Name>commandStatus</Name>
    <Value>Set Log Level Success</Value>
</Argument>
    </Arguments>
</Response>]]>]]>
```

To exit from the ViewCheck XML:

```
<Script>
<Command name="Exit"/>
<Script>]]>]]>
```

3.6 ViewCheck Service LOG Information

ViewCheck service logs are generated in `EmrDiag_Debug.log` file and is located at `/opt/diagnostics/var/log/diag/service/`

The Test result logs and raw logs generated by various tests are available at `/opt/diagnostics/var/log/diag/testlog/`

3.7 ViewCheck TestLog Information

ViewCheckinternally retains data related to tests invoked by the user. If the number of tests invoked by the user exceed 1000, all this information is saved in the `Emr_TestResultsMib.txt` file and the internal storage is erased.

Commands Execution

4.1 Overview

CLI and XML are the primary ways to access ViewCheck capabilities on the blade. These mechanisms allows you to perform activities such as start, stop and query on the ViewCheck software. Using CLI and XML you can also set the parameters for monitoring.

Commands are classified into:

- Generic commands
- Commands for test management and control
- Commands for monitoring

4.2 Generic Commands

Following are the general functional commands provided by the ViewCheck service.

NOTICE

The command syntaxes for XML interface is given for the diagnostic class with instance as vee0. This instance varies depending on the blade.

4.2.1 configure-error-strings

`configure-error-strings` command allows to add user-defined kernel error strings to the diagnostics database.

Syntax for CLI

```
configure-error-strings errorStrings <string>
```

The following table provides the `configure-error-strings` command arguments.

Table 4-1 configure-error-strings

Argument	Data Type	Description
string	String	Kernel error string to be added to the diagnostics database.

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
```

```
<Name>vee0</Name>
</Object>
<Method>configureErrStrings</Method>
<Argument>
<Name>Error String</Name>
<Value>{error String}</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

4.2.2 configure-log-path

`configure-log-path` command allows to configure the location of diagnostics logs. You can also specify the maximum size of logs. Once the log size reaches the user-defined limit, a trap is sent to the user. Specifying log size is optional and by default, its value is 1GB.

Syntax for CLI

```
configure-log-path logpath <PATH> logSize <size>
```

The following table provides the `configure-log-path` command arguments.

Table 4-2 configure-log-path

Argument	Data Type	Description
logpath	String	Location of the log path where diagnostics should place the log files.
size	String	Optional parameter, size of the log path. For example, 10M, 2G, 100K, 1048576.

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>configureLogPath</Method>
<Argument>
<Name>logpath</Name>
<Value>{logPath String}</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

4.2.3 device-category

`device-category` command allows to configure the available device categories.

Syntax

```
device-category <Dev category>
```

Expected Output

The `CLI prompt will show the device category that you have selected.`

NOTICE
This command is valid only in CLI.

4.2.4 diag-service

diag-service command allows to start/stop/restart/status the diagnostics service.

Syntax for CLI

```
diag-service operation <restart/start/stop/status>
```

The following table provides the `diag-service` command arguments.

Table 4-3 diag-service

Argument	Data Type	Description
operation	String	Requested operation to the diagnostics core.

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>diagService</Method>
<Argument>
<Name>operation</Name>
<Value>{operation String}</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

4.2.5 hw-inventory-list

`hw-inventory-list` command provides the detailed information of hardware components available on the blade. The command displays the Hardware Type, Vendor ID, Major Number, Minor Number and any associated details that are identified by the ViewCheck.

Syntax for CLI

```
hw-inventory-list
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>hwInventoryList</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.2.6 list-all-tests

`list-all-tests` command lists all the tests available on the board.

Syntax for CLI

```
list-all-tests
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>showListAllTests</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.2.7 list-device-instances

`list-device-instances` command lists all possible device instances in present device category.

Syntax for CLI

```
list-device-instances
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>listInstances</Method>
</InvokeMethod>
</Script>]]>]]>
```

NOTICE

This command will be deprecated in future.

4.2.8 purge-all-log

`purge-all-log` command allows to zip all log files and store away.

Syntax for CLI

```
purge-all-log
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>SP_VEE0vee0</Name>
</Object>
<Method>purgeAllLog</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.2.9 purge-log

purge-log command deletes the LOG files generated by ViewCheck software.

Syntax for CLI

```
purge-log logFileName <logfile Name>
```

The following table provides the purge-log command arguments.

Table 4-4 purge-log

Argument	Data Type	Description
logfile Name	String	Type the name of the Log file that you want to delete or clear

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>purgeLog</Method>
<Argument>
<Name>logFileName</Name>
<Value>{Name of the log file}</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

Example

```
BSF(diag-vee0)#purge-log logFileName NETWORK_base1.log
Purge Log Success
```

NOTICE

You can purge only raw log files. The raw log file naming convention is in the form of *.log.

4.2.10 set-log-level

`set-log-level` command sets the current log level of ViewCheck to value X. This is an internal debug command to be used mainly for generating detailed debug log information. The valid log level values are listed in the following table:

Table 4-5 set-log-level Values

Values	Description
1-Info	All logs are logged. Even functions like entry and exit are also logged.
2- Normal	Details of function flows are logged.
3- Critical	High level errors are logged.

Syntax for CLI

```
set-log-level logLevelValue <x>
```

The following table lists the `set-log-level` command arguments.

Table 4-6 set-log-level

Argument	Data Type	Description
X	Integer	Possible values are 3, 2, 1 (3-Critical, 2-Normal, 1-Info)

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>setLogLevel</Method>
<Argument>
<Name>logLevelValue</Name>
<Value>1</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

Example

```
BSF(diag-vee0)#set-log-level logLevelValue 2
Set Log Level Success
```

4.2.11 show-all-monitors

`show-all-monitors` command lists all the monitors available on the board.

Syntax for CLI

```
show-all-monitors
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>showAllMonitors</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.2.12 show-all-test-status

`show-all-test-status` command allows you to view the current status of all tests available on the blade.

Syntax for CLI

```
show-all-test-status
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>ShowAllTestStatus</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.2.13 show-diag-scan-result

`show-diag-scan-result` command shows result of the last `diag-scan` command.

Syntax for CLI

```
show-diag-scan-result
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>showDiagScanResult</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.2.14 show-log-content

`show-log-content` command displays the content of the mentioned log file.

Syntax for CLI

```
show-log-content logfilename <logfile Name>
```

The following table provides the show-log-content command arguments.

Table 4-7 show-log-content

Argument	Data Type	Description
logfile Name	String	Name of the log file to be displayed.

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>showLogContent</Method>
<Argument>
<Name>logFileName</Name>
```

```
<Value>{logfilename String}</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

4.2.15 show-log-info

`show-log-info` command provides the details of the various LOG files along with the diagnostics data.

Syntax for CLI

```
show-log-info
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>showLogInfo</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.2.16 show-log-level

`show-log-level` command displays the current logging level of ViewCheck.

Syntax for CLI

```
show-log-level
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>showLogLevel</Method>
</InvokeMethod>
</Script>]]>]]>]
```

4.2.17 show-running-mode

`show-running-mode` command displays the running mode of the ViewCheck (INSM or OOSD).

Syntax for CLI

```
show-running-mode
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>showRunningMode</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.2.18 show-systemerror-log

`show-systemerror-log` command displays the kernel critical and error messages captured by ViewCheck application.

Syntax for CLI

```
show-systemerror-log
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>showSystemErrorLog</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.2.19 start-all-monitors

`start-all-monitors` command starts all the monitors of the mentioned device category.

Syntax for CLI

`start-all-monitors device-category <Dev category>`

The following table provides the `start-all-monitors` command arguments.

Table 4-8 start-all-monitors

Argument	Data Type	Description
Dev category	String	Name of the device category.

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>startAllMonitors</Method>
<Argument>
<Name>device-category</Name>
<Value>{Dev category String}</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

4.2.20 start-diag-scan

`start-diag-scan` starts the diag-scan on the mention device categories.

Syntax for CLI

```
start-diag-scan deviceCategory <Dev category-1>,…,<Dev category-N>
Iterations <Itr-num> haltOnError <halt-string> timeout <timeout-
val>
```

`Iterations`, `haltonError`, and `timout` are optional arguments.

The following table provides the `start-diag-scan` command arguments.

Table 4-9 start-diag-scan

Argument	Data Type	Description
Dev category-N	String	Name of the device category. You can specify multiple device category using comma ',' in between.
Itr-num	Integer	Enter the number of times that diag-scan has to run. By default value is "1". Maximum number of iterations that user can specify is 1000.
halt-string	String	Type "Yes" or "No". `haltOnError` specifies whether to continue or stop with test case execution on the occurrence of any error. By default, value is "No".
timeout-val	Integer	Enter the maximum time period to be taken by each test to execute.

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>startDiagScan</Method>
<Argument>
<Name>device-category</Name>
<Value>{Dev category String}</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

4.2.21 stop-all-monitors

`stop-all-monitors` command stops all the monitors of the mentioned device category.

Syntax for CLI

```
stop-all-monitors device-category <Dev category>
```

The following table provides the `stop-all-monitors` command arguments.

Table 4-10 stop-all-monitors

Argument	Data Type	Description
Dev category	String	Name of the device category.

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>stopAllMonitors</Method>
<Argument>
<Name>device-category</Name>
<Value>{Dev category String}</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

4.2.22 stop-diag-scan

`stop-diag-scan` stops the currently running diag-scan command.

Syntax for CLI

```
stop-diag-scan
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>stopDiagScan</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.2.23 switch-mode

`switch-mode` command allows to switch ViewCheck from OOS mode to InService mode and vice versa.

Syntax for CLI

```
switch-mode modeVal <x>
```

The following table lists the `switch-mode` command arguments..

Table 4-11 switch-mode

Argument	Data Type	Description
X	String	Possible values are `insm` and `oosd`. Parameter given for the command is case-insensitive.

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>switchMode</Method>
<Argument>
<Name>modeVal</Name>
<Value>oosd</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

Example

```
BSF(diag-vee0)#switch-mode modeVal insm
Mode of ViewCheck is successfully changed to INSM
```

4.2.24 version

`version` command displays the RPM versions of all the ViewCheck packages installed.

Syntax for CLI

```
version
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>diagnostic</Class>
<Name>vee0</Name>
</Object>
<Method>showVersion</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.3 Test Management and Control Commands

This section describes the CLI commands used for test management and control of diagnostics tests.

NOTICE

Execute all test management and control commands only after entering a specific device category.

4.3.1 start-test

`start-test` command allows you to start and run a particular diagnostic test.

Syntax for CLI

```
start-test testId <Test ID> deviceInstance <Dev Instance> arguments
-t <timeout-val> -Iterations <Itr-num> -Halt-onerror <halt-string>
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>startTest</Method>
<Argument>
<Name>deviceInstance</Name>
<Value>Dev Instance</Value>
</Argument>
```

```
<Argument>
<Name>testId</Name>
<Value>Test ID</Value>
</Argument>
<Argument>
<Name>arguments</Name>
<Value>Param1</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

`Iterations`, `haltonError`, and `timout` are optional arguments.

The following table lists the `start-test` command arguments.

Table 4-12 start-test

Argument	Data Type	Description
Test ID	Integer	Type the unique ID of a particular test that you want to start
Dev Instance	Enum	Use `list-device-instances` CLI command to get the equivalent Enumerated value of supported Device Instance
timeout-val	Integer	Timeout value for the test.
Itr-num	Integer	Number of iterations of the test.
halt-string	String	"Yes" or "No" value. if "Yes", the test halts on error. if "No", the test does not halts.

4.3.2 show-test-help

`show-test-help` command provides the brief information on how to use a particular test. This command provides information such as how to start, stop, and query the specified test ID.

Syntax for CLI

```
show-test-help testId <Test ID>
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{dev Cat}</Name>
</Object>
```

```
<Method>showTestHelp</Method>
<Argument>
<Name>testId</Name>
<Value>Test ID</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

The following table lists the `show-test-help` command arguments.

Table 4-13 show-test-help

Argument	Data Type	Description
Test ID	Integer	Type the unique ID of a particular test to get the details of it

4.3.3 list-tests

`list-tests` command provides a supported list of diagnostics tests on the blade with associated test IDs.

Syntax for CLI

```
list-tests
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{dev Cat}</Name>
</Object>
<Method>showTestList</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.3.4 show-test-status

`show-test-status` command allows you to view the status of a particular test. The status of a test can be In Progress, Test Execution Completed, Test Stopped, and Test Timed Out.

Syntax for CLI

```
show-test-status testId <Test ID> deviceInstance <Dev Instance>
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>showTestStatus</Method>
<Argument>
<Name>deviceInstance</Name>
<Value>Dev Instance</Value>
</Argument>
<Argument>
<Name>testId</Name>
<Value>Test ID</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

The following table lists the `show-test-status` command arguments.

Table 4-14 show-test-status

Argument	Data Type	Description
Test ID	Integer	Type the unique ID of a particular test that you want to view the status
Dev Instance	Enum	Use list-device-instances CLI command to get the equivalent enumerated value of supported Device Instance

4.3.5 show-test-result

`show-test-result` command allows you to view the latest result of a particular test. This command displays the start and end time of the test, the test status such as Passed, Failed, Aborted, and Timed Out, and additional test arguments.

Syntax for CLI

```
show-test-result testId <Test ID> deviceInstance <Dev Instance>
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>showTestResult</Method>
<Argument>
<Name>deviceInstance</Name>
<Value>Dev Instance</Value>
</Argument>
<Argument>
<Name>testId</Name>
<Value>Test ID</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

Table 4-15 show-test-result

Argument	Data Type	Description
Test ID	Integer	Type the unique ID of a particular test that you want to view the latest result
Dev Instance	Enum	Use list-device-instances CLI command to get the equivalent enumerated value of supported Device Instance

4.3.6 stop-test

`stop-test` command allows you to stop or cancel any running diagnostic test.

Syntax for CLI

```
stop-test testId <Test ID>  deviceInstance <Dev Instance>
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>stopTest</Method>
<Argument>
<Name>deviceInstance</Name>
<Value>Dev Instance</Value>
</Argument>
<Argument>
<Name>testId</Name>
<Value>Test ID</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

The following table provides the `Stop-test` command arguments.

Table 4-16 stop-test

Argument	Data Type	Description
Test ID	Integer	Type the unique ID of the Test that you want to stop
Dev Instance	Enum	Use `list-device-instances` CLI command to get the equivalent enumerated value of supported Device Instance

4.4 Monitoring Commands

The following list of commands are used for management of the InService Monitoring functionality of ViewCheck.

NOTICE

Execute all commands related to monitors only after entering the specific device category mode.

4.4.1 show-monitor-id

`show-monitor-id` command displays the list of parameters that are monitored using the InService Diagnostic. This command displays default monitor ID values of the parameters also.

Syntax for CLI

```
show-monitor-id
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>showMonitorId</Method>
</InvokeMethod>
</Script>]]>]]>
```

NOTICE

This command displays a list of monitors, if supported, only in that particular device category.

4.4.2 show-poll-interval

`show-poll-interval` command displays list of default Poll intervals that are associated with the monitors. The Poll interval values are in seconds.

Syntax for CLI

```
show-poll-interval monitorId <Monitor ID> deviceInstance <Dev
Instance>
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>showPollInterval</Method>
<Argument>
<Name>deviceInstance</Name>
<Value>Dev Instance</Value>
</Argument>
<Argument>
<Name>monitorId</Name>
<Value>Monitor ID</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

The following table provides the `show-poll-interval` command arguments.

Table 4-17 show-poll-interval

Argument	Data Type	Description
Monitor ID	Integer	Type the unique ID of the Monitor of which you want to view its default Poll interval values, if any
Dev Instance	Enum	Use `list-device-instances` CLI command to get the equivalent enumerated value of supported Device Instance

4.4.3 start-monitor

`start-monitor` command allows you to trigger a specific monitor to start monitoring if it is not already initiated by default.

Syntax for CLI

```
start-monitor monitorId <Monitor ID> deviceInstance <Dev Instance>
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>startMonitor</Method>
<Argument>
<Name>deviceInstance</Name>
<Value>Dev Instance</Value>
</Argument>
<Argument>
<Name>monitorId</Name>
<Value>Monitor ID</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

The following table provides the `Start-monitor` command arguments.

Table 4-18 start-monitor

Argument	Data Type	Description
Monitor ID	Integer	Type the unique ID of the Monitor that you want to start
Dev Instance	Enum	Use list-device-instances CLI command to get the equivalent enumerated value of supported Device Instance

4.4.4 set-poll-interval

`set-poll-interval` command is used to set the Poll interval value for monitors.

Syntax for CLI

```
set-poll-interval monitorId <Monitor ID> deviceInstance <Dev
Instance> pollInterval <PollIntervaLvalue>
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>setPollInterval</Method>
<Argument>
<Name>deviceInstance</Name>
<Value>Dev Instance</Value>
</Argument>
<Argument>
<Name>monitorId</Name>
<Value>Monitor ID</Value>
</Argument>
<Argument>
<Name>pollInterval</Name>
<Value>PollInterval Value</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

The following table provides the `set-poll-interval` command arguments.

Table 4-19 set-poll-interval

Argument	Data Type	Description
Monitor ID	Integer	Type the unique ID of the Monitor for which you want to set the Poll interval value
Dev Instance	Enum	Use `list-device-instances` CLI command to get the equivalent enumerated value of supported Device Instance
PollInterval Value	Integer	Type the Poll interval value for the specified monitor ID

4.4.5 set-poll-interval-default

`set-poll-interval-default` command is used to reset the Poll interval value to default value of a particular monitor.

Syntax for CLI

```
set-poll-interval-default monitorId <Monitor ID> deviceInstance
<Dev Instance>
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>setPollIntervalDefault</Method>
<Argument>
<Name>deviceInstance</Name>
<Value>Dev Instance</Value>
</Argument>
<Argument>
<Name>monitorId</Name>
<Value>Monitor ID</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

The following table provides the `set-poll-interval-default` command arguments.

Table 4-20 set-poll-interval-default

Argument	Data Type	Description
Monitor ID	Integer	Type the unique ID of the Monitor for which you want to reset the Poll interval value to default value
Dev Instance	Enum	Use `list-device-instances` command to get the equivalent enumerated value of supported device instance

4.4.6 stop-monitor

`stop-monitor` command allows you to trigger a specific monitor to stop monitoring, which is already in service.

Syntax for CLI

```
stop-monitor monitorId <Monitor ID> deviceInstance <Dev Instance>
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>stopMonitor</Method>
<Argument>
<Name>deviceInstance</Name>
<Value>Dev Instance</Value>
</Argument>
<Argument>
<Name>monitorId</Name>
<Value>Monitor ID</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

The following table provides the `stop-monitor` command arguments.

Table 4-21 stop-monitor

Argument	Data Type	Description
Monitor ID	Integer	Type the unique ID of the Monitor that you want to stop monitoring
Dev Instance	Enum	Use `list-device-instances` command to get the equivalent enumerated value of supported device instance

4.4.7 set-rate

`set-rate` command sets the rate of change value for network monitors.

Syntax for CLI

```
set-rate monitorId <Monitor Id> deviceInstance <Dev Instance> rate
<Rate>
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>setRate</Method>
<Argument>
<Name>deviceInstance</Name>
<Value>Dev Instance</Value>
</Argument>
<Argument>
<Name>monitorId</Name>
<Value>Monitor ID</Value>
</Argument>
<Argument>
<Name>rate</Name>
<Value>Rate</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

The following table provides the `set-rate` command arguments.

Table 4-22 set-rate

Argument	Data Type	Description
Monitor ID	Integer	Type the unique ID of the Monitor for which rate of change value is to be set
Dev Instance	Enum	Use `list-device-instances` command to get the equivalent enumerated value of supported device dnstance
Rate	Integer	Rate of change value

4.4.8 show-rate

`show-rate` command shows the rate of change value of network monitors.

Syntax for CLI

```
show-rate monitorId <Monitor Id> deviceInstance <Dev Instance>
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>showRate</Method>
<Argument>
<Name>deviceInstance</Name>
<Value>Dev Instance</Value>
</Argument>
<Argument>
<Name>monitorId</Name>
<Value>Monitor ID</Value>
</Argument>
</InvokeMethod>
</Script>]]>]]>
```

The following table provides the `show-rate` command arguments.

Table 4-23 show-rate

Argument	Data Type	Description
Monitor ID	Integer	Type the unique ID of the Monitor of which rate of change value is to be shown
Dev Instance	Enum	Use `list-device-instances` command to get the equivalent enumerated value of supported device instance

4.4.9 show-networkcounter-log

`show-networkcounter-log` command displays the list of network counters on devices that have crossed the maximum rate value. For more information on command `rate`, refer *show-rate on page 65*.

Syntax for CLI

```
show-networkcounter-log
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev Cat}</Name>
</Object>
<Method>showNetworkCounterLog</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.4.10 show-networkerror-log

`show-networkerror-log` command displays the list of network counters on devices that have crossed the maximum rate value. For more information on command `rate`, refer *sshow-rate on page 65*.

Syntax for CLI

```
show-networkerror-log
```

Syntax for XML

```
<Script>
<InvokeMethod>
<Object>
<Class>deviceCategory</Class>
<Name>vee0-{Dev cat}</Name>
</Object>
<Method>showNetworkErrorLog</Method>
</InvokeMethod>
</Script>]]>]]>
```

4.4.11 exit

`exit` command allows you to exit from the ViewCheck CLI.

Syntax for CLI

```
exit
```

Syntax for XML

```
<Script>
<Command name="Exit"/>
<Script>]]>]]>
```

4.5 InService Monitoring Specifics

The In Service Monitoring of ViewCheck provides functionality to:

- Monitor HW Device Status
- Monitor Device Critical Errors

4.5.1 Monitoring HW Device Status

InService Monitoring also periodically polls the following for various devices status on the blade:

- Storage Device HDD Health Status
- Network Devices Counters Statistics
- Network Devices Error Statistics

Each of the above is identified by a unique Monitor ID. For more information on Monitor ID, refer *Monitor ID on page 16*.

The default Poll Interval for monitoring each of the Monitor ID is set to 10 seconds. Commands are provided to edit the default settings.

For more information on the commands provided for InService Monitoring, refer *Monitoring Commands on page 58*.

The following table provides list of Monitor IDs and Device Instances for each of the Monitors being monitored on ATCA-F140.

Table 4-24 Monitors

Monitor Description	Monitor ID	Valid Device Instances	Remarks
HDD Health Status	1010	Sda1 to Sda8 (83 to 9)	Monitors the health status of the sda1 to sda8 partitions on the HDD and reports.
Network Errors	1020	eth0(62),eth1(63)	Monitors the various error counters for each of the Network Device Instances and provides an error counter exceeds the rate of change.
Network Statistics	1021	eth0(62),eth1(63)	Monitors the various counters for each of the Network Device Instances

4.5.2 Monitoring Device Critical Errors

Under Linux OS, the Device Drivers log abnormal behavior and potential errors occurring in the hardware device with `KERN_ERR` or `KERN_CRIT` category. These notifications are considered as potential errors as they could manifest into latent faults in the live system.

As part of monitoring the Device Critical errors, all such Kernel CRITICAL and Kernel ERROR notifications have been extracted from the PNE Kernel (3.0.3) driver sources and represented in the form of a database.

The InService Monitoring Module of ViewCheck looks for the occurrence of these notifications and on detection sends a notification to XML.

The device errors are captured and are identified uniquely with their ERROR IDs. For definition of ERROR ID, refer *Error ID on page 16*.

4.6 OOSD Tests

OOSD Tests are used to monitor and manage the performance of the hardware components of blades. You can execute these tests only when blades are offline, that is blades are not providing any service.

4.6.1 CPU

- **CpuBurnTest:** This test constantly cycles FPU intensive functions.The resultant calculations are constantly checked for data integrity. If the test detects erroneous data, the test fails.
- **CpuBenchMark:** Tests different arithmetic operations and gives the results.

4.6.2 HDD

- **DiskBadBlksTest:** Searches for bad blocks on a device (usually a disk partition).
- **DiskCntlrTest:** Verifies the random data written to disks remains unchanged. It verifies only IDE and SCSI controllers that are associated with mounted file systems. Disk controllers associated with read-only mounted file systems are not verified.

4.6.3 Memory

- **MemCntlrTest:** Randomly writes to areas of memory, then reads the memory back to ensure the written values remain unchanged.
- **RandomMemoryTest:** Performs stress testing on the memory subsystem. This test is effective in finding intermittent and non-deterministic faults. The problems in other hardware areas such as overheating CPU, out-of-specification power supply, and so on can cause memory faults.
- **EdacErrorStatsTest:** Checks for the occurrence of one bit and two bit errors in the RAM.
- **MemBandwidth:** Measures the ability to copy, read, and write data over a varying set of sizes.
- **MemLatency:** Measures the time taken by the memory to respond with the data for readrequest.

4.6.4 OS

- **MemSepTest:** Ensures that user space programs cannot read and write to areas of memory utilized by items such as Video RAM and kernel code.
- **SupervisorInstrTest:** Ensures that the enforcement of the property that privileged instructions should only be in supervisor mode is still in effect. The set of privileged instructions tested to confirm this is architecture dependent.
- **DmesgCheckTest:** Checks for user-given keywords in the kernel dmesg logs.

4.6.5 PCI

- **PCIScanTest:** Enumerates all active PCI devices in the blade and ensures board default configuration is active. This test need to be executed under supervisory mode.
- **PCILinkTest:** Scans for device ID and speed of on-board PCI devices.

4.6.6 IPMC

- **IPMITest:** Executes basic IPMI commands and ensures the basic IPMI controller, IPMI bus functionalities are working properly by reading the response back.
- **IPMCSENSOREVENTTest**: Verifies IPMC Event generation.
- **BMCWatchdogTest:** Verifies working of local BMC watchdog timer.

4.6.7 NETWORK

- **EthLinkTest:** Verifies Ethernet device (for example: eth0, eth1, eth2, etc...) link status (active/inactive) and also captures various statistics of Ethernet device.
- **FloodPingTest:** Uses the ICMP protocol's mandatory `ECHO_REQUEST` datagram to elicit an `ICMP_ECHO_RESPONSE` from a host or gateway.
- **EthStatsTest:** Tests basic network packet consistency, that is checks network Rx, Tx errors, generates a warning if errors are less than or equal to 100 and marks the test as fail if number of errors are more than 100.
- **NetworkCntlrTest:** Verifies random data transmitted and also the data received for each configured network device. It verifies only Ethernet and token ring devices that are configured and active. The asynchronous devices are not verified.
- **BcmRegReadTest:** Reads error reporting registers of various modules (like DMA, I2c, CMIC etc.) in BCM switch and reports major, minor errors occurred in them and also ensures that switch is configured to defaults.
- **BcmPCIComplianceTest:** This test does read/write operations on PCI configuration space register and ensures that there are no read/write errors during access of PCI configuration space. Failure of this test may indicate faulty switch.
- **BcmCMICTest:** This test does writes and verifies patterns in CMIC bus interface S-channel message registers and ensures that there are no read/write errors during access CMIC interface. Failure of this test may indicate faulty switch.
- **BcmCounterRdTest:** Tests the width (in bits) of all BCM counters against the width specified in the register manual and ensures that all counters are configured properly and accessible. Failure of this test may indicate faulty switch.

- **BcmCounterRdWrTest:** Writes different patterns in to BCM counter registers and verifies the same, ensuring that there are read/write errors during register access. Failure of this test may indicate faulty switch.
- **BcmLinkscanTest:** Enumerates all phy-device (ports) ID in a loop and verifies that it does not change. Failure of this test may indicate faulty port/switch.
- **BcmTXSpeedTest:** Provides a fairly accurate estimate of packet transmission speed (bytes/sec) by calculating the time difference between DMA start and DMA end. Failure of this test may indicate a faulty switch.
- **BcmDMATableVerify:** Verifies DMA table entries and ensures that table is populated properly. Failure of this test may indicate a faulty DMA module in the BCM switch.
- **SfpScanTest:** Enumerates all SFP modules connected to SFP ports of blade and their operational speeds.
- **SfpDetectionTest:** Enumerates all SFP modules connected to SFP ports of blade and their operational speeds.
- **NetworkThroughputServ:** Starts the server for network throughput testing.
- **NetworkThroughput:** Tests the network throughput.

NOTICE

Any of the BCM tests could result in random failures when executed in stress testing.

4.6.8 I2C

- **I2CProbeTest:** Mod-probes an I2c module and checks for default I2c devices on board.

4.6.9 USB

- USB

4.6.10 Flash

NorFlashTest: This test checks for the sanity of NorFlash by comparing the written test data with the data read from the device.

4.6.11 RTC

- **RTCText:** Checks for proper functioning of real time clock.

Related Documentation

A.1 SMART Embedded Computing Documentation

The documentation listed is referenced in this manual. Technical documentation can be found by using the Documentation Search at *https://www.smartembedded.com/ec/support/* or you can obtain electronic copies of SMART EC documentation by contacting your local sales representative.

Table A-1 SMART EC Documentation

Document Title	Document Number
ATCA-F140 Data Sheet	ATCA-F140-DS
ATCA-F140 BBS Programmer's Reference	6806800N23M

Related Documentation