User Guide

Pigeon Point Chassis Manager

Release 1.3.0

August 05, 2022

nVent

Schroff GmbH

hardware.management@nVent.com

www.pigeonpoint.com

schroff.nVent.com

This document is furnished under license and may be used or copied only in accordance with the terms of such license. The content of this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment bynVent. nVent assumes no responsibility or liability for any errors or inaccuracies that may appear in this book.

Except as permitted by such license, no part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, manual, recording, or otherwise, without the prior written permission of nVent.

The Pigeon Point Chassis Manager uses an implementation of the MD5 Message-Digest algorithm that is derived from the RSA Data Security, Inc. MD5 Message-Digest algorithm.

All nVent marks and logos are owned or licensed by nVent Services GmbH or its affiliates. All other trademarks are the property of their respective owners. nVent reserves the right to change specifications without notice.

Table of contents

1 About this document 7

1.1 Chassis Manager documentation 7

1.1.1 Conventions used in this document 7

1.2 Additional resources 8

2 Introduction 9

2.1 In this section 9

2.2 Intelligent Platform Management: a VITA 46.11 overview 9

2.3 Pigeon Point Board Management Reference: hardware and firmware 11

2.4 Pigeon Point Chassis Manager and ChMM 11

2.4.1 Chassis Manager compatibility with the Pigeon Point Shelf Manager 12

2.4.2 Pigeon Point Chassis Manager features 13

2.4.3 Support for dual redundant operation 14

2.4.4 System Manager Interface 16

3 Configuration 18

3.1 In this section 18

3.2 Setting up U-Boot 18

3.2.1 U-Boot environment variables 20

3.2.2 Assigning values to environment variables 28

3.2.3 Configuring U-Boot environment variables for the Chassis Manager 28

3.2.4 Establishing the secondary RC script 30

3.3 Setting up Chassis Manager configuration file 30

3.3.1 Carrier-specific configuration file 85

3.3.2 Obtaining configuration variables from Chassis FRU Information 85

3.3.3 Verbosity level description 86

3.3.4 Verbosity console level description 87

3.4 Setting up Ethernet 87

3.4.1 Usage of the first Ethernet interface 88

3.4.2 Usage of the second Ethernet interface 89

3.4.3 Gigabit Ethernet Support 93

3.4.4 Using the alternate software redundancy interface 94

3.4.5 Changing the default ChMM network parameters 96

3.4.6 Assigning VLAN IDs 104

3.4.7 Assigning IP addresses to the Chassis Manager via DHCP (version 4) 105

3.4.8 Using IPv6 for chassis-external access to the Chassis Manager 112
3.5 Configuring the FRU Information 114
3.5.1 Accessing the Chassis FRU Information 114
3.5.2 Setting up the Chassis FRU Information 116
3.5.3 Setting up the Chassis FRU Information using the CLI 119
3.5.4 Other FRU Information repositories 120
3.6 Configuring carrier and chassis attributes using HPDL 120
3.6.1 Compiling HPDL definitions 121
3.6.2 Compiling SDRs 121
3.6.3 Deploying HPDL data to the ChMM file system 122
3.6.4 Deploying HPDL data to FRU Information areas 122
3.7 Configuring the cooling management strategy 126
3.7.1 Default cooling management strategy 126
3.7.2 Configuring a specific cooling management strategy 127
3.8 Configuring local sensors 128
3.9 Setting the Auxiliary Firmware Revision 133
3.10 Setting up the clock 134
3.10.1 Obtaining date and time from a time server 135
3.11 Setting up and using ChMM Power On Self Tests 136
3.12 Configuring external event handling 137
3.12.1 Detailed steps to configure external event handling 138
3.12.2 External event handler operation 140
3.13 Configuring the Platform Event Trap format 141
3.13.1 Parsed example of SNMP Trap 143
3.14 Configuring system services 144
3.15 Configuring the speed of I^2C buses 147
3.16 Configuring VPX Control/GPIO signals 148
4 Using the Chassis Manager 150
4.1 In this section 150
4.2 ChMM login 150
4.3 Starting the Chassis Manager 150
4.4 Starting SNMP and HTTP daemons manually 192
4.5 Redundant operation 192
4.5.1 Initialization of the redundant Chassis Managers 195
4.5.2 Redundancy and HWRI State sensor 195

4.5.3 Reboot Reason sensor 198
4.6 Operation in the chassis with radial System IPMB 199
4.6.1 Upgrading the routing element SmartFusion cSoC(s) 201
4.7 Automatic SEL truncation 202
4.8 Cooling state sensors 203
4.9 Deadlock detection 205
4.10 I^2C bus fault isolation 208
4.11 Assignment of LAN configuration parameters to boards and modules 210
4.11.1 Structure and composition of the supported parameters 210
4.11.2 Obtaining LAN configuration parameters on the Chassis Manager level 211
4.11.3 Dispatching LAN configuration parameters to boards 215
4.12 ChMM tests available via the diagnostic infrastructure 215
5 Using the IPMI analysis tools 219
5.1 In this section 219
5.2 IPMB trace collection daemon (ipmb_traced) 219
5.3 IPMI trace analyzer 223
5.3.1 Introduction to the Wireshark GUI 223
5.3.2 Introduction to terminal-oriented Wireshark 226
5.3.3 Installing IPMI analyzer software 226
5.4 Collecting and analyzing a trace using the GUI and command line tools 226
5.4.1 Collecting IPMI traces on System IPMB in unattended mode 226
5.4.2 Collecting IPMI traces on System IPMB in controlled mode 233
5.4.3 Collecting traces of IPMI traffic over the network 235
5.4.4 Analyzing a trace using the GUI 238
5.4.5 Analyzing a trace using command-line tools 241
6 Re-initializing the ChMM 243
6.1 In this section 243
6.2 Re-initializing the U-Boot environment 243
6.3 Re-initializing the file system 244
6.4 Resetting the login password 245
7 Re-programming the ChMM-700 246
7.1 In this section 246
7.2 Firmware reliable upgrade procedure overview 246
7.3 Flash partitioning and image layout 248

7.4 Changing the 'user' flash partition format to JFFS2 250
7.5 Reliable upgrade procedure status file 253
7.6 Reliable upgrade utility 254
7.7 Reliable upgrade of SmartFusion cSoC image 261
7.8 Reliable upgrade utility use scenarios 263
7.9 Finding out component versions via the imls utility 264
7.10 Reliable upgrade examples 266
7.10.1 Example 1 266
7.10.2 Example 2 274
7.10.3 Example 3 278
7.10.4 Example 4 287
8 Customer support 294
9 Revision History 295
9.1 Release 1.0.0 295
9.2 Release 1.0.1 295
9.3 Release 1.1.0 295
9.4 Revision 1.2.0 296
9.5 Revision 1.2.1 297
9.6 Revision 1.2.2 297
9.7 Revision 1.3.0 297

List of figures

Figure 1: Management aspects and potential Pigeon Point product sites in an example VITA 46.11 chassis 10
Figure 2: Pigeon Point Chassis Manager architecture and context 15
Figure 3: Implementation options for ChMC cross-connects 92
Figure 4: Dual star radial System IPMB topology 200
Figure 5: Redundant dual star radial System IPMB topology 200
Figure 6: Main window (IPMB trace analysis) 223
Figure 7: Main window (IPMI over network trace analysis) 225
Figure 8: Open Capture File window 228
Figure 9: GUI Main window (unattended mode) 229
Figure 10: Capture Options window 233
Figure 11: Collected trace (controlled mode) 234
Figure 12: Capture Options window (analyzing networked IPMI traffic) 236
Figure 13: Collected trace (networked IPMI traffic) 237
Figure 14: Message 244 is selected 238

Figure 15: IPMI protocol layers (IPMB trace) 239
Figure 16: Filtered trace 240

List of tables

Table 1: Chassis Manager documentation 7
Table 2: Conventions used in this document 7
Table 3: U-Boot environment variables 20
Table 4: Configuration parameter types and descriptions 31
Table 5: Chassis Manager configuration parameters 33
Table 6: U-Boot environment variables and descriptions 84
Table 7: Verbosity configuration parameters and levels 87
Table 8: HPM.3 DHCP Client ID 107
Table 9: HPM.3 DHCP Unique Identifier format 107
Table 10: Slot Location record for VITA 46.11 108
Table 11: Chassis FRU Information and configuration variable settings 116
Table 12: Network services 145
Table 13: OS level bus number mapping for ChMM-700 148
Table 14: Format of event message 196
Table 15: Cooling state sensors 203
Table 16: ChMM tests implemented in the diagnostic infrastructure 216
Table 17: Options for the IPMB trace collection daemon 220
Table 18: IPMB trace collection packet field 222
Table 19: IPMB message list basic information 223
Table 20: IPMI filter primitives 239
Table 21: Flash partitioning for ChMM-700s (64MB Flash) in UBIFS mode 248
Table 22: File names for firmware images stored as UBIFS files 249
Table 23: Flash partitioning for ChMM-700s (64MB Flash) in JFFS2 mode 251
Table 24: Reliable upgrade stages 254
Table 25: Protocols recognized in the URLs by the reliable upgrade utility 257
Table 26: Standard source file names for upgrade components 260

1 About this document

This document describes the Pigeon Point Chassis Manager. This PDF file requires Adobe Acrobat Reader 7.0 or later for full functionality. For a free reader, go to http://www.adobe.com.

1.1 Chassis Manager documentation

This document is one of two documents in the Chassis Manager documentation set. These documents are available in PDF file format. The complete set includes:

Table 1: Chassis Manager documentation

DOCUMENT	DESCRIPTION
Pigeon Point Chassis Manager User Guide	This document describes the overall configuration and use of the Pigeon Point Chassis Manager.
Pigeon Point Chassis Manager External Interface Reference	This document describes how to use the Chassis Manager command line interface, web interface, Simple Network Management Protocol (SNMP) interface and Remote Management Control Protocol (RMCP) interface.

1.1.1 Conventions used in this document

This table describes the textual conventions used in this document.

Table 2: Conventions used in this document

CONVENTION SAMPLE	DESCRIPTION
`setenv`	This 10 point bold Courier font is used for text entered at keyboard in example dialogues, which typically occur as one or more separate lines.
`U-Boot 1.0.2 (Apr 18 2006 - 14:58:54)`	This 10 point normal Courier font is used for ChMM output in example dialogues.
`addmisc`	This 12 point italic Courier New font is used for special text within normal paragraphs. The types of such special text include command names, file names, configuration parameters and command parameters, plus other text that could be entered by or displayed to a Chassis Manager user. This

	font is also used for command syntax definitions.
“Get Device ID”	IPMI commands defined by the IPMI specification or as VSO or PICMG extensions are shown in the normal font, surrounded by double quotes. This matches the corresponding convention used in VSO and PICMG specifications.

1.2 Additional resources

For more information about Pigeon Point products for VPX and OpenVPX, please visit the following website: http://www.pigeonpoint.com/products_vita.html.

2 Introduction

This section provides an overview of the Pigeon Point Chassis Manager and Chassis Management Mezzanine (or ChMM, currently the ChMM-700R) products. The Pigeon Point Chassis Manager is a chassis-level management solution for VITA 46.11 chassis.

The Pigeon Point ChMM, when coupled with a corresponding carrier board, provides the necessary hardware to run the Chassis Manager within a VITA 46.11 chassis. This document focuses on aspects of the Chassis Manager and ChMM that are common to any ChMM carrier used in a VITA 46.11 context. Carrier-specific and chassis-specific details are documented by chassis providers.

The ChMM-700R was designed to be RoHS-compliant from the start and there are no ChMM-700 products, though references to ChMM-700R may be simplified as ChMM-700.

2.1 In this section

This section contains the topics listed below. Just click on a topic to go to it.

- Intelligent Platform Management: a VITA 46.11 overview
- Pigeon Point Board Management Reference: hardware and firmware
- Pigeon Point Chassis Manager and ChMM

2.2 Intelligent Platform Management: a VITA 46.11 overview

The Pigeon Point products are the first Intelligent Platform Management building blocks designed from the ground up for modular platforms like VITA 46.11, with its strong leverage of the ATCA hardware platform management layer, adapted for the needs of the VPX platform.

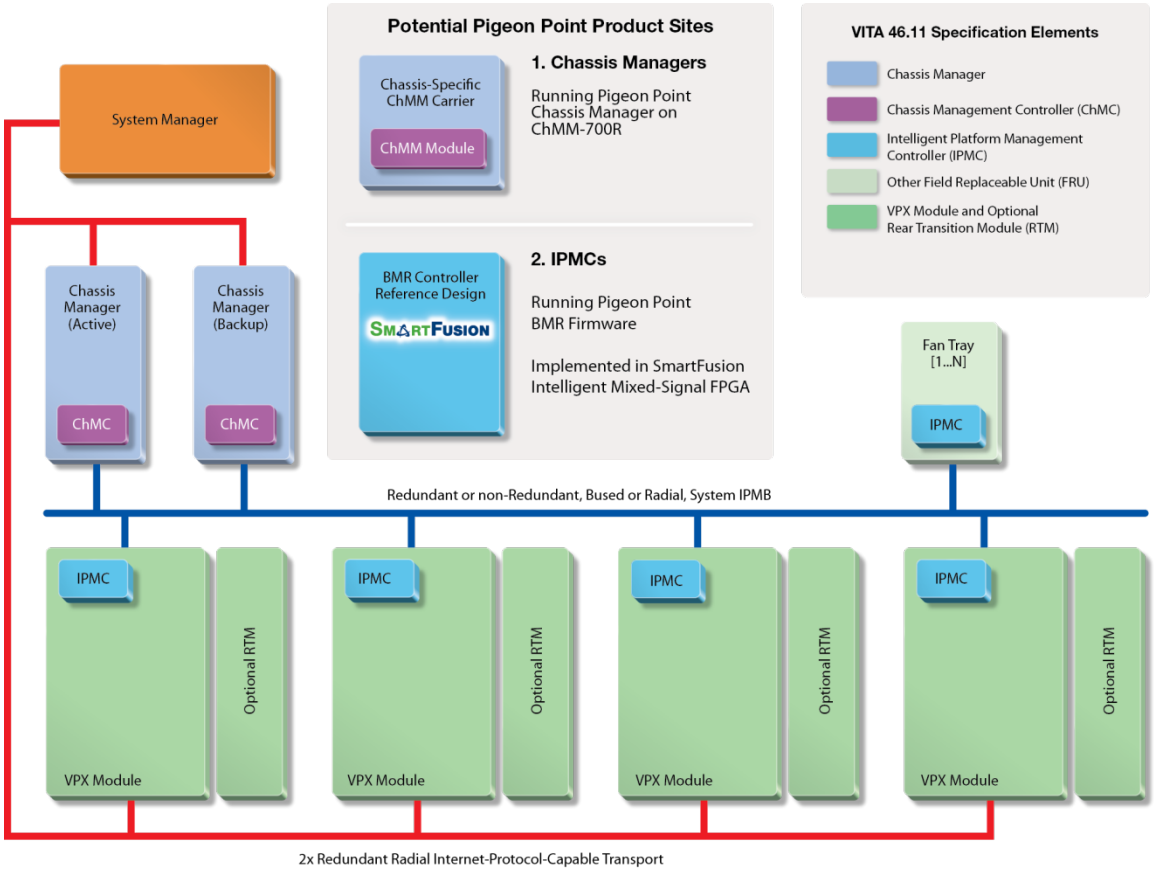
The Intelligent Platform Management Interface (IPMI) specification provides a solid foundation for the management of such platforms, but requires significant extension to support them well.

IPMI defines a management infrastructure that is widely used across the PC and server industry. The VITA 46.11 specification defines the necessary extensions to IPMI for VPX, most of them leveraged and adapted from the corresponding extensions done by PICMG for ATCA. Those PICMG and VSO specifications are based on revision 1.5 of IPMI. However, the Pigeon Point Chassis Manager conforms to revision 2.0 of the IPMI specification and implements numerous IPMI 2.0 features, including: RMCP+, Virtual LAN support, and firmware firewall commands.

VSO has significantly extended IPMI to cover the needs of open modular architectures. In fact, the VITA 46.11 specification includes the definition of 30 new commands, 6 new FRU Information data structures—several quite complex—and 7 new sensor types.

Figure 1 shows the logical elements of an example VITA 46.11 chassis, identified in terms of the VITA specification, and potential sites for incorporation of Pigeon Point products.

Figure 1: Management aspects and potential Pigeon Point product sites in an example VITA 46.11 chassis



A VITA 46.11 Chassis Manager communicates inside the chassis with Intelligent Platform Management Controllers (IPMCs), each of which is responsible for local management of one or more Field Replaceable Units (FRUs), such as boards, fan trays or power entry modules. Management communication within a chassis occurs primarily over the Intelligent Platform Management Bus (IPMB), which is implemented on an optionally dual redundant basis as the System IPMB in VITA 46.11.

An overall System Manager (typically external to the chassis) can coordinate the activities of multiple chassis. A System Manager typically communicates with each Chassis Manager over Ethernet. The *Pigeon Point Chassis Manager External Interface Reference* document details the interfaces and protocols that are available for such interactions.

The next two sections address the board and chassis levels of management, highlighting the following Pigeon Point products and their capabilities as well as the relevant VITA 46.11 functionality:

- Pigeon Point Board Management Reference (BMR) firmware and corresponding hardware reference design, which together implement IPMCs for various types of FRUs.

- Pigeon Point Chassis Manager software and ChMM mezzanine module, which, together with an appropriate ChMM carrier board, implement a VITA 46.11-compliant Chassis Manager and Chassis Management Controller (ChMC).

2.3 Pigeon Point Board Management Reference: hardware and firmware

This hardware and firmware level includes the local management of VPX modules as well as other auxiliary FRUs, such as fan trays or power entry modules. Based on the interfaces specified by IPMI and extended by VITA 46.11, any compliant Chassis Manager can work with any compliant IPMC and the FRUs that it represents.

This section focuses on controllers based on Pigeon Point technology as a concrete example.

The Pigeon Point BMR reference design can be implemented as part of any module or other Intelligent FRU, and executes the corresponding firmware, thereby realizing a compliant IPM Controller. The BMR firmware represents one or more FRUs via System IPMB to the Chassis Manager, including:

- Providing inventory information identifying each such FRU, including its manufacturer and other data.
- Describing and implementing a set of logical sensors (such as for temperature, state of System IPMB, and operational state for each FRU (activated, deactivated, etc.)).
- Generating events (typically directed to the Chassis Manager) for exceptional conditions detected by any sensor, based on its configured event generation settings.

The BMR offering for VITA 46.11 is based on the Microsemi SmartFusion customizable System-on-Chip (cSoC). Please see http://www.pigeonpoint.com/products_vita.html for more details, including product briefs, on this offering.

The Pigeon Point Board Management Starter Kit for VITA 46.11 IPMCs includes all the materials necessary (documentation, schematics, bill of materials, firmware source code and development tools, etc.) for Intelligent FRU developers to integrate a reference design directly into their boards and take immediate advantage of the fully validated BMR firmware.

2.4 Pigeon Point Chassis Manager and ChMM

The Pigeon Point Chassis Manager is the Pigeon Point solution for chassis level management in VITA 46.11. Since the VITA 46.11 specification leverages a lot of the infrastructure and design principles from the PICMG 3.0 AdvancedTCA specification, the Chassis Manager is built on the base of the Pigeon Point Shelf Manager, Pigeon Point's solution for shelf level management in AdvancedTCA shelves, and has a common design with it, and also substantial common code. Also, the hardware platform for the Chassis Manager (ChMM-700) is based on the same physical hardware mezzanine as the ShMM-700 mezzanine, which is one of the hardware platforms for the Shelf Manager. File names, configuration variables, commands and output for the Chassis Manager have much in common with the Shelf Manager, and sometimes use AdvancedTCA terminology (e.g.

"shelf" vs "chassis"). Over time, more of these uses of AdvancedTCA terminology will be removed from the Chassis Manager.

The Pigeon Point Chassis Manager (consistent with VITA 46.11 Chassis Manager requirements) has two main responsibilities:

- Manage/track the FRU population and common infrastructure of a chassis, especially the cooling resources and their usage. Within the chassis, this management/tracking primarily occurs through interactions between the Chassis Manager and the IPM Controllers over System IPMB.
- Enable the overall System Manager to join in that management/tracking through the System Manager Interface, which is typically implemented over Ethernet.

Much of the Pigeon Point Chassis Manager software is devoted to routine missions such as powering a chassis up or down and tracking the FRU population. In addition, the Chassis Manager can take direct action when exceptions are raised in the chassis. For instance, in response to temperature exceptions the Chassis Manager can raise the fan levels or, if that step is not sufficient, even start deactivating FRUs to reduce the heat load in the chassis.

2.4.1 Chassis Manager compatibility with the Pigeon Point Shelf Manager

Here are the main differences in the Pigeon Point Chassis Manager compared to the Pigeon Point Shelf Manager:

1. The Shelf Manager manages an ATCA shelf populated with ATCA-compliant FRUs; in contrast, the Chassis Manager manages a VITA 46.11-compliant VPX chassis populated with VITA 46.11 compliant intelligent FRUs.
2. Physical hot swap of FRUs is not supported in a VPX chassis; therefore the M-state diagram is different in this context.
3. To communicate with the IPMCs in the chassis, the Chassis Manager uses VSO Group Extension commands (instead of PICMG Group Extension commands) in addition to regular IPMI commands.
4. The CLI command `shelf` has been replaced with the command `chassis`. The supported subcommands are `address_table` or `at`, `cooling_state` or `cs`, `fan-state` or `fs.` Use the command `clia chassis at` to show the current Chassis Address Table.
5. A new CLI command `persistentcontrol` has been implemented; it provides access to persistent reset facilities of VPX modules.
6. The CLI, SNMP and WEB interfaces have been preserved in the Chassis Manager and remain largely unchanged.
7. For SNMP, a new MIB has been designed (and the MIB file PPS-CHMGR-MIB.txt replaces PPS-SENTRY-MIB.txt). The root OID for the Chassis Manager differs from the Shelf Manager root OID in the last component:
   `iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).pps(16394).product`

`s(2).chassis-management(1).ipm-sentry-chmm(2)`. Numeric codes of subsidiary SNMP variables remain unchanged.

8. The file *`/var/nvdata/chassis_fru_info`* is used as the Chassis FRU Information storage in *`-sf-`* mode (instead of *`/var/nvdata/shelf_fru_info`* for the Shelf Manager).
9. The Chassis Manager requires the mandated VITA 46.11–defined records to be present in the Chassis FRU Information to work properly.
10. For proper operation of the Chassis Manager, and in accordance with the VITA 46.11 specification, all hardware addresses that can host IPMCs in a VITA 46.11 chassis must be listed in the VITA 46.11 Address Table in the Chassis FRU Information. This is especially important for Tier-1 IPMCs because they are only detected through polling and only addresses listed in the Chassis Address Table are polled.
11. Many log messages have been updated to refer to the Chassis FRU Information (instead of Shelf FRU Information).

Note: in `-sf` *mode, the Chassis Manager takes Chassis FRU Information from an EEPROM. If a BTC-700R carrier is used, this EEPROM may be not initialized at all or may be initialized with ATCA Shelf FRU Information. In either of those cases, it is necessary to write the Chassis FRU Information to the EEPROM using the command* `eepromw –i 5 a0 /var/nvdata/chassis_fru_info`.

2.4.2 Pigeon Point Chassis Manager features

The Pigeon Point Chassis Manager features are listed below:

- Executes on the ChMM, a compact mezzanine module, installed on a suitable carrier board for the chassis.
- Conforms to the VITA 46.11 specification.
- Monitors activities within the chassis via the VITA-specified optionally dual redundant Intelligent Platform Management Bus (System IPMB).
- Accepts and logs events posted by any intelligent FRU in the chassis (reflecting exceptions in temperatures, voltages, etc.); posts alerts outside the chassis based on configurable IPMI Platform Event Filters.
- Supports redundant Chassis Manager instances for high availability.
- Integrates a watchdog timer, which resets the ChMM if not periodically strobed; such resets automatically trigger a switchover to the backup ChMM, if configured.
- Includes battery-backed real-time clock for time-stamping events.
- Implements rich set of chassis-external interfaces accessible over Ethernet, including Remote Management Control Protocol (RMCP, required by VITA 46.11), command line, web browser, Simple Network Management Protocol (SNMP).

- Includes a Chassis Adaptation Layer based on the Pigeon Point Hardware Platform Description Language (HPDL) that makes it straightforward to adapt the Chassis Manager for operation across a wide range of VITA chassis architectures.

2.4.3 Support for dual redundant operation

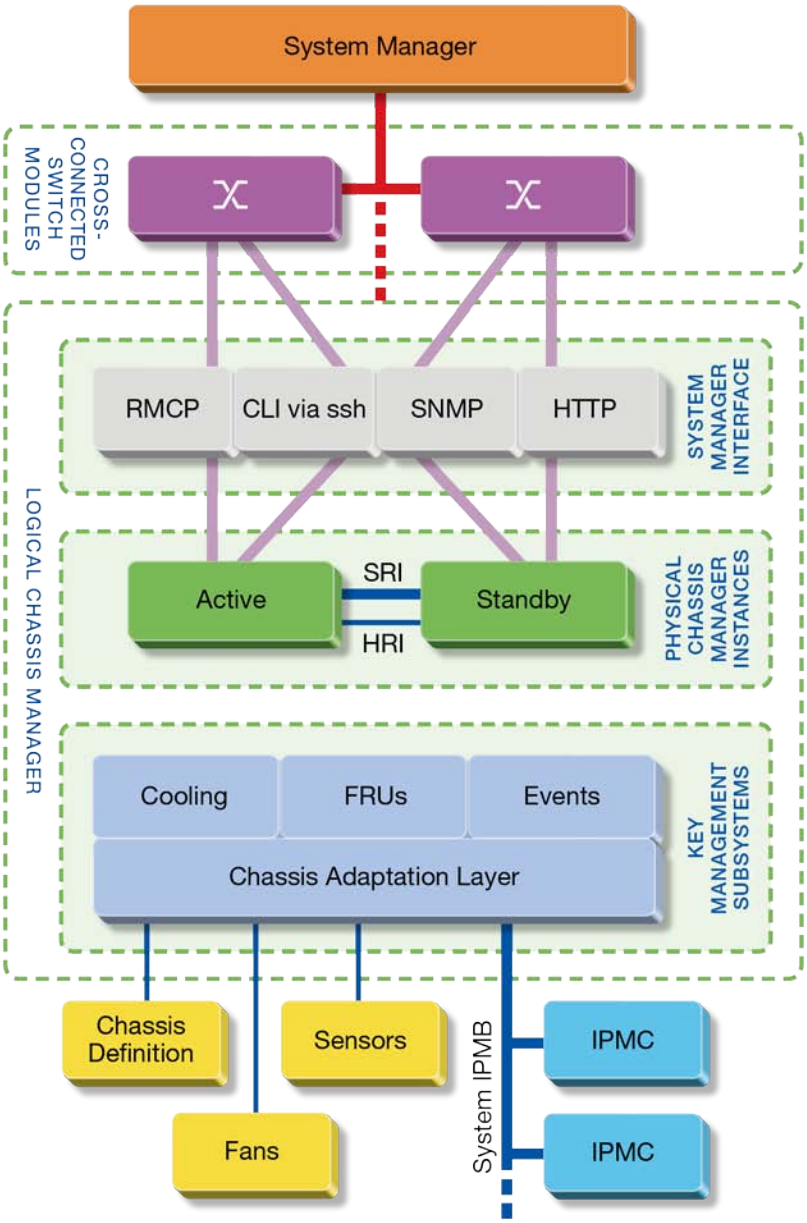
The Pigeon Point Chassis Manager can be configured with active/backup instances to maximize availability.

Figure 2 shows how both Chassis Manager instances are accessible to the System Manager, with primarily the active instance interacting at any given time. Only the active instance communicates over System IPMB with the IPM Controller population in the chassis.

The two Chassis Manager instances communicate over TCP/IP, with the active instance posting incremental software level state updates to the backup instance via a Software Redundancy Interface (SRI) implemented between them. As a result, the backup can quickly step into the active role if necessary. Due to historical reasons, a non-Ethernet SRI is often referenced elsewhere in this document as an Alternate Software Redundancy Interface. The SRI is implemented via a pair of cross-connected USB links on the ChMM-700R.

Hardware level status updates are communicated between the Chassis Manager instances via the Hardware Redundancy Interface (HRI).

Figure 2: Pigeon Point Chassis Manager architecture and context



As shown in Figure 2, ChMC cross connects allow both ChMMs to be connected with both switch modules. This improves system availability because either switch modules or ChMMs can switchover independently, if necessary. ChMC cross connect support requires the use of both ChMM Ethernet links.

The HRI ensures that three key items of coordination status are available to each Chassis Manager instance:

- Presence: each Chassis Manager instance knows whether the other instance is present in the chassis.
- Health: each instance knows whether the other instance considers itself “healthy.”
- Switchover Request: the backup instance can force a switchover if necessary.

Figure 2 shows other aspects of the architecture and context of the Pigeon Point Chassis Manager, as well. These include the Chassis Adaptation Layer (and the chassis description that drives it) and the main Chassis Manager subsystems that implement the features summarized in Section 2.4.1.

2.4.4 System Manager Interface

Another major subsystem of the Pigeon Point Chassis Manager implements the System Manager Interface. “System Manager” is a logical concept that may include software as well as human operators in the “swivel chairs” of an operations center. The Pigeon Point Chassis Manager provides a rich set of System Manager Interface options, which provide different mechanisms of access to similar kinds of information and control regarding a chassis. Figure 2 shows the System Manager Interface options supported by the Pigeon Point Chassis Manager.

One such mechanism is the IPMI LAN Interface. To maximize interoperability among independently implemented chassis products, this interface is required by the VITA 46.11 specification and supports IPMI messaging with the Chassis Manager via the IPMI Remote Management Control Protocol (RMCP).

A System Manager that uses RMCP to communicate with chassis should be able to interact with any VITA 46.11-compliant Chassis Manager. This relatively low level interface provides essentially complete access to the IPMI aspects of a chassis, including the ability for the System Manager to issue IPMI commands to IPMCs in the chassis, using the Chassis Manager as a proxy.

The Pigeon Point Chassis Manager also supports Simple Network Management Protocol (SNMP) access to the chassis. This popular management protocol is supported with a custom Management Information Base (MIB) providing Get and Set access to a wide range of information and controls regarding the chassis.

In addition, the Pigeon Point Chassis Manager provides two interfaces oriented towards human users rather than programmatic ones:

- Command Line Interface (CLI): This interface provides a comprehensive set of textual commands that can be issued to the Chassis Manager via either a physical serial connection or a telnet connection.
- Web-based Interface: This interface enables essentially the same functionality as the CLI, with access to the Chassis Manager via a web browser.

Using these mechanisms, the System Manager can access information about the current state of the chassis, including current FRU population, sensor values, threshold settings, recent events and overall chassis health.

These aspects of VITA 46.11 System Manager Interface are considered to be part of the chassis-external interfaces. They are documented separately in the *Chassis Manager External Interface Reference*.

3 Configuration

The Chassis Manager application runs on a specialized distribution of Linux - Pigeon Point Linux. The lowest layer is the firmware monitor, which is called U-Boot.

3.1 In this section

This section contains the topics listed below. Just click on a topic to go to it.

- Setting up U-Boot
- Setting up Chassis Manager configuration file
- Setting up Ethernet
- Configuring the FRU Information
- Configuring carrier and chassis attributes using HPDL
- Configuring the cooling management strategy
- Configuring local sensors
- Setting the Auxiliary Firmware Revision
- Setting up the clock
- Setting up and using ChMM Power On Self Tests
- Configuring external event handling
- Configuring the Platform Event Trap format
- Configuring system services
- Configuring the speed of I^2C buses

3.2 Setting up U-Boot

On a power-up/reset of the ChMM, the hardware starts executing the U-Boot firmware in Flash. The firmware performs basic initialization of the ChMM, and unless the user explicitly disables the Autoboot feature (thus forcing the firmware to switch to the maintenance user command interface), commences booting the Linux kernel. Linux is booted from the kernel and root file system images residing in Flash. U-Boot relocates the kernel image to RAM, sets up kernel parameters, and passes control to the kernel entry point.

```
PowerPrep start initialize power...
Battery Voltage = 0.80V
No battery or bad battery detected!!!.Disabling battery voltage measurements.
Feb 24 201207:45:29
FRAC 0x92926152
memory type is DDR2
Wait for ddr ready 1
```

```
power 0x00820616
Frac 0x92926152
start change cpu freq
hbus 0x00000003
cpu 0x00010001
start test memory access
ddr2 0x40000000
finish simple test
U-Boot 2009.08 (Oct 24 2017 - 11:44:23)

Freescale i.MX28 family
CPU:   297 MHz
BUS:   99 MHz
EMI:   130 MHz
GPMI:  24 MHz
I2C:   ready
DRAM:  128 MB
SFGEN: S25FL512S detected, total size 64 MB
A2F:   SPICOMM protocol v1.7, M3 firmware v1.8, FPGA design v1.1.0.0
A2F:   A2F firmware, version 1.8
A2F:   Device type: A2F060M3E-FG256
A2F:   MSS clock frequency: 40 MHz
A2F:   Fabric clock frequency: 20 MHz
A2F:   Fast delay calibration: 1330 cycles per 100uS
A2F:   eNVM: 128 KB (00000000 - 00020000)
A2F:   eSRAM: 16 KB (20000000 - 20004000)
A2F:   Extram start: 20002750
A2F:   Last reset cause: HARD
A2F:   VMON flags: 00000000
RUPG:  booting from image 1 (confirmed)
In:    serial
Out:   serial
Err:   serial
Net:   FEC0: 00:18:49:02:b2:06, FEC1: 00:18:49:02:b2:07
FEC0, FEC1
Hit any key to stop autoboot:  0
ChMM700
```

Note: "ChMM700" is the U-Boot prompt for user commands. For a complete set of supported commands, type `help`.

3.2.1 U-Boot environment variables

U-Boot includes a set of environment variables that should be configured prior to U-Boot use. The following table describes the default set of variables available. Some of these variables are set automatically by U-Boot and should not be modified by a user.

Table 3: U-Boot environment variables

ENVIRONMENT VARIABLE	DESCRIPTION
a2f_upg_file	This variable contains the name of the SmartFusion cSoC image file that U-Boot looks for on the boot volume. This variable is set automatically by U-Boot during bootstrap and must not be changed by a user; its value is used by U-Boot to pass information about the SmartFusion (A2F) cSoC upgrade image file to the Linux kernel. Only three values can be used: *a2f-upgrade.dat* (in the case of a normal A2F upgrade), *a2f-auto-rollback.dat* (in the case of an A2F upgrade rollback), or empty (if no A2F upgrade is in progress). *Note: U-Boot expects the* `a2f-auto-rollback.dat` *file to be stored on the boot volume and to contain an image of the currently running firmware. The image is used to rollback an A2F upgrade if new candidate firmware is not programmed successfully or fails any of the post-upgrade tests.*
alt_image	This variable contains the rollback image set index. This variable is used by *load_ubifs* command to determine which of the Linux kernel and RFS images to boot if the current image set is broken and there is no reliable upgrade running. The variable is set automatically by U-Boot during bootstrap and should not be changed by the user; its value is always opposite to *image_sel*.
baudrate	Serial port baud rate, default is *115200*.

Environment variable	Description
bootargs	Command line to be passed to the Linux kernel. May contain references to other U-Boot environment variables, which is resolved at run-time. The default value is: *console=ttySP0,115200n8 mtdparts=m25p80-1:512K(uboot0),512K(uboot1),256K(envvars0),256K(envvars1),256K(ubifs_cache),30976K(boot),32M(user) quiet fec_mac=${ethaddr} root=/dev/ram*
bootargs_common	U-Boot command that sets up common kernel bootargs parameters. It's called by *bootargs_initrd* and *bootargs_nfs* commands to set up *console*, *mtdparts*, *quiet*, *fec_mac* kernel arguments.
bootargs_initrd	U-Boot command that sets up kernel bootargs string so that the kernel shall be booted with RFS image stored in RAM. It executes *bootargs_common* command and then extends bootargs string with *root=/dev/ram* kernel parameter.
bootargs_nfs	U-Boot command that sets up kernel bootargs string to prepare the system to be booted with NFS volume as RFS. It executes *bootargs_common* command and then extends bootargs string with the following NFS parameters: *root=/dev/nfs ip=${ipaddr}:::::eth0:off nfsroot=${serverip}:${rootpath},v3,tcp*. So, the kernel shall use *ipaddr* to configure *eth0* interface and access *serverip* host to use its exported *rootpath* share as root.
bootcmd	U-Boot command executed to accomplish auto-booting. Normally, this is something similar to: *run mount_ubifs; a2f upgrade; run ubifs*
bootdelay	Autoboot delay value, in seconds. Default setting is *3*.
bootfile	Parameter that specifies what kernel image should be used by the *net* and *nfs* boot options.
console	Setting for the kernel and init script console port and baud rate. The default is *ttySP0,115200n8*

Environment variable	Description
ethact	Parameter that specifies which Ethernet controller should be used by U-Boot when handling *bootp*, *dhcp*, *nfs*, *ping*, *rarpboot*, *tftpboot* commands. Supported values are: *FEC0*, *FEC1*. *FEC0* selects *eth0* interface, *FEC1* selects *eth1* interface. Default value is *FEC0*.
ethaddr	MAC address of the primary on-chip Ethernet controller. The value of this variable is set automatically by U-Boot. This address is passed to the kernel Ethernet driver.
eth1addr	MAC address of the secondary Ethernet controller. The value of this variable is set automatically by U-Boot. This address is not passed to the kernel Ethernet driver directly, but both the U-Boot and the Linux kernel FEC driver set up the MAC address for the second Ethernet controller as *ethaddr* incremented by 1, so both U-Boot and Linux use the same MAC for the second Ethernet controller.
eth0_switch_bus	This variable defines the logical I2C bus that is connected to the GbE Switch for the ChMM-700R Ethernet 0 PHY interface. The default value is *0*, which corresponds to physical I2C interface I2C4. *Note: this feature is supported starting in release 1.3.0.*
eth0_switch_address	This variable defines the 7-bit I2C address of the GbE Switch for the ChMM-700R Ethernet 0 PHY interface. The default value is *0x5F*. *Note: this feature is supported starting in release 1.3.0.*
eth1_switch_bus	This variable defines the logical I2C bus that is connected to the GbE Switch for the ChMM-700R Ethernet 1 PHY interface. The default value is *5*, which corresponds to physical I2C interface I2C3. *Note: this feature is supported starting in release 1.3.0.*
eth1_switch_address	This variable defines the 7-bit I2C address of the GbE Switch for the ChMM-700R Ethernet 1 PHY interface. The default value is *0x5F*. *Note: this feature is supported starting in release 1.3.0.*

Environment Variable	Description
flash_reset	Instructs Linux to erase the Flash filesystems (*/etc* and */var*), restoring them from the contents of the current RFS image (*y*/*n*). The system startup script sets this variable back to *n* after the Flash erase. Default is *n*. This variable can also be set to *user* (*y*/*n*/*user*) to cause the system startup script to erase the entire 'user' partition (where */etc* and */var* are located). This mode can be used in case of unrecoverable 'user' partition corruption or together with the *user_partition* variable while converting the 'user' partition to the JFFS2 or UBIFS format. *Note: this mode is supported starting in release 1.1.0.*
gatewayip	Default gateway IP address. This variable can be passed as a part of the kernel command line to automatically configure routing for the network interfaces. Default setting: *192.168.0.1*.
hostname	Network host name; default is *chmm700*.
image_sel	This variable contains the current image set index. The variable is used in the *load_ubifs* command to determine which of the Linux kernel and RFS images to boot during bootup. The variable is set automatically by U-Boot during bootstrap.
ipaddr	IP address used by the primary on-chip Ethernet interface. This variable configures the network interface specified by *ipdevice* automatically if the *rc_ifconfig* variable is set to *y*. Note that the system startup script sets the least significant bit of this variable to the least significant bit of the Hardware Address for the ChMM carrier; that is, if the Hardware Address is an even value, the last bit in the IP address is set to 0, otherwise it is set to 1. This is done in the startup script */etc/netconfig* to support coordinated IP address configurations on redundant ChMMs. To disable this functionality, simply remove the */etc/readhwaddr* file.
ip1addr	IP address used by the secondary Ethernet interface. This variable can be passed as a part of the kernel command line to automatically configure the corresponding kernel network interface.

Environment Variable	Description
ip2addr	IP address used by the third network interface, if present. This variable can be passed as a part of the kernel command line to automatically configure the corresponding kernel network interface. This variable is not assigned by default.
ipdevice	Device corresponding to *$ipaddr*. Default value is *eth0*.
ip1device	Device corresponding to *$ip1addr*. The default is *usb0*.
ip2device	Device corresponding to *$ip2addr*. Not assigned by default, but can be assigned if more than two network interfaces are used on the ChMM (e.g. a USB- or serial-based network interface can be designated this way).
kernel_start	The absolute starting address of the kernel image in RAM. This variable is used in *ubifs*, *nfs*, *net*, *load_ubifs*, *load_ubifs_alt* to specify the address in physical memory where the kernel image should be loaded and where it is expected to be during system boot. The default value is *0x42000000*.
load_ubifs	U-Boot command that loads current Linux kernel and RFS images from the Flash drive into RAM. Images are selected on the basis of the *image_sel* variable value. The kernel image is loaded at the *loadaddr* address, and the RFS image is loaded at the *rfsaddr* address. This command is invoked by *ubifs*, explicitly.
load_ubifs_alt	U-Boot command that loads alternative Linux kernel and RFS images from the Flash drive into RAM. Images are selected on the basis of the *alt_image* variable value. The kernel image is loaded at the *loadaddr* address and the RFS image is loaded at the *rfsaddr* address. This command is invoked by *ubifs* command if one of the images from the current image set (selected by the *image_sel* value) is broken and there is no reliable upgrade running.
log_max	Specifies the size limit for the syslog file in bytes. Default is *250000*. When the log file size reaches the limit, the file is renamed as a backup file, and a new syslog file is started (replacing any existing backup), so that the maximum space used by the overall syslog is twice the *log_max* value.

Environment Variable	Description
log_remote	Specifies the IP address for the remote syslog facility. By default, this option is not set. The syslog daemon on Linux systems can be configured to receive messages from remote hosts, normally using the *-r* option, so using the *log_remote* setting it is possible to send the ChMM system log to a remote system.
log_remote2	Specifies the IP address for the secondary remote syslog facility. By default, this option is not set. The syslog daemon on Linux systems can be configured to receive messages from remote hosts, normally using the *-r* option. Note this option is only used when the *log_remote* option is set.
logging	Specifies if messages log file should be maintained in ram or Flash. Default is *ram*, which is the recommended option.
mac_override	This variable allows overriding MAC address configuration for both Ethernet controllers for the Linux kernel. This variable is used in *bootargs* generation and by default configures *ethaddr* to be passed as a configuration parameter. Default value is *set bootargs ${bootargs} fec_mac=${ethaddr}*. If a different MAC address is specified and it is a valid MAC address, the FEC0 controller is configured with it, while FEC1 controller is configured with an incremented value. If no value is specified (i.e. *mac_override* is empty or doesn't set *fec_mac* value), the Linux kernel driver selects MAC addresses by its built-in algorithm. The MAC addresses generated by the kernel in this latter case may be the same as what the U-Boot reports via *ethaddr* and *eth1addr* variables, but this is not guaranteed.
mount_ubifs	U-Boot command that invokes MTD configuration U-Boot scripts. This command is executed by U-Boot during bootstrap. Default value is *run set_mtdparts*.
mtdids	This variable contains SPI interfaces configuration and is used privately by the U-Boot to configure the SPI subsystem to access Flash drives. Default value is *spi0=SPI-0*.

Environment Variable	Description
mtdpart1	This variable contains the Flash drive partition map. The variable is used by the U-Boot and its value is also passed to the Linux kernel as part of *bootargs* string. The variable specifies how the Flash drive is split into partitions and consists of pairs of partition sizes and their names. Default value is *512K(uboot0),512K(uboot1),256K(envvars0),256K(envvars1),256K(ubifs_cache),30976K(boot),32M(user)*. See the ChMM-700R Hardware Architecture Specification for more information.
net	This variable can be used as a replacement for *bootcmd* as a means of booting a kernel and RFS image from TFTP. Use *run net*.
netmask	Network netmask, default value is *255.255.255.0*
nfs	This variable can be used as a replacement for *bootcmd* as a means of booting and running with an NFS mounted root filesystem. *Note:* the kernel image is fetched from a TFTP server by default.
password_reset	Instructs Linux to restore the password from the current RFS image. The default password for user *root* in the standard provided RFS images is the empty password “”. Default value for *password_reset* is *n*.
post_normal	Determines the list of POST tests that are executed on each boot-up. If not set, compile-time default settings are used. The test names listed in a value of this variable are separated by space characters.
post_poweron	Determines the list of POST tests that are executed after power-on reset only (vs. on each boot-up). If not set, compile-time default settings are used. The test names listed in a value of this variable are separated by space characters.
serial#	This variable contains the ChMM serial number assigned during manufacturing. The variable is set automatically by U-Boot during bootstrap.
rootpath	This variable specifies the NFS root to be used by the Linux kernel when booting the ChMM via the *bootcmd_nfs* command. Default value is */rootfs*.
quiet	Instructs the kernel upon bootup not to print progress messages to the serial console. Default is *quiet*.

Environment Variable	Description
ramdisk	Specifies what *.rfs* image should be used by the *net* boot option.
rc_ifconfig	Allows the */etc/rc* script to set up the IP address instead of *shelfman*. Default is *n* (allow *shelfman* to set up IP addresses).
rc2	Specifies secondary RC script that is to be invoked. Default is */etc/rc.chmm700-hpdl* or other appropriate script for the relevant target platform. This variable must be set to a carrier-specific value matching the carrier on which the ChMM is installed.
rfs_start	The absolute starting address of the RFS image in RAM. This variable is used in the *ubifs*, *net*, *load_ubifs*, and *load_ubifs_alt* commands to specify the address in physical memory where the RFS image should be loaded and where it is expected to be during ChMM boot. The default value is *0x46000000*.
rmcpaddr	Default IP address for the RMCP service.
serverip	IP address of the TFTP server
set_mtdparts	U-Boot command that is invoked to set up the MTD configuration in U-Boot to perform boot volume access. The command is invoked by *mount_ubifs* during bootstrap.
start_rc2_daemons	Instructs the secondary startup script to start or not start the *snmpd*, *httpd* and *shelfman* daemons after bootup. Default is *y*.
time_proto	Protocol used to retrieve time from a network time server; possible values are *ntp* and *rdate*.
time_server	Time server for synchronization at runtime. If this variable is not specified, time is extracted from the hardware clock at system startup.
timezone	Local time zone in CCCn format where n is the offset from GMT and optionally negative, while CCC identifies the time zone. The default is *UTC*.

ENVIRONMENT VARIABLE	DESCRIPTION
ubifs	U-Boot command executed to perform system boot from the Flash drive. The command performs setting *bootargs* variable, loading both kernel and RFS images into RAM and booting the system. If at least one of the images is broken and there is no reliable upgrade in progress, the command also outputs an error message and performs a single attempt to boot the ChMM using the alternative (rollback) kernel and RFS images.
user_partition	Specifies the format of the 'user' partition, which can be either *jffs2* or *ubifs*, where *ubifs* is the default value. This variable value must only be changed along with the *flash_reset=user* setting to change the 'user' partition format. Note: such partition format changes are supported starting with release 1.1.0.
ver	The U-Boot version string. The value is taken from the U-Boot image. The version string contains the base U-Boot version and the build timestamp, as well. For example: *U-Boot 2009.08 (Feb 24 2012 - 07:44:12)*

3.2.2 Assigning values to environment variables

To assign a value to an environment variable, use the format:

ChMM700 **setenv <variable_name> <new_value>**

For example:

ChMM700 **setenv bootdelay 1**

Once all of the environment variables have been properly set, you need to save them back out to the Flash so that they remain after the ChMM is powered down. The *saveenv* command is used for this purpose.

ChMM700 **saveenv**

The *setenv* functionality is also available as a Linux utility with the same usage. To display U-Boot variables at the shell prompt, use the additional *getenv* utility or issue the *setenv* command without parameters.

3.2.3 Configuring U-Boot environment variables for the Chassis Manager

When U-Boot is started for the first time, the following default environment variables are defined.

```
bootargs=console=${console}
bootcmd=run mount_ubifs; a2f upgrade; run ubifs
bootdelay=3
baudrate=115200
```

```
netmask=255.255.255.0
bootfile="sentry.chmm700.kernel"
loadaddr=0x42000000
console=ttySP0,115200n8
rfsaddr=0x46000000
mtdids=spi0=SPI-0
mtdparts1=512K(uboot0),512K(uboot1),256K(envvars0),256K(envvars1),256K(ubifs_cac
he),30976K(boot),32M(user)
bootfile=sentry.chmm700.kernel
ramdisk=sentry.chmm700.rfs
quiet=quiet
mac_override=set bootargs ${bootargs} fec_mac=${ethaddr}
rootpath=/rootfs
ipdevice=eth0
ip1device=usb0
ip2device=eth1
ipaddr=192.168.0.22
ip1addr=192.168.1.2
gatewayip=192.168.0.1
rmcpaddr=192.168.0.2
netmask=255.255.255.0
hostname=chmm700
flash_reset=n
password_reset=n
logging=ram
timezone=UTC
rc_ifconfig=n
start_rc2_daemons=y
rc2=/etc/rc.chmm700-hpdl
set_mtdparts=set mtdparts mtdparts=SPI-0:${mtdparts1}
bootargs_common=set bootargs console=${console} mtdparts=m25p80-1:${mtdparts1}
${quiet}; run ${mac_override}
bootargs_initrd=run bootargs_common; set bootargs ${bootargs} root=/dev/ram
bootargs_nfs=run bootargs_common; set bootargs ${bootargs} root=/dev/nfs
ip=${ipaddr}:::::eth0:off nfsroot=${serverip}:${rootpath},v3,tcp
mount_ubifs=run set_mtdparts
load_ubifs=ubifsload ${loadaddr} uImage.${image_sel}; ubifsload ${rfsaddr}
rfs.${image_sel}
load_ubifs_alt=ubifsload ${loadaddr} uImage.${alt_image}; ubifsload ${rfsaddr}
rfs.${alt_image}
```

```
ubifs=run bootargs_initrd; run load_ubifs; bootm ${loadaddr} ${rfsaddr}; echo
ERROR: booting active image failed, trying alternate image...; run
load_ubifs_alt; bootm ${loadaddr} ${rfsaddr}; echo ERROR:bad image(s), aborting
nfs=tftp ${loadaddr} ${bootfile}; run bootargs_nfs; bootm ${loadaddr}
net=run bootargs_initrd; tftp ${loadaddr} ${bootfile} ; tftp ${rfsaddr}
${ramdisk} ; bootm ${loadaddr} ${rfsaddr}
```

Several of these environment variables need to be reconfigured with values that are appropriate to the network context in which the ChMM is used.

3.2.4 Establishing the secondary RC script

The secondary RC script gets invoked when the system configuration is established during the boot process. It is called from the primary RC script *`/etc/rc`*. The secondary script is by default *`/etc/rc.chmm700-hpdl`* or some other script that is appropriate for that platform. A typical name for this script is *`/etc/rc.<target_platform>`*.

The name of this carrier-specific startup script is defined by the U-Boot environment variable *`rc2`*. The variable *`rc2`* is the one environment variable that must definitely be changed for a ChMM and its carrier to work properly in a chassis.

The RC2 script sets up environment variables *`CARRIER`* and *`CARRIER_OPTIONS`*. These variables inform the Chassis Manager about the carrier on which it is installed and define carrier-specific options as necessary for each supported carrier. By default, the values of these environment variables are propagated to the corresponding configuration variables (see Table 5: Chassis Manager configuration parameters) *`CARRIER`* and *`CARRIER_OPTIONS`*. The configuration variables, in their turn, are retrieved and used by the Chassis Manager.

The U-Boot variable *`start_rc2_daemons`* instructs the secondary startup script to start or not start the daemons *`snmpd`* (SNMP server), *`httpd`* (HTTP server) and *`shelfman`* (the Chassis Manager) after Linux boots. If the U-Boot variable *`start_rc2_daemons`* is set to *`y`*, the secondary RC script should also define command-line options for automatic invocation of the *`shelfman`* daemon. It may also provide other configuration services.

3.3 Setting up Chassis Manager configuration file

The Chassis Manager configuration file (*`shelfman.conf`*) is located in the *`/etc`* directory. Each line in the file is either a comment line (starting with #) or a *`<name> = <value>`* pair, representing the assignment for the configuration parameter. The name and the value are separated with the equal sign =.

The configuration parameter name is case-insensitive. Each configuration parameter is one of the following types: Boolean, number, string, or IP-address. The values of string type of configuration parameters are case-sensitive. The format of the value conforms to the type of the configuration parameter as shown in the following table.

Table 4: Configuration parameter types and descriptions

CONFIGURATION PARAMETER TYPE	DESCRIPTION
Boolean	A Boolean can be represented by either the strings *FALSE* (*false*) or *TRUE* (*true*), or by their numerical representations of *0* and *1*, respectively.
Number	A whole (possibly signed) numeric value; hexadecimal notation "0x…" is also supported.
String	A string, quoted (always with double quotes "") or unquoted. Quoted strings may contain blanks; unquoted strings are terminated by the first blank. The maximum string size is specified separately for each string-oriented configuration parameter. These values are case-sensitive.
IP-address	An Internet Protocol address in decimal-dot ("xxx.xxx.xxx.xxx") notation.

It is possible to specify a value of an environment variable as a configuration parameter value, using the notation *$<envvar>*; in that case, the value of the variable *<envvar>* is substituted when the configuration file is read. Here is an example:

```
DEFAULT_RMCP_IP_ADDRESS = $IPADDR
```

After the Chassis Manager has been brought up for the first time, the IP addresses are stored with the IPMI LAN configuration parameters. The LAN configuration parameters can be accessed or modified via any of the RMCP, CLI, web, or SNMP external interfaces and any such modifications take precedence over the *shelfman* configuration file when the Chassis Manager is restarted. This is to ensure the persistency of any modifications that are made to the LAN IP Addresses and gateway via those interfaces.

If the Chassis Manager IP Connection record in the Chassis FRU Information contains an IP address, it takes precedence over all other settings of the external or RMCP IP address. Preferably, the Chassis FRU Information should either not specify this address at all or set it to *0.0.0.0* to ensure that addresses can be controlled through the Chassis Manager configuration file and the IPMI LAN configuration parameters. The value of *0.0.0.0* for an IP-address type of configuration parameter is interpreted as 'undefined'.

The Chassis FRU Information should specify the RMCP IP address; the Chassis Manager uses it and propagates it to the LAN configuration parameters. RMCP is available in the absence of the Chassis FRU Information only if the configuration parameter *RMCP_WITHOUT_SHELF_FRU* is defined and set to *TRUE*. In this case, the Chassis Manager uses the IP address stored in the channel parameters. If there are no stored channel parameters, it uses the IP address specified in the boot parameters of the ChMM.

Some Chassis Manager configuration variables can get their values from the Chassis FRU Information. This may be useful to users who need to associate certain configuration parameters with a specific instance of a chassis. In that case, even if ChMMs or ChMM carriers are moved between chassis, the configuration parameters specified in the Chassis FRU Information stay with the chassis and override the parameter values specified in configuration files on the ChMM.

If the same variable is specified in a configuration file on the ChMM and in the Chassis FRU Information, the value from the Chassis FRU Information overrides the value from configuration files. Note that configuration files are parsed early during the Chassis Manager initialization, while the Chassis FRU Information can be found substantially later.

Not all variables can be specified in the Chassis FRU Information; the main reason is that many variables control the Chassis Manager initialization behavior, which happens before the Chassis FRU Information is found. For such variables, overriding the value when the Chassis FRU Information is found, could not have any effect, since the choice they control has typically already been made. If such variables are specified in the Chassis FRU Information, they are parsed successfully but value assignments are silently ignored.

The configuration variable *SHELF_MANAGER_CONFIGURATION_IN_SHELF_FRU_INFO* controls whether this functionality of the Chassis Manager is enabled. By default, the value of this variable is *FALSE* and the functionality is disabled.

The following table lists the configuration parameters that are currently supported.

Table 5: Chassis Manager configuration parameters

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
ACTIVATE_LOCAL_WITHOUT_SHELF_FRU	Boolean	FALSE	If set to *TRUE*, both IPM Controllers exposed by the active Chassis Manager (representing the physical and the logical Chassis Managers) are activated even if the Chassis FRU Information cannot be found. This option should be used with caution, because the power consumption of the payload of the physical Chassis Manager IPM Controllers may potentially exceed the power capability of the corresponding slot in the chassis.	No
ALARM_CUTOFF_TIMEOUT	Number	600	The alarm cutoff timeout (time after which the alarm cutoff is deactivated), in seconds.	Yes
ALLOW_ALL_COMMANDS_FROM_IPMB	Boolean	TRUE	If set to *TRUE*, most of the commands allowed from the RMCP interface are allowed from System IPMB as well (except for session-related commands). For example, “Cold Reset” and user management commands are accepted from System IPMB in this case. In this case, a malicious IPM Controller can seriously jeopardize the functionality of the chassis.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
ALLOW_CHANGE_EVENT_RECEIVER	Boolean	TRUE	If set to *TRUE*, the Event receiver address for the Chassis Manager can be set to an address other than 20h, LUN 0. If set to *FALSE*, any attempt to change event receiver address for the Chassis Manager is rejected.	Yes
ALLOW_RESET_STANDALONE	Boolean	FALSE	If set to *TRUE*, the command "Cold Reset" is accepted even if the Chassis Manager does not have an available backup, and reboots the Chassis Manager. By default, the command "Cold Reset" is accepted only in a dual redundant configuration and causes a switchover.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
ALLOWED_CIPHER_SUITES	Number	0xFFFFFFFF (decimal -1)	The bit mask of the IPMI 2.0 cipher suite IDs that a LAN client is allowed to use when establishing a session to the Chassis Manager. By default, all supported cipher suites can be used. If security considerations require, the set of allowed cipher suites can be restricted using this parameter. A bit value of 1b in the bit position corresponding to a cipher suite ID (0 to 14) indicates that the corresponding cipher suite can be used. There must be at least one cipher suite enabled, so a value 0 of this parameter is treated as if all cipher suites are allowed (just as with the default value).	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
ALTERNATE_CONTROLLER	Boolean	TRUE	Use alternate controller on the Chassis Manager with the address that is equal to the ChMM hardware address. If this variable is set to *TRUE*, the active Chassis Manager exposes two IPMB addresses: 20h and a second address based on its hardware address; the backup Chassis Manager exposes only the IPMB address based on its hardware address. After a switchover, the address 20h is exposed by the former backup Chassis Manager, which now exposes two IPMB addresses. If this variable is set to *FALSE*, the Chassis Manager exposes only the logical address 20h, this is allowed only for a non-redundant Chassis Manager. In the redundant configuration, this variable must be set to *TRUE*.	No
ATCA_TESTER_COMPATIBILITY	Boolean	FALSE	This variable, if set, turns off event handling optimizations in the Chassis Manager, so that the Chassis Manager behavior is compatible with the Polaris ATCA Tester; in particular, duplicate operational state events are handled as separate events if they have different IPMI sequence numbers.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
AUTO_SEND_MESSAGE	Boolean	TRUE	Automatically convert an RMCP request sent to a non-Chassis Manager IPMB address into a "Send Message" request directed to that address.	No
BOARD_LAN_PARAMETERS_CHANNEL_LIST	String(64)	""	The list of IPMI channel numbers on boards and modules that are available for assignment of LAN configuration parameters by the Chassis Manager. Channel numbers, each in the range 1 to 7, can be separated by commas, spaces or any other separators, for example: *3, 4, 5*. LAN parameters are assigned only to channels specified in this list and in the order in which they appear in this list. *Note: The facility that uses this configuration parameter has been superceded for the ATCA context by the PICMG HPM.3 specification. It will be removed from the Chassis Manager at some point, when an HPM.3 mapping for VPX systems becomes available.*	Yes
CARRIER	String(16)	"PPS"	The name of the specific carrier board on which the ChMM is installed.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
CARRIER_OPTIONS	String(256)	Variable	The carrier-specific options; defined separately for each supported carrier. By default, this parameter is set from the environment variable *$CARRIER_OPTIONS*.	No
CONSOLE_LOGGING_ENABLED	Boolean	TRUE	Output log messages to the console on which the Chassis Manager was started.	No
COOLING_EXHAUST_SENSOR_LIST	String(32)	“” (undefined)	This parameter is used in certain chassis-specific cooling algorithms and specifies the exhaust temperature sensors of the chassis. The format of the parameter is a comma-separated list of tuples in the form “<owner-address>/<sensor-number>”. The sensor LUN must be 0. For example, the value *"0x20/120,0x20/121"* indicates that sensors 120 and 121 on the logical Chassis Manager (IPMB address 20h) represent the exhaust temperature sensors of the chassis.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
COOLING_FAN_DECREASE_TIMEOUT	Number	0	The minimum timeout between successive decrements of the fan speed during operation of the cooling algorithm in Normal state. Should be a multiple of *COOLING_POLL_TIMEOUT*; if not, it is rounded up to the next multiple. If the parameter is omitted or set to *0*, this timeout is equal to *COOLING_POLL_TIMEOUT*.	Yes
COOLING_FAN_INCREASE_TIMEOUT	Number	0	The minimum timeout between successive increments of the fan speed during operation of the cooling algorithm in Minor Alert state. Should be a multiple of *COOLING_POLL_TIMEOUT*; if not, it is rounded up to the next multiple. If the parameter is omitted or set to *0*, this timeout is equal to *COOLING_POLL_TIMEOUT*.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
COOLING_INTAKE_SENSOR_LIST	String(32)	“” (undefined)	This parameter is used in certain chassis-specific cooling algorithms and specifies the intake temperature sensors of the chassis. The format of the parameter is a comma-separated list of tuples in the form “<owner-address>/<sensor-number>”. The sensor LUN must be 0. For example, the value *"0x20/124,0x20/125"* indicates that sensors 124 and 125 on the logical Chassis Manager (IPMB address 20h) represent the intake temperature sensors of the chassis.	Yes
COOLING_KEEP_POWERED_OFF_FRUS_IN_M1	Boolean	FALSE	If set, the FRUs that are powered off by the Chassis Manager due to a critical temperature stay in state M1 and are not automatically activated and powered on when the temperature condition goes away; manual intervention is needed to reactivate each such FRU.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
COOLING_NO_ALL_FRUS_POWER_DOWN_IN_CRITICAL_ALERT	Boolean	FALSE	Do not power down the whole chassis if a critical thermal alert is reported by a chassis-wide temperature sensor or a sensor that cannot be associated with a specific FRU. Specific FRUs can still be powered down in the case of a critical thermal alert in this mode. This mode should be used with caution.	Yes
COOLING_NO_POWER_DOWN_IN_CRITICAL_ALERT	Boolean	FALSE	Do not power down FRUs that experience critical thermal alerts. This mode should be used with extreme caution since it may cause critical overheating and damage to FRUs in the chassis.	Yes
COOLING_MANAGEMENT	String(64)	“” (undefined)	If specified, the name of the shared library that implements cooling management. The actual name of the library is *libcooling_<xxx>.so*, where *<xxx>* is the value of this configuration parameter. This library is dynamically loaded by the Chassis Manager and must be located in */var/bin* or */lib*.	No
COOLING_POLL_TIMEOUT	Number	30	The maximum time (in seconds) between successive invocations of the cooling monitoring and management thread.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
DEFAULT_GATEWAY_IP_ADDRESS	IP-address	None	The default IP address used for the gateway for chassis-external (RMCP-based) communication, if the corresponding parameter is set to *0.0.0.0* in the IPMI LAN Configuration Parameters for channel 1. If a non-zero gateway IP address is provided in the LAN Configuration Parameters, the value provided in the Chassis Manager configuration file is ignored.	No
DEFAULT_GATEWAY_IP_ADDRESS2	IP-address	None	The default IP address used for the gateway for chassis-external (RMCP-based) communication on the second network interface, if the corresponding parameter is set to *0.0.0.0* in the IPMI LAN Configuration Parameters for channel 2. If a non-zero gateway IP address is provided in the LAN Configuration Parameters, the value provided in the Chassis Manager configuration file is ignored.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
DEFAULT_RMCP_IP_ADDRESS	IP-address	None	The default IP address used for chassis-external (RMCP-based) communication; it is switched over between the redundant instances of the Chassis Manager. This IP address is used only if the corresponding parameter is set to *0.0.0.0* in the IPMI LAN Configuration Parameters for channel 1 and in the Chassis Manager IP Connection record in Chassis FRU Information. If a non-zero IP address is provided in the LAN Configuration Parameters and/or Chassis FRU Information, the value provided in the Chassis Manager configuration file is ignored.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
DEFAULT_RMCP_IP_ADDRESS2	IP-address	None	The default IP address used for chassis-external (RMCP-based) communication on the second network interface; it is switched over between the redundant instances of the Chassis Manager. This IP address is used only if the corresponding parameter is set to *0.0.0.0* in the IPMI LAN Configuration Parameters for channel 2. If a non-zero IP address is provided in the LAN Configuration Parameters, the value provided in the Chassis Manager configuration file is ignored.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
DEFAULT_RMCP_NETMASK	IP-address	Variable	The network mask for the network adapter used for RMCP communication. This mask is used only if the corresponding parameter is set to *0.0.0.0* in the IPMI LAN Configuration Parameters for channel 1 and in the Chassis Manager IP Connection record in Chassis FRU Information. The default value depends on the class of the default IP address used for chassis-external (RMCP-based) communication. (see parameter *DEFAULT_RMCP_IP_ADDRESS*). For example, for an IP address of class C, this parameter is set to *255.255.255.0*	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
DEFAULT_RMCP_NETMASK2	IP-address	Variable	The network mask for the second network adapter used for RMCP communication. This mask is used only if the corresponding parameter is set to *0.0.0.0* in the IPMI LAN Configuration Parameters for channel 2. The default value depends on the class of the default IP address used for chassis-external (RMCP-based) communication. (see parameter *DEFAULT_RMCP_IP_ADDRESS2*). For example, for an IP address of class C, this parameter is set to *255.255.255.0*	No
DEFAULT_VLAN_ID	Number	0	The default Virtual LAN ID used for the first LAN channel. This value is used only as the default value for the corresponding LAN configuration parameter; once LAN configuration parameters are defined and stored on the ChMM, the value provided in the Chassis Manager configuration file is ignored.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
DEFAULT_VLAN_ID2	Number	0	The default Virtual LAN ID used for the second LAN channel. This value is used only as the default value for the corresponding LAN configuration parameter; once LAN configuration parameters are defined and stored on the ChMM, the value provided in the Chassis Manager configuration file is ignored.	Yes
DETECT_DEADLOCKS	Boolean	TRUE	This variable turns on the deadlock detection in CLI and RMCP server facilities in the Chassis Manager. The detection is based on an internal watchdog that must be periodically strobed by the threads serving CLI and RMCP requests. If one of the threads fails to strobe the internal watchdog, the actual watchdog does not get strobed, and ultimately the ChMM resets, initiating a failover to the backup Chassis Manager.	No
DEVICE_POLL_TIMEOUT	Number	15	The time (in seconds) between successive polls of the System IPMB devices by the Chassis Manager via sending the “Get Device ID” command to them.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
DHCP_FOR_RMCP_ONLY	Boolean	FALSE	If this variable is set, only the RMCP IP addresses are assigned via the DHCP mechanism; private IP addresses of the ChMMs are not touched by DHCP.	Yes
DHCP_REQUEST_OPTION_LIST	Boolean	FALSE	This variable causes the DHCP client to request an explicit list of options via parameter request list option 55. By default, the DHCP request does not specify desired options, so the DHCP server does not necessarily return such information.	Yes
DHCP_SERVER_ADDRESS	IP-address	None	This parameter is the IP address of the DHCP server; it applies only if the variable *USE_DHCP* is *TRUE*. If this parameter is omitted or set to *0.0.0.0*, and the *USE_DHCP* variable is *TRUE*, the Chassis Manager accepts address information from any DHCP server that responds to its broadcast discovery request.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
DHCP_SPECIAL_CLIENT_ID_FORMAT	Boolean	FALSE	If this parameter is set to *TRUE,* the DHCP Client ID has a special format that is used by specific customers for Pigeon Point products. By default this variable is *FALSE* and a standard Pigeon Point Client ID format is used, as described in section 3.4.7 of this document.	Yes
DISABLE_SESSION_AUDIT_EVENTS	Number	0	This bit mask controls whether assertion events are generated by the Session Audit sensor in response to specific state changes of the sensor. By default, assertion events are generated by the Session Audit sensor (sensor 139, IPMB address 20h) and logged to the SEL for all defined states. The bits of the bit mask have the following meaning: [0] - 1b disables generation of "Session Activated" events. [1] - 1b disables generation of "Session Deactivated" events. [2] - 1b disables generation of "Invalid username or password" events. [3] - 1b disables generation of "Invalid password disable" events.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
ENABLE_DIRECT_SHELF_FRU_WRITE	Boolean	FALSE	This variable controls whether the Chassis Manager allows direct writes to the Chassis FRU Information (FRU Device ID #254 on the IPM Controller at 20h on System IPMB) via the IPMI command "Write FRU Data". By default, direct writes are prohibited (as mandated by PICMG 3.0 R2.0 ECN-002, which requires that Chassis FRU Information writes use a special locking protocol so that only one writer is active at once). Setting this variable to *TRUE* enables direct writes for compatibility with System Manager applications that rely on the pre-ECN-002 behavior.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
ENABLE_LOCKS_LOGGING	Boolean	TRUE	If this variable is *TRUE* (default value), the Chassis Manager records all requests to acquire and release locks (mutexes) in an internal memory area. This information can be subsequently recovered and analyzed by the analysis tool *dumplog* in case of a deadlock or another synchronization violation. Setting this variable to *FALSE* makes post-mortem lock analysis impossible, but substantially improves the performance of the Chassis Manager. See section 4.9 for further background.	No
ENABLE_NEW_DHCPV4_CLIENT	Boolean	TRUE	This variable determines which DHCPv4 client is used by the Chassis Manager if *USE_DHCP* is set to *TRUE*. By default, the new DHCPv4 client is used. See section 3.4.6 for more information.	No
ENABLE_RTC_TRICKLE_CHARGER	Boolean	FALSE	If this variable is *TRUE*, the trickle charge feature of the RTC DS1339 device on the ChMM is enabled. This setting is required for ChMM carriers that connect a super cap to the RTC backup power supply input (VBAT or VBACKUP_RTC).	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
EXIT_IF_HEALTHY_LOST_IN_STANDALONE_MODE	Boolean	FALSE	This variable defines what to do if the Chassis Manager runs without backup and detects a loss of the local Healthy bit. If this variable is *TRUE*, the Chassis Manager exits, the ChMM reboots and the Chassis Manager is restarted. If this variable is *FALSE*, the Chassis Manager sets the Healthy bit and continues operation	Yes
EXIT_IF_NO_SHELF_FRU	Boolean	FALSE	If *TRUE*, the Chassis Manager exits (probably resetting the ChMM) if no Chassis FRU Information can be found.	No
EXTERNAL_EVENT_HANDLER	String(255)	“”	This is the path to an executable file (or a script file) on the Chassis Manager that performs local handling of events via PEF.	No
FAN_FULL_SPEED_DELAY	Number	0	The delay in seconds after Chassis Manager startup or after a switchover, during which the cooling algorithm does not check the number of present fan trays, giving the existing fan trays enough time to activate. This applies to the carriers where cooling management raises the fan speed to maximum if the actual number of fan trays in the chassis is fewer than what is specified in the Chassis Address Table.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
FAN_LEVEL_STEP_DOWN	Number	1	The number of fan steps by which the fan speed is decreased during operation of the cooling algorithm in the Normal state. This parameter may be overridden by a ChMM carrier-specific cooling algorithm.	Yes
FAN_LEVEL_STEP_UP	Number	1	The number of fan steps by which the fan speed is increased during operation of the cooling algorithm in the Minor Alert state. This parameter may be overridden by a carrier-specific cooling algorithm.	Yes
HPDL	Boolean	FALSE	Turns on HPDL support in the Chassis Manager. The carrier and chassis HPDL data and SDRs are taken from the FRU Information or from the files and are used to define the behavior of the platform, plus the number and types of managed FRUs and sensors.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
HPDL_NETWORK_ELEMENT_ID_EEPROM_OFFSET	Number	0	If non-zero, specifies the offset to the Network Element ID data structure in Chassis FRU Information EEPROMs. This 33-byte data structure consists of the three separate 11-byte Network Element IDs and is transparent for the Chassis Manager but can be read and modified with CLI commands. A positive value of this parameter indicates the offset from the beginning of the EEPROM, a negative value means the offset from the end of the EEPROM. Zero value indicates the absence of the Network Element ID. This parameter makes sense only for HPDL-based systems and only if local EEPROMs are used to store Chassis FRU Information.	Yes
HPDL_ON_SUBSIDIARY_FRUS	Boolean	TRUE	Turns on support of HPDL information stored on subsidiary FRUs. If *TRUE*, the Chassis Manager looks for HPDL data and SDRs in the FRU Information of its subsidiary FRUs. If these data are found for a specific FRU, they are used to substitute definitions for that FRU from the carrier or chassis HPDL data and/or SDRs.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
HPM3_ENABLE	Boolean	TRUE	Enables HPM.3-compliant format of DHCP client IDs when retrieving addressing information for the logical and physical Chassis Manager(s) over DHCP. If this parameter is *FALSE*, the Chassis Manager uses the old format of DHCP client IDs; this mode of operation is compatible with previous releases of the Chassis Manager that did not support HPM.3.	No
IGNORE_ FAILED_ DIRECTED_ POWER_DOWN	Boolean	TRUE	This parameter tells the Chassis Manager to ignore board deactivation failure(s) during a Critical Alert and not to deactivate all boards in the chassis in this case. If this parameter is *FALSE*, all boards in the chassis are deactivated, if board(s) that caused the Critical Alert cooling state fail to deactivate.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
INITIALIZATION_SCRIPT	String(256)	“”	This parameter specifies a command line that is executed by the active Chassis Manager during initialization. This initialization script can perform additional platform-specific initialization of the Chassis Manager; when it is started, the Local Healthy and Active bits in HRI are already set. The Chassis Manager waits for the completion of the initialization script. The initialization script is run only during the startup of the Chassis Manager; it is not run after a switchover.	No
INITIAL_FAN_LEVEL	Number	15	This parameter specifies the initial fan level that the Chassis Manager applies to fan trays. Usually fan level values are in 0.15 range, where *0* is the slowest, and *15* is the fastest possible fan speed.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
INITIAL_SLOW_LINK_DELAY	Number	0	The initial delay, in seconds, before the Chassis Manager starts testing the integrity of the physical network link between the Chassis Manager and the System Manager (the RMCP link; see the description of the configuration parameter *SWITCHOVER_TIMEOUT_ON_BROKEN_LINK*). A non-zero delay can be used to accommodate slow network links that need significant time to initialize after chassis power up.	Yes
INNER_SEQUENCE_NUMBER_IN_SEND_MSG_RESPONSE	Boolean	TRUE	This variable controls which sequence number is used in the response to a "Send Message" command bridged from LAN to IPMB. If *TRUE*, the sequence number of the command encapsulated in the Send Message request is used. If *FALSE*, the sequence number of the Send Message request itself is used. According to a clarification being proposed for the latest version of the IPMI specification, the first variant is correct, while the Chassis Manager historically used the second variant.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
IPMB_ADDRESS	Number	0	The IPMB address of the Chassis Manager, overriding the hardware address. The value of *0* causes the Chassis Manager to read the hardware address from hardware and set IPMB address to hardware address * 2.	No
IPMB_LINK_ ISOLATION_ TIMEOUT	Number	-1	In radial chassis, if an IPMB link is disabled due to the isolation algorithm, the link is automatically enabled after this time interval (in seconds). *–1* (the default) stands for "forever".	Yes
IPMB_LOCAL_ FAILURE_ RETRIES	Number	10	In the case of a local IPMB failure (as reported by the IPMB driver), the number of times the IPMB operation is retried. After the specified number of retries is exhausted on IPMB-A and is also exhausted on IPMB-B (if IPMB-B is implemented in the chassis), a switchover to the backup Chassis Manager takes place.	No
IPMB_RETRIES	Number	3	The number of attempts to send an IPMB request before finally giving up, if no response is received to this request.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
IPMB_RETRY_TIMEOUT	Number	0	The amount of time (in seconds) that the Chassis Manager waits for a response after sending an IPMB request, before retrying the request.	No
IPMB_RETRY_TIMEOUT_MSEC	Number	500	The millisecond part of the retry timeout value that combines with the *IPMB_RETRY_TIMEOUT* value to form the full timeout value. This variable allows retry timeout values less than a second, given a zero value for the configuration variable *IPMB_RETRY_TIMEOUT*. Using an overall IPMB timeout value less than 250 msec is not recommended.	No
IPMC_PRESERVE_ON_REVISION_CHANGE	Boolean	TRUE	Setting this variable to *TRUE* preserves the Chassis Manager's identification of an IPM Controller after a firmware upgrade if only the Firmware Revision and/or Auxiliary Firmware Revision information is changed. If the variable is set *FALSE*, any change in "Get Device ID" response data during the Chassis Manager's regular polls is considered to signal the presence of a different IPM Controller.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
ISOLATE_MUX_ADDRESS	Number	0x70	The 7-bit I^2C multiplexer address (on platforms where SHMM_GPIO8 is used to control access from the Chassis Manager to the multiplexer on the master-only I^2C bus, via the original mechanism that is not based on HPDL).	No
ISOLATE_MUX_BUS	5	Number	The number of the logical I^2C bus on which the I^2C switch/multiplexer resides (on platforms where SHMM_GPIO8 is used by the Chassis Manager to reset the I^2C switch/multiplexer via the original mechanism that is not based on HPDL).	No
ISOLATE_MUX_IGNORE_COUNT	Number	10	This parameter instructs the isolation algorithm to skip this number of accesses to a faulty bus before trying to enable it (on the platforms where SHMM_GPIO8 is used to control access from the Chassis Manager to the multiplexer on the master-only I^2C bus, via the original mechanism that is not based on HPDL)	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
ISOLATE_MUX_IGNORE_TIME	Number	10	This parameter instructs the isolation algorithm to avoid accessing the faulty bus and immediately return a failure for specified number of seconds after a failure detection (on the platforms where SHMM GPIOs are used to control access from the Chassis Manager to multiplexers on the master-only I^2C bus, via an HPDL-based mechanism).	No
ISOLATE_MUX_ON_GPIO8	Boolean	FALSE	Must be set to *TRUE* for platforms where SHMM_GPIO8 is used by the Chassis Manager to reset an I^2C switch/multiplexer on an I^2C bus (but must be *FALSE* if the HPDL-based reset mechanism is used).	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
ISOLATE_MUX_SWITCHOVER_THRESHOLD	Number	0	If this parameter is non-zero and *ISOLATE_MUX_ON_GPIO8* is *TRUE*, the I^2C isolation algorithm will perform a switchover to the backup Chassis Manager if a fault exists on the main I^2C bus after the I^2C switch is reset. In this case, this parameter specifies the number of consecutive failed I^2C switch resets that are attempted, after which, if the main I^2C bus remains inaccessible, a switchover takes place and the active Chassis Manager reboots. This parameter is only applicable on platforms where SHMM GPIO8 is used by the Chassis Manager to reset an I^2C switch via the original mechanism that is not based on HPDL.	No
LOCAL_SHELF_FRU	Boolean	TRUE	Create a local FRU #1 on the Chassis Manager that exposes the Chassis FRU Information (obtained from the file *`/var/nvdata/chassis_fru_info`*).	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
M7_TIMEOUT	Number	-1	The maximum time (in seconds) for a FRU to stay in M7 state; after the expiration of this time the FRU is automatically transitioned to M0. *–1* (the default) stands for "forever". Setting this parameter to *0* completely prevents FRUs from going into state M7.	Yes
MAX_ALERT_POLICIES	Number	64	The maximum number of PEF Alert Policies available. The maximum allowed value for this parameter is *128*.	No
MAX_ALERT_STRINGS	Number	64	The maximum number of PEF Alert Strings available.	No
MAX_DEFERRED_ALERTS	Number	32	The maximum number of outstanding PEF alerts.	No
MAX_EVENT_FILTERS	Number	64	The maximum number of PEF event filters available.	No
MAX_EVENT_SUBSCRIBERS	Number	64	The maximum number of entities that can simultaneously subscribe to receive event notifications from the Chassis Manager.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
MAX_EVENT_SUBSCRIBER_IDLE_TIME	Number	60	The maximum timeout for an event subscriber, in seconds, between the moment when an event arrives and the moment when the subscriber retrieves this event from the Chassis Manager. If this timeout is exceeded, the subscriber is considered dead and is automatically unregistered.	Yes
MAX_INCOMING_IPMB_REQUESTS	Number	128	The size of the internal Chassis Manager queue for incoming IPMB requests. Incoming IPMB requests are stored in this queue before processing.	No
MAX_NODE_BUSY_TRANSMISSIONS	Number	255	The maximum number of transmissions of an IPMB command if the receiver always returns the completion code Node Busy in response.	Yes
MAX_OEM_FILTERS	Number	16	The maximum number of PEF OEM event filters available.	No
MAX_PENDING_EVENT_NOTIFICATIONS	Number	1024	The maximum number of outstanding event notifications for each active subscriber.	No
MAX_PENDING_IPMB_REQUESTS	Number	64	The maximum number of pending IPMB requests awaiting response.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
MAX_ SEL_ENTRIES	Number	1024	The maximum number of entries in the System Event Log. CAUTION: Large values of this parameter can significantly degrade Chassis Manager performance.	No
MAX_SESSIONS	Number	32	The maximum number of simultaneous IPMI sessions. The maximum allowed value for this parameter is 63.	No
MAX_USERS	Number	32	The maximum number of IPMI users. The maximum allowed value for this parameter is 63.	No
MIN_FAN_LEVEL	Number	1	The minimum fan level; the cooling management code does not reduce the fan level of any fan below this value when controlling the fan level automatically	Yes
MIN_ SHELF_FRUS	Number	1	The minimum number of Chassis FRU Information instances in the chassis that the Chassis Manager must detect to start up successfully.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
NORMAL_STABLE_TIME	Number	3600	The time in seconds for which the Chassis Manager preserves the minimum fan level dynamically found in Normal mode (that is, the minimum fan level that does not cause thermal alerts). After this time expires, the cooling algorithm decreases the minimum fan level, if possible, to allow the chassis to decrease the fan level if the thermal load in it has also decreased.	Yes
NO_M0_M1_EVENT_AT_STARTUP	Boolean	FALSE	If *TRUE*, the physical Chassis Manager does not send an M0->M1 event when the Chassis Manager application is restarted on the ChMM; the event is sent only the first time after the ChMM is plugged into a chassis. The default value is *FALSE* for backward compatibility.	No
PEF_USE_KEYED_ALARMS	Boolean	FALSE	If *TRUE*, the Chassis Manager uses special methods of managing TELCO alarms raised via PEF as OEM actions. Each event condition that causes a TELCO alarm is counted separately in this case, so that if a specific alarm is raised due to multiple event conditions, it stays raised while any of these conditions stay asserted.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
PET_FORMAT	Number	0	Specifies the format of the Platform Event Traps that are sent by the Chassis Manager as the Alert action initiated by event processing in the Platform Event Filtering facility. The values are defined as follows: *0* = the default IPMI format defined by IPMI Platform Event Trap Format v1.0 specification. *1* = plain text format; all the event details are sent as plain ASCII text in a single variable. *2* = multi-variable format; each event field is encoded as a separate variable defined in PPS-PET-MIB.	Yes
PET_OEM_WITH_SEVERITY_STRING	Boolean	FALSE	If *TRUE*, the event severity and alert string is added to a trap message in the nVent OEM (Plain Text and Multi-Variable) PET formats. The default IPMI PET format always includes these fields from the corresponding event filter.	Yes
PET_PORT	Number	162	Specifies destination port number for the Platform Event Trap datagrams. The default UDP port number is 162.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
PET_SNMP_VERSION	Number	1	Specifies the SNMP version for traps generated by the Chassis Manager. Currently supported versions are *1* (the default) and *2*.	Yes
POWER_UNLISTED_FRUS	Boolean	TRUE	Allow the FRUs not listed in the power management table in the Shelf FRU Information to be activated and powered up.	No
PREFER_CHASSIS_HPDL_SDRS_FROM_FILE	Boolean	FALSE	If *TRUE*, the preferred source of chassis HPDL configuration data is files on the ChMM, not records in Chassis FRU Information. Default is *FALSE*.	No
PROPAGATE_RMCP_ADDRESS	Boolean	FALSE	If *TRUE*, the active Chassis Manager propagates the RMCP IP address to the backup Chassis Manager, which configures the network interface specified by the *RMCP_NET_ADAPTER* variable using that IP address, but with the least significant bit inverted.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
REAPPLY_POWER_MAX_COOLING_STATE	String(16)	"NORMAL"	Sets the most severe chassis thermal state in which the Chassis Manager can power a FRU up after that FRU has been powered down due to a Critical thermal alert. Possible values of this variable are *NORMAL*, *MINOR_ALERT*, *MAJOR_ALERT*. The *CRITICAL_ALERT* value is not supported, since it could lead to oscillations caused by immediate reactivation of the problematic FRU.	No
REDUNDANCY_COMPRESSION_THRESHOLD	Number	-1	This parameter controls compression of messages sent over the redundancy interface. A message is sent in compressed format if its size exceeds the parameter value. The value *-1* turns compression off. Using message compression can improve the speed of redundancy information transfer on relatively slow redundancy links.	No
REDUNDANCY_ENABLED	Boolean	TRUE	Run the Chassis Manager in redundant mode.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
REDUNDANCY_NET_ADAPTER	String(16)	"eth0"	The name of the network adapter used for communication between redundant instances of the Chassis Manager. *usb0* is the recommended value if USB network interfaces are used for redundancy.	No
REDUNDANCY_NET_ADAPTER2	String(16)	"" (undefined)	The name of the second network adapter used for communication between redundant instances of the Chassis Manager (if the dual USB network interface is used for this purpose). *usb1* is the recommended value if USB network interfaces are used for redundancy. If the configuration variable *REDUNDANCY_NET_ADAPTER* is set to *usb0*, this variable defaults to *usb1*.	No
REDUNDANCY_NETMASK	IP-address	0.0.0.0	The netmask to assign to the redundancy IP address; by default (if 0), the netmask is determined automatically from the class of the IP address. *255.255.255.128* is the recommended value if USB network interfaces are used for redundancy.	No
REDUNDANCY_PORT	Number	1040	The TCP port used for interactions between redundant instances of the Chassis Manager.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
REDUNDANT_ IP_ADDRESS	IP-address	None	The IP address used for redundant communications. This address actually specifies a pair of IP addresses that differ only in the least significant bit. They are assigned to redundant Chassis Managers according to their hardware addresses.	No
RESERVATION_ RETRIES	Number	10	The maximum number of times the Chassis Manager retries the "Reserve Device SDR" command.	Yes
RMCP_NET_ ADAPTER	String(16)	"eth0:1"	The name of the network adapter used for RMCP-based communication.	Yes
RMCP_NET_ ADAPTER2	String(16)	None	The name of the alternate network adapter used for RMCP-based communications, if cross-connect links are supported by the hardware. *eth1* is the recommended value if the parameter is specified.	Yes
RMCP_PLUS_ ONLY	Boolean	FALSE	If this variable is *TRUE*, support of the regular RMCP protocol (from IPMI 1.5) is disabled on the Chassis Manager and only the RMCP+ protocol (from IPMI 2.0) is supported. (This mode may be needed for security reasons.)	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
RMCP_WITHOUT_SHELF_FRU	Boolean	FALSE	RMCP is available in the absence of the Chassis FRU Information only if this configuration parameter is defined and set to *TRUE*.	No
SAVE_CRASH_INFORMATION	Boolean	FALSE	If *TRUE*, the Chassis Manager saves crash information in the `/var/crash` directory in addition to setting a corresponding reboot reason. The following system signals are treated as indicators of a Chassis Manager crash: SIGSEGV, SIGABRT, SIGILL, SIGFPE, SIGBUS. The saved crash information includes a compressed system log and core file.	No
SDR_READ_RETRIES	Number	3	The maximum number of times the Chassis Manager retries the “Read Device SDR” command.	Yes
SEL_FILE_COMPRESSION_ENABLED	Boolean	TRUE	If *TRUE*, the Chassis Manager compresses the SEL before writing it to the ChMM Flash; this reduces the size of the file that is periodically written to the Flash and thus increases the lifetime of the Flash.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
SEL_FILE_JOURNALING_ENABLED	Boolean	TRUE	If *TRUE*, the Chassis Manager uses journaling when writing the SEL to the ChMM Flash; this reduces the size of the data periodically written to the Flash and thus increases the lifetime of the Flash. The reduction is even greater if SEL compression is simultaneously enabled.	No
SEL_FILE_WRITE_DELAY	Number	3	The minimum number of seconds between subsequent updates of the SEL file on the ChMM Flash. Setting this number to a higher value causes SEL updates to be less frequent; this reduces the total amount of data written to the Flash and thus increases the lifetime of the Flash, but increases the probability of event loss if both redundant Chassis Managers go out of service.	No
SEL_HIGH_WATERMARK	Number	0	This value is the “high watermark” for the algorithm that controls automatic purging of the SEL; if the actual percentage of free entries in the SEL falls below this value, or the SEL overflows, the Chassis Manager starts a thread that purges old records from the SEL in order of decreasing age.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
SEL_LOW_WATERMARK	Number	0	This value is the "low watermark" for the algorithm that controls automatic purging of the SEL; if the thread that purges old records from the SEL starts, it purges records until the percentage of occupied entries in the SEL falls below this value.	Yes
SENSOR_POLL_INTERVAL	Number	1	The time (in seconds) between successive polls of local Chassis Manager sensors by the Chassis Manager.	No
SESSION_SEQUENCE_WINDOW	Number	128	This is the window of acceptable RMCP sequence numbers; the wider this window is, the more tolerant is the Chassis Manager to RMCP packets being dropped during transfer. If the difference in the sequence numbers of a received packet and the previous packet exceeds the window size, the Chassis Manager closes the RMCP session as a corrupted one.	No
SHELF_FRU_IN_EEPROM	Boolean	TRUE	If *TRUE*, the Chassis FRU Information is retrieved from EEPROMs on the backplane in a carrier-specific way; if *FALSE*, the Chassis FRU Information is obtained from a file on the Flash file system.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
SHELF_FRU_IPMB_SOURCE1	Number	0	If defined (non-zero), specifies the IPMB address of the first designated source of Chassis FRU Information in the chassis. (Chassis FRU Information is located at FRU 1.) If this value is defined, the search for the Chassis FRU on the IPMB is limited to the designated sources only.	No
SHELF_FRU_IPMB_SOURCE2	Number	0	If defined (non-zero), specifies the IPMB address of the second designated source of Chassis FRU Information in the chassis. (Chassis FRU Information is located at FRU 1.) If this value is defined, the search for the Chassis FRU Information on the IPMB is limited to the designated sources only.	No
SHELF_FRU_TIMEOUT	Number	5 or 20	The time interval (in seconds) during initialization that the Chassis Manager waits for Chassis FRU Information devices to be detected. If the configuration variable *SHELF_FRU_IN_EEPROM* is *TRUE*, the default value is *20*. The actual time is set to *20* if the value of this parameter is less than *20*	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
SHELF_ MANAGER_ CONFIGURATION _IN_SHELF_ FRU_INFO	Boolean	FALSE	If *TRUE*, instructs the Chassis Manager to retrieve configuration parameters from a special set of records in the Chassis FRU Information. If these parameters are successfully retrieved, they override the values from configuration files. Not all configuration variables can be overridden from the Chassis FRU Information; for example, this variable cannot be overridden (for obvious reasons) and, to be effective, must be specified in the configuration file.	No
SHORT_SEND_ MSG_RESPONSE	Boolean	TRUE	Determines the type of the "Send Message" response provided by the Chassis Manager: required by the latest version of the IPMI specification (if *TRUE*) or compatible with the previous versions of the Chassis Manager (if *FALSE*).	No
SWAPPED_ CROSS_ CONNECTS	Boolean	FALSE	Swaps the names of network adapters used for cross-connects on the ChMC with the odd hardware address.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
SWITCHOVER_ON_BROKEN_LINK_BACKUP_DELAY	Number	0	This parameter specifies the delay on the backup side during a switchover caused by a broken physical network link. When the active Chassis Manager requests the switchover, the backup Chassis Manager agrees to it (if its physical network link is present) only after this number of seconds passes, following the first request from the active Chassis Manager. A non-zero value for this parameter can be used to prevent switchovers in scenarios where a switch module activates the active Chassis Manager link several seconds after the backup Chassis Manager link.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
SWITCHOVER_ON_HANDLE_OPEN	Boolean	FALSE	If *TRUE*, switchover-related behavior of the active Chassis Manager is affected by the operational state of its physical IPM Controller, as follows: - If the physical IPM Controller on the active Chassis Manager goes to state M1 due to to programmatic deactivation, a switchover to the backup Chassis Manager is initiated; - If the physical IPM Controller on the active Chassis Manager goes to state M5 and there is no available backup Chassis Manager, the physical IPM Controller is not deactivated and stays in M5 indefinitely.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
SWITCHOVER_TIMEOUT_ON_BROKEN_LINK	Number	-1	This parameter affects when or whether the Chassis Manager initiates a switchover when the physical network link between the Chassis Manager and the System Manager (the RMCP link) is broken. If the link remains broken for at least the number of seconds given in this parameter, a switchover takes place; if the link is restored during this timeout period, no switchover takes place. If the value of this parameter is *-1*, no automatic switchovers take place on broken RMCP links.	Yes
SYSLOG_LOGGING_ENABLED	Boolean	TRUE	Output log messages to the system log.	No
SYSTEM_MANAGER_TRUNCATES_SEL	Boolean	FALSE	If *TRUE*, the Chassis Manager algorithm for truncating the SEL automatically is disabled; the System Manager is responsible for truncating the SEL by monitoring the value of the sensor SEL State of the type Event Logging Disabled on the Chassis Manager, and removing events from the SEL by sending the "Delete SEL Entry" command to the Chassis Manager.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
TACHOMETER_ THRESHOLD_ UPDATE_DELAY	Number	15	This parameter applies to fan trays managed by the Chassis Manager and implementing adaptive setting of tachometer thresholds depending on the current fan level. This parameter specifies the delay in seconds between setting a new fan level that is higher than the current fan level and increasing the tachometer threshold value; the delay is needed to allow the fans to reach the higher speed.	Yes
TASKLET_ RETRIES	Number	3	This parameter specifies the total number of Chassis Manager retries to execute a tasklet (activation, deactivation, getting information, etc.) if it encounters a transient failure when doing that. If the Chassis Manager fails to perform a tasklet after the *TASKLET_RETRIES* retries, it gives up. Increasing this value may improve the robustness of the Chassis Manager operation, but degrade its performance.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
TIMEPROTO	String(16)	“”	Protocol used to retrieve time from a network time server; possible values are *ntp* and *rdate*. The default value is an empty string which makes the script to use *rdate*.	Yes
TIMESERVER	IP-address	None	IP address of the time server for synchronization at runtime. If this variable is not specified, time is extracted from the hardware clock at chassis startup.	Yes
TURBO_MODE_ MIN_FAN_ FAILURES	Number	1	This parameter specifies the number of tachometer underspeed faults (crossing Major or Critical thresholds) cause the remaining fans to go to TURBO mode (that is, run at full speed) in the default HPDL cooling management algorithm. If the chassis implements zoned cooling, the specified number of tachometer faults is counted relative to each zone and setting TURBO mode affects only the fan trays that participate in cooling of the corresponding zone.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
TURBO_MODE_MIN_MISSING_FAN_TRAYS	Number	1	This parameter specifies the minimum number of missing/inactive fan trays that triggers the remaining fan trays to go to TURBO mode (that is, run at full speed) in the default HPDL cooling management algorithm. If this parameter is set to *0*, this feature of setting fans to their maximum speed (due to missing/inactive fan trays) is disabled, entirely. If the chassis implements zoned cooling, the specified number of missing/inactive fan trays is counted relative to each zone and setting TURBO mode affects only the fan trays that participate in cooling of the corresponding zone	Yes
UNCONDITIONAL_SDR_REREAD_ON_VERSION_CHANGE	Boolean	FALSE	If the value of this parameter is *TRUE* the Chassis Manager unconditionally re-reads SDRs from an IPMC when it receives a Version Change event from that controller, even if the Sensor Population Change Indicator in the "Get Device SDR Info" response does not change. This is done for the benefit of VPX modules or other intelligent FRUs that are not fully IPMI-compliant in this respect.	Yes

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
USE_DHCP	Boolean	FALSE	Requests assignment of RMCP-accessible and private IP addresses for the Chassis Manager from a DHCP server; the configuration parameter *DHCP_SERVER_ADDRESS* can be used to specify the IP address of the DHCP server.	Yes
USE_SECOND_CHANNEL	Boolean	FALSE	This parameter applies only if two network interfaces on the ChMM are used for RMCP communication. If *TRUE*, the two network interfaces on the ChMM are used in parallel mode; if *FALSE*, they are used in redundant mode.	Yes
VERBOSITY	Number	7	The Chassis Manager verbosity level.	No
VERBOSITY_CONSOLE	Number	N/A	The Chassis Manager verbosity level for console output.	No
VERIFY_SHELF_FRU_CHECKSUM	Boolean	TRUE	Enable verification of checksums in Chassis FRU Information records; if set to *FALSE*, Chassis Manager ignores checksums	No
WATCHDOG_ENABLED	Boolean	TRUE	This configuration parameter is used to enable or disable the hardware watchdog timer. When enabled, the watchdog will cause the Chassis Manager to reboot if the watchdog expires.	No

Name	Type	Default	Description	Can be obtained from Chassis FRU Info
WATCHDOG_TIMEOUT	Number	2	The hardware watchdog timeout, in seconds. If the software does not strobe the watchdog at least once during this period, the hardware resets the ChMM. The supported range for this parameter is 2..30 seconds, which is valid for all supported ChMM platforms. Increasing the default value may be necessary if additional software besides the Chassis Manager application runs on the ChMM, increasing the CPU load. Also see section 4.5.3 for the possible reasons to increase the watchdog timeout. The downside of increasing the watchdog timeout is that the length of time to switch over to the backup ChMM will be also be increased in the event of a watchdog expiration.	No

By default, the configuration file variables are used automatically when the ChMM is brought up for the first time. The default configuration file imports several environment variables set by U-Boot.

Table 6: U-Boot environment variables and descriptions

U-Boot environment variable	Description
$IPADDR	Default RMCP IP Address
$IPDEVICE	Default RMCP network adapter
$IP1ADDR	Default Redundant IP Address

U-BOOT ENVIRONMENT VARIABLE	DESCRIPTION
$IP1DEVICE	Default Redundant network adapter
$GATEWAY	Default gateway used for RMCP communication

The environment variables *$CARRIER* and *$CARRIER_OPTIONS* are set by the secondary RC script. The name of this carrier-specific startup script is defined by the U-Boot environment variable *rc2*. If needed, the U-Boot environment variables can be reset to default values; refer to section 6.2.

3.3.1 Carrier-specific configuration file

After reading the common configuration file */etc/shelfman.conf*, the Chassis Manager reads the carrier-specific configuration file */etc/shelfman.conf.<carrier-name>*, where *<carrier-name>* is the name of the ChMM carrier used in the relevant chassis, in lower case characters. Settings in the carrier-specific configuration file override settings for the same variable in the common configuration file.

This mechanism allows redefinition of common settings on a carrier-specific basis. Typically, only a few critical configuration variables are defined in the carrier-specific file. For instance, the appropriate value for the *MIN_FAN_LEVEL* parameter may well be determined by the chassis architecture and the fan facilities that it implements. This mechanism allows such chassis-specific constraints to be enforced.

One result of this mechanism is that to change the effective value of a configuration parameter that is specified in the carrier-specific configuration file, the change must be made in that configuration file. A change for such a variable in the common configuration file will not have any effect.

3.3.2 Obtaining configuration variables from Chassis FRU Information

Besides carrier-specific redefinition of configuration variables, there is another mechanism for retrieving and applying configuration variables from the Chassis FRU Information. This mechanism allows redefining certain configuration variables for a specific instance or specific type of chassis.

Configuration variables are stored in the FRU Information as a sequence of OEM-type multirecords, similar to the multirecords defined in the VITA specification. These multirecords have the following Manufacturer ID (IANA Enterprise Number) assigned to nVent, Schroff GmbH (formerly Pentair Technical Products) (00400Ah)† and the manufacturer-specific record type is 10. The concatenated contents of these multirecords represent the configuration variables in the same text format as in configuration files, but optionally compressed using *gzip*.

Some restrictions apply to the configuration variables in the Chassis FRU Information. First, not all variables can be redefined from the Chassis FRU Information. Table 5 indicates, for each variable,

†This IANA Enterprise Number was originally assigned to Pigeon Point Systems, which has since been acquired by Pentair.

whether it can be redefined from the Chassis FRU Information. Second, retrieval of configuration variables from the Chassis FRU Information must be explicitly enabled for the Chassis Manager via the configuration variable *SHELF_MANAGER_CONFIGURATION_IN_SHELF_FRU_INFO*. By default, this variable is set to *FALSE*. This is done to allow the Chassis Manager to start in the case of a bad configuration stored in the Chassis FRU Information.

3.3.2.1 Using the FRU Info Compiler to place configuration variables in the Chassis FRU Information

The FRU Information Compiler is a Pigeon Point tool that generates a FRU Information image from its textual description. The image can then be stored in an appropriate EEPROM device using various techniques. The FRU Information Compiler implements special syntax that can be used to incorporate a configuration file into a FRU Information image that is being created.

For example, the following lines add a compressed configuration file *shelfman.conf.acb3.gz* to the current FRU Information image:

```
...
[nVent.Shelf Manager Configuration]
File = shelfman.conf.acb3.gz
```

3.3.2.2 Using the Chassis Manager Command Line Interface to Place Configuration Variables in the Chassis FRU Information

An extended version of the CLI command *frudataw* can be used to dynamically update Chassis FRU Information with configuration information. The corresponding configuration file (optionally compressed) must be first downloaded to the ChMM.

The corresponding command line has the following syntax:

clia frudataw -p <ipmb-address> <fru-id> <in-file>

clia frudataw -p -c <ipmb-address> <fru-id>

The parameters *<ipmb-address>* and *<fru-id>* specify the IPMB address and FRU device ID of the FRU Information to update. For the Chassis FRU Information, use IPMB address 20h and FRU device ID 254.

The parameter *<in-file>* indicates the configuration file (which is possibly compressed) to be stored in the target FRU Information.

The option *-c* removes (clears) the configuration data from the target FRU Information.

For example, the following command can be used to update the Chassis FRU Information with a new version of the configuration file:

```
#clia frudataw -p 20 254 shelfman.conf.acb3.gz
```

3.3.3 Verbosity level description

This parameter controls the verbosity of output sent by the Chassis Manager to the system log. This parameter also controls the verbosity of output to the console, if the configuration parameter

VERBOSITY_CONSOLE is omitted. The *VERBOSITY* configuration parameter is a hexadecimal bit mask, with each bit enabling output of a specific type of message:

Table 7: Verbosity configuration parameters and levels

VERBOSITY CONFIGURATION PARAMETER	VERBOSITY LEVEL
0x01	Error messages
0x02	Warning messages
0x04	Informational messages
0x08	Verbose informational messages
0x10	Trace messages (not recommended)
0x20	Verbose trace messages (not recommended)
0x40	Messages displayed for important commands sent to the IPM Controllers during their initialization (not recommended)
0x80	Verbose messages about acquiring and releasing internal locks (not recommended)

The default debug level is 0x07, enabling error, warning and informational messages.

3.3.4 Verbosity console level description

This parameter controls the verbosity of output sent by the Chassis Manager to the console. The *VERBOSITY_CONSOLE* configuration parameter is a hexadecimal bit mask, with each bit enabling output of a specific type of message, according to Table 7.

If this parameter is omitted, the value of the *VERBOSITY* configuration parameter is used to control the verbosity of output to both the system log and the console.

3.4 Setting up Ethernet

The ChMM uses two Ethernet ports, one of them being used for chassis-external access. Since RMCP is the only chassis-external interface that is required by VITA 46.11, the chassis-external Ethernet port is referenced as the RMCP port, though the other chassis-external interfaces (HTTP, TELNET, FTP) are accessible via this port as well. The other Ethernet port can be used for communication between redundant Chassis Managers, or potentially for other purposes if the redundancy link is provided by another mechanism.

Note that manual changes in the ChMM network configuration that are initiated separately from the Chassis Manager (e.g. via the `ifconfig` command) can disrupt Chassis Manager operations. The implications of any such changes must be thoroughly understood before they are undertaken.

3.4.1 Usage of the first Ethernet interface

Since the RMCP Ethernet port is directly connected to the site network, the IP address should be set up appropriately for that network.

For example, if the site uses the IP address range 192.168.0.x, the RMCP Ethernet port should be set to a unique IP address within that range such as 192.168.0.2. In a redundant ChMM setup, it is important to note that only one ChMM (the active ChMM) has the RMCP IP address enabled on the RMCP Ethernet port. The backup ChMM assigns the same IP address to the RMCP Ethernet port, but only enables it when that ChMM assumes the active role. This way, the RMCP IP address maintains availability in a failover situation. If the configuration variable *PROPAGATE_RMCP_ADDRESS* is set to *TRUE*; in that case the backup ChMM assigns a conjugate to the RMCP IP address to its RMCP Net Adapter (that is, it assigns an address that is different in the least significant bit of the fourth byte).

3.4.1.1 Assigning an additional IP address to the first network interface

In the default configuration, no IP address is assigned to the first network interface (and the ChMM is not accessible over the network) until the Chassis Manager starts and the RMCP IP address is assigned. However, it may be useful in some cases to assign an IP address to the RMCP network interface and have the ChMM accessible over the network as soon as the operating system is booted. In that case, it is also desirable that when the Chassis Manager is started, for the RMCP IP address to coexist with the originally assigned IP address rather than replacing it.

To achieve this configuration, it is necessary to instruct the Chassis Manager to assign the RMCP IP address not to the first network adapter itself (*eth0*) but to its first alias (*eth0:1*). The initial IP address is assigned in that case to the network adapter itself (*eth0*) during the start of the operating system. This initial assignment happens in the initialization script */etc/rc*; it is accomplished by: 1) enabling the U-Boot variable *rc_ifconfig* (setting it to *y*), 2) assigning the original IP address to the U-Boot variable *ipaddr*, for example:

```
ChMM700 setenv rc_ifconfig y
ChMM700 setenv ipaddr 192.168.1.240
```

and 3) changing the value of the Chassis Manager configuration parameter *RMCP_NET_ADAPTER* to *eth0:1*.

In a redundant configuration, the U-Boot variable *ipaddr* is allowed to have the same value on both ChMMs. The actual initial IP address assigned to each of the two redundant ChMMs is based on the value of *ipaddr* but is modified depending on the hardware address of the ChMM. The least significant bit of the IP address is set to the least significant bit of the hardware address. In the example above, the IP address is 192.168.1.240 for the ChMM with an even hardware address, and is 192.168.1.241 for the ChMM with an odd hardware address. This modification of the IP address can be turned off by removing the file */etc/readhwaddr*.

3.4.1.2 RMCP address propagation

There is an optional feature of the Chassis Manager that allows the backup ChMM also to be exposed on the external network with an IP address that is different from the RMCP IP address only in the least significant bit. The netmask and default gateway on the backup ChMM is the same as on the active ChMM. For example, if the RMCP IP address is 192.168.0.2, the backup ChMM has the corresponding IP address 192.168.0.3, with the same netmask and default gateway. To enable this feature, it is necessary to define the Chassis Manager configuration parameter *PROPAGATE_RMCP_ADDRESS* as *TRUE* in the Chassis Manager configuration file.

3.4.2 Usage of the second Ethernet interface

The second Ethernet interface can be dedicated for use as a private network between redundant ChMMs and used to synchronize state information between the active and backup ChMMs.

Unlike the RMCP Ethernet port, this second Ethernet interface is always enabled on both the active and backup ChMM, but with a small twist – both the active and backup ChMM specify the same IP address for the redundancy interface, but software assigns the next logical IP address to the ChMM with an odd hardware address. For instance, the default setting for the redundancy Ethernet port is 192.168.1.2. The odd-addressed ChMM assumes the address 192.168.1.3. This way the active and backup ChMM can be identically configured but still assume unique IP addresses for the redundancy Ethernet link.

When the ChMMs use a non-Ethernet network interface for communication between the redundant Chassis Managers (see section 3.4.4), the second network interface can potentially be used for other purposes, such as for support in the Chassis Manager for ChMC cross connects, in accordance with the PICMG ECN 3.0-2.0-001. In that case, the second network interface connects the Chassis Manager with one of the VPX network switch modules.

To configure the Chassis Manager to support cross-connects, it is necessary to define the configuration parameter *RMCP_NET_ADAPTER2*. In that case, if the backplane and switch modules also support cross-connects, the Chassis Manager uses the two network adapters (*RMCP_NET_ADAPTER* and *RMCP_NET_ADAPTER2*) for RMCP communication. Three usage models are available for this feature:

- active-standby,
- active-active, and
- bonded.

3.4.2.1 Active-standby usage of the two network interfaces

The two network interfaces can be used in an active-standby way. In this mode, at any given time, RMCP communication passes through only one adapter (initially this is *RMCP_NET_ADAPTER*). However, if the Chassis Manager detects that the adapter currently used for RMCP communication becomes physically disconnected from the network (link broken), it automatically switches to the other (alternate) adapter. The first adapter is turned off, the RMCP IP address is transparently

moved to the other adapter, and three ARP notifications are broadcast to notify other systems about the MAC address change. This change is transparent for the System Manager and does not break existing RMCP connections.

However, if the Chassis Manager detects that the other network adapter is also physically disconnected from the network, it does not perform the IP address switchover described above, but performs a full switchover to the backup Chassis Manager. A full switchover is also performed if the Chassis Manager detects physical disconnection of the network adapter used for RMCP communication, in non cross-connect configurations.

Detection of physical disconnection of the RMCP network adapter is controlled by the Chassis Manager configuration parameter *SWITCHOVER_TIMEOUT_ON_BROKEN_LINK*. The value of this parameter is the time interval in seconds during which the adapter stays physically disconnected, before the Chassis Manager performs an IP address switchover or full switchover. Detection and switchover is disabled if the value of this parameter is equal to *-1*.

Another configuration parameter, *INITIAL_SLOW_LINK_DELAY*, specifies the time interval from the start of the Chassis Manager, during which detection of physical disconnection is not performed. This allows using Ethernet links that are slow to start and need some time after chassis power up to establish the physical connection.

In the case of a Chassis Manager switchover (as opposed to the IP address switchover mentioned above) caused by a broken link, a special situation occurs when the first switch module is activated in a chassis with no switch modules and cross-connect Chassis Manager links.

From the physical connection perspective, this is a transition from a situation where both physical network links were broken on both Chassis Managers, to a situation where at least one physical network link is present on both Chassis Managers. However, if in this case, due to the specifics of the switch module, the link to the backup Chassis Manager comes up earlier than the link to the active Chassis Manager, an undesirable switchover may occur. To prevent a switchover in this situation, the backup Chassis Manager can delay (for a specified number of seconds) doing a switchover after the original switchover request from the active Chassis Manager, if the physical link to the backup Chassis Manager is present. If, during this time, the physical link to the active Chassis Manager also comes up, the active Chassis Manager will stop sending switchover requests, and the switchover won't happen. Otherwise, if the physical link to the active Chassis Manager remains broken after the expiration of this delay, the switchover will finally take place. The value of the delay is specified in the configuration variable *SWITCHOVER_ON_BROKEN_LINK_BACKUP_DELAY*.

Usually the parameter *RMCP_NET_ADAPTER2* is assigned the value eth1, while the value of the configuration parameter *RMCP_NET_ADAPTER* (the main network adapter used for RMCP communication) is eth0, as in the following sample.

```
RMCP_NET_ADAPTER = "eth0"
RMCP_NET_ADAPTER2 = "eth1"
```

However, other configurations are possible. For example, values `eth0:1` and `eth1:1` can be used if additional (permanent) IP addresses need to be assigned to both network interfaces.

The two network interfaces are used in active-standby mode if the configuration parameter *USE_SECOND_CHANNEL* is set to *FALSE*.

If the Chassis Manager configuration parameter *PROPAGATE_RMCP_ADDRESS* is defined as *TRUE*, active-standby management also applies to the network interface on the backup Chassis Manager, which is assigned the RMCP-derived IP address on the backup Chassis Manager. That is, if the backup Chassis Manager detects that the currently used adapter becomes physically disconnected from the network (link broken), it automatically switches to the other (alternate) adapter. The first adapter is turned off and the RMCP-derived IP address of the backup Chassis Manager is transparently moved to the other adapter.

3.4.2.2 Active-active usage of the two network interfaces

The approach to network interface redundancy outlined in the previous section seems to be insufficient for some configurations where the network connection between the Chassis Manager and the System Manager goes through several switches and may break on an Ethernet segment that is not adjacent to the ChMM.

This type of failure cannot be immediately recognized by the Chassis Manager. Checking the accessibility of the System Manager from the Chassis Manager does not seem practical in this case, since the architecture of the System Manager and its usage of IP addresses is not defined in the VITA specification and the Chassis Manager design should not artificially limit it. The solution in this case should be implemented at the System Manager level.

The command “Get Chassis Manager IP Addresses” allows the System Manager to retrieve the IP addresses exposed by the Chassis Managers, both active and backup. Using this information, the System Manager can check the availability of the Chassis Managers and the current distribution of responsibilities between them (active vs. backup).

In active-active mode, instead of having a single RMCP network address that is switched between the two network interfaces (see description of active-standby mode in 3.4.2.1), the Chassis Manager supports RMCP on both interfaces with different IP addresses, as two separate IPMI channels (channels 1 and 2).

Each channel has its own set of LAN configuration parameters that includes the IP address, network mask, default gateway, etc. Both addresses are available via the “Get Chassis Manager IP Addresses” command.

The System Manager, in this case, is responsible for switching over to a different IP address if the currently used IP address becomes unavailable, or maintaining two parallel RMCP sessions to both addresses. If IP addresses on both interfaces become unavailable, the System Manager can access one of the IP addresses on the backup Chassis Manager and initiate a switchover in a non-standard way (for example, log in to the backup ChMM via `telnet` and issue the command `clia switchover`).

The active-active mode is enabled by setting the configuration parameter *USE_SECOND_CHANNEL* to *TRUE*. In this case, the configuration parameter *DEFAULT_RMCP_IP_ADDRESS2* should be set to a non-zero IP address. This IP address becomes the default IP address for the channel 2, and the configuration variables *DEFAULT_RMCP_NETMASK2* and *DEFAULT_GATEWAY_IP_ADDRESS2* specify the network mask and default gateway IP address for the channel 2, respectively.

The values of the above-mentioned configuration parameters have no effect if the corresponding values in the channel configuration parameter file are non-zero. The channel configuration parameters for the channel 2 are stored in the files */var/nvdata/ch2_param* and */var/nvdata/ch2_param_v2* on the ChMM.

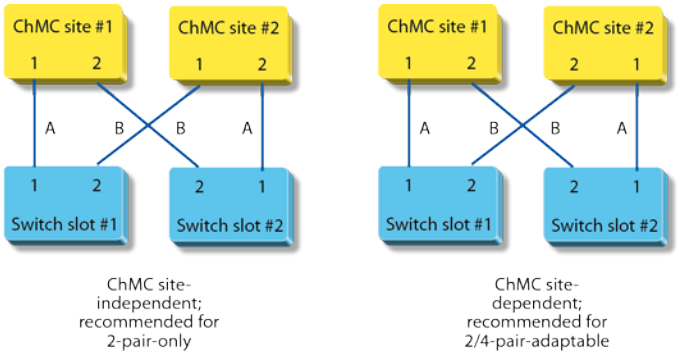
The Chassis Manager recognizes a second instance of the Chassis Manager IP Connection record in the Chassis FRU Information as the source of the IP address, netmask and default gateway for the second RMCP network interface. Therefore, it is possible to specify RMCP addressing information for both interfaces in the Chassis FRU Information.

In active-active mode, the IP addresses on both network interfaces are switched over to the backup ChMM as a result of a switchover.

3.4.2.3 Support for site-dependent ChMC cross-connects

According to the AdvancedTCA specification, ChMC cross-connects can be implemented with either a site independent or a site dependent configuration, as shown in Figure 3.

Figure 3: Implementation options for ChMC cross-connects



In the site-dependent variant, after a switchover in the default Chassis Manager configuration, each specific RMCP address is propagated to the corresponding network interface (*eth0* or *eth1*) on the other ChMC, and thus becomes associated with a different switch module/slot. In this case, from the perspective of the switch module, the ChMC switchover becomes non-transparent, which may be undesirable if, for example, each switch module hosts a System Manager. The configuration parameter *SWAPPED_CROSS_CONNECTS* can be used to avoid this non-transparency.

If the configuration parameter *SWAPPED_CROSS_CONNECTS* is defined and set to *TRUE* and if the hardware address of the ChMC is odd, the Chassis Manager swaps the values of the

configuration parameters *RMCP_NET_ADAPTER* and *RMCP_NET_ADAPTER2*. As a result, after a switchover, the RMCP addresses on the newly active Chassis Manager correspond to the same switch modules as before the switchover.

3.4.2.4 Bonded usage of the two network interfaces

The two physical network interfaces (*eth0* and *eth1*) can be used in a bonded fashion using the kernel bonding driver that joins several physical devices into one logical device and selects an active device for data transfers. In this mode, both physical network interfaces that can be used for outside connections are enslaved by the bonding driver at system startup and the Chassis Manager uses the single bonding interface for RMCP-based communication. This mode is useful when both devices are connected to the same network switch module and it does not matter which device is used at any given moment, as long as the Chassis Manager remains available via the network.

The bonding driver in the kernel is configured to use an active-backup policy, which means that only one slave device in the bond is active. A different slave device becomes active only if the active slave device fails. The bond's MAC address is externally visible on only one network adapter port to avoid problems with switch module hardware. Link status of the slave devices in the bond is polled every 100 milliseconds, which is the recommended value. Also, in the case of an active device switch, the bonding driver sends out a gratuitous ARP packet to notify switch hardware about the port change, since bonding slave devices share the same MAC address taken from the first device in the slave device list. When a network link loss is detected, active device switching is performed only after 500 milliseconds to avoid potential problems with short-term link loss. The idea is similar to the Active-Standby usage of two network interfaces except that device link state monitoring and switching is performed at low level by the kernel instead of the Chassis Manager.

The bonding mode is activated by changing the ipdevice variable value in U-Boot to *bond0*. This has to be done at the U-Boot level so that the system startup scripts perform the device initialization for *bond0* before the Chassis Manager starts. By default, this value is passed to the Chassis Manager as the default RMCP network adapter via the setting in */etc/shelfman.conf*:

```
RMCP_NET_ADAPTER = $IPDEVICE
```

It is possible to use an alias interface of *bond0* if it is necessary to have a permanent IP address on the ChMM while the RMCP address floats between active and standby ChMMs i.e.

```
RMCP_NET_ADAPTER = "bond0:1"
```

The parameter *RMCP_NET_ADAPTER2* must not be set while using the bonding interface as an RMCP adapter. The bonding mode assumes single RMCP network adapter usage from the Chassis Manager perspective, so the parameter *USE_SECOND_CHANNEL* must be set to *FALSE*.

3.4.3 Gigabit Ethernet Support

Starting from release 1.3.0, the Chassis Manager supports Gigabit Ethernet connections using one or two Microchip KSZ9563RNX devices that can be implemented on the carrier board. By default when a Microchip KSZ9563R switch is connected to the ChMM-700R Ethernet 0 PHY interface it is assumed to be connected to ChMM-700R I2C Bus I2C4 (logical I2C Bus 0, refer to Table 13) at 7-bit I2C address 5Fh. Likewise when a

Microchip KSZ9563R switch is connected to the ChMM-700R Ethernet 1 PHY interface it is assumed to be connected to ChMM-700R I2C Bus I2C3 (logical I2C Bus 5) at 7-bit I2C address 5Fh. If needed the I2C bus and address values can be changed through four dedicated U-Boot environment variables *eth0_switch_bus, eth0_switch_address, eth1_switch_bus, eth1_switch_address*, refer to Table 3. Note if these U-Boot environment variables are not defined the default values are used, thus these U-Boot variables only need to be defined if the default value needs to be changed.

WARNING: Upon boot-up the U-Boot and kernel image will attempt to access the GbE Switch (Microchip KSZ9563RNX) at I2C address 5Fh on physical I2C buses I2C4 and I2C3. Customers that have ChMM-700R Carrier board designs that implement an I2C device with a 7-bit address of 5Fh on physical I2C bus I2C4 or I2C3, that is not the Microchip KSZ9563RNX, must contact the Pigeon Point Support (support.pigeonpoint@nVent.com).

3.4.4 Using the alternate software redundancy interface

If the mezzanine is configured for ChMC Cross-connect operation, two additional network interfaces are implemented over the two USB connections. In this configuration, they always connect the two redundant Chassis Managers. These interfaces are named usb0 and usb1. The interface usb0 always exists, while the interface usb1 exists only if the interface usb0 is active on the peer Chassis Manager (which means that the peer Chassis Manager is physically installed and running). Also, the interfaces are cross-connected: usb0 on the first Chassis Manager is connected to usb1 on the second Chassis Manager, and vice versa.

The Chassis Manager supports usage of the USB network interfaces for communication between the redundant Chassis Managers. To use this feature, it is necessary to define two redundancy network adapters in the Chassis Manager configuration file */etc/shelfman.conf*, as follows:

```
REDUNDANCY_NET_ADAPTER = usb0
REDUNDANCY_NET_ADAPTER2 = usb1
```

One additional consideration relates to the definition of the subnet mask for the redundancy network interfaces. In the legacy case, when only one redundant network adapter is used, two different IP addresses are derived from the redundancy IP address specified in */etc/shelfman.conf*. They are assigned to the two endpoints of the redundancy connection and differ only in the least significant bit.

However, when two redundancy network adapters are used, four different IP addresses are used, one for each of the endpoints (two endpoints on each of the two redundant Chassis Managers). To ensure proper operation, the two endpoints on the same Chassis Manager (usb0 and usb1) must belong to different logical networks, while usb0 on one Chassis Manager and usb1 on the other Chassis Manager must belong to the same logical network. This is achieved by dividing the IP address space into two ranges.

These ranges (logical networks) are defined by the subnet mask given by the parameter *REDUNDANCY_NETMASK* from the configuration file */etc/shelfman.conf*. If the network

mask is 255.255.255.128 then the first range is 192.168.1.0 - 192.168.1.127 and the other is 192.168.1.128 – 192.168.1.255.

The usb0 endpoint on the first Chassis Manager and the usb0 endpoint on the other Chassis Manager are in the first range. The usb0 endpoint on the first Chassis Manager and the usb0 endpoint on the other Chassis Manager are in the second range.

The 4 IP addresses in question can be derived from one IP address (for example, the IP address assigned to usb0 on the Chassis Manager with the even hardware address) and the network mask (for which the recommended value is 255.255.255.128). The rules are as follows:

- To compute the IP address for usb0 on the Chassis Manager with the even hardware address when the file `/etc/readhwaddr` is present you should set the least significant bit of *REDUNDANT_IP_ADDRESS* to 0.
- To compute the IP address for usb1 on the other Chassis Manager you should toggle the least significant bit in the usb0 IP address on the first Chassis Manager. This guarantees that usb0 on Chassis Manager with the even hardware address and usb1 on the Chassis Manager with the odd hardware address are in the same logical network and are not equal to each other.
- To compute the IP address for usb1 on the Chassis Manager with even hardware address you should take the IP address for usb0 on the same Chassis Manager and toggle the least non-zero bit of the network mask. This guarantees that the IP addresses for usb0 and usb1 on the Chassis Manager with the even hardware address are in different logical networks.
- The last step is to compute the IP address for usb0 on the Chassis Manager with the odd hardware address. You should either toggle the least significant bit in the IP address for usb1 on the Chassis Manager with the even hardware address or toggle the least non-zero bit of the network mask in the IP address for usb0 on the Chassis Manager with odd hardware address. The result is the same. This guarantees that usb0 on Chassis Manager with the odd hardware address and usb1 on Chassis Manager with the even hardware address are in the same logical network and are not equal.

Here is an example of deriving IP addresses for the USB network interfaces, under the assumption that the following definitions are in `/etc/shelfman.conf`:

```
REDUNDANT_IP_ADDRESS = 192.168.1.2
REDUNDANCY_NETMASK = 255.255.255.128
```

The least significant non-zero bit in the network mask is the 7th bit (where smaller bit numbers are less significant). To toggle this bit in an IP address it is sufficient to add 128 (if this bit is set to zero in the IP address) or subtract 128 (if this bit is set to 1 in the IP address).

To toggle the least significant bit in an IP address it is sufficient to add 1 if the IP address is even or subtract 1 if the IP address is odd. Since *REDUNDANT_IP_ADDRESS* is even, the computations are the same whether the file `/etc/readhwaddr` is present or not.

On the ChMM with the even hardware address the assignment of IP addresses looks like this:

- usb0: 192.168.1.2 (no changes)

- usb1: 192.168.1.130 (toggling the least significant non-zero bit of the netmask)

On the ChMM with the odd hardware address the assignment of IP addresses looks like this:

- usb0: 192.168.1.131 (toggling the least significant bit of the IP address and the least non-zero bit of the netmask)
- usb1: 192.168.1.3 (toggling the least significant bit of the IP address)

Here is another example of deriving IP addresses for the USB network interfaces, under the assumption that the following definitions are in *`/etc/shelfman.conf`*:

```
REDUNDANT_IP_ADDRESS = 192.168.1.13
REDUNDANCY_NETMASK = 255.255.255.128
```

Suppose also that the file *`/etc/readhwaddr`* is present.

The least significant non-zero bit in the network mask is 7th bit. To toggle this bit in an IP address it is sufficient to add 128 (if this bit is set to zero in the IP address) or subtract 128 (if this bit is set to 1 in the IP address). To toggle the least significant bit in an IP address it is sufficient to add 1 if the IP address is even or subtract 1 if the IP address is odd.

On the ChMM with the even hardware address the assignment of IP addresses looks like this:

- usb0: 192.168.1.12 (since the file *`/etc/readhwaddr`* is present, the least significant bit should be set to zero)
- usb1: 192.168.1.140 (toggling the least significant non-zero bit of the netmask)

On the ChMM with the odd hardware address the assignment of IP addresses looks like this:

- usb0: 192.168.1.141 (toggling the least significant bit of the IP address and the least significant non-zero bit of the netmask)
- usb1: 192.168.1.13 (toggling the least significant bit of the IP address)

Data transferred over the Software Redundancy Interface can be compressed. A configuration variable *`REDUNDANCY_COMPRESSION_THRESHOLD`* specifies the threshold for message size; messages that are bigger than this threshold are compressed by the *`gzip`* algorithm before sending and uncompressed after receiving. Smaller messages are sent in uncompressed form. The default value of the threshold is *`-1`*, which effectively turns compression off.

3.4.5 Changing the default ChMM network parameters

Configuring a ChMM to work in a specific network environment requires changing the following network parameters:

- RMCP IP address (*`rmcpaddress`*)
- RMCP gateway address (*`gatewayip`*)
- RMCP netmask (*`netmask`*)

If the second ChMM Ethernet is used for software redundancy communication between a pair of ChMMs, the network parameters above would typically not need to be adapted to a different

network environment, since this is a dedicated private network between the redundant ChMMs. However, if the second network interface is used for RMCP communication, please use steps 10 and 11 below to change the settings; U-Boot environment variables do not apply to the second network interface.

Changing the RMCP network parameters is a two step process. First, the U-Boot network environment variables need to be updated, then the booted active ChMM module network settings need to be updated using the Chassis Manager command line interface (CLIA). Specific steps are shown below:

1. Attach a serial port console connection to the ChMM module.
   This typically is 115200 Baud, N/8/1.
2. Reset the ChMM carrier and press the space bar to interrupt the automatic boot-up process. You should see:

```
PowerPrep start initialize power...
Battery Voltage = 1.44V
No battery or bad battery detected!!!.Disabling battery voltage measurements.
Feb 24 201207:45:29
FRAC 0x92926152
memory type is DDR2
Wait for ddr ready 1
power 0x00820616
Frac 0x92926152
start change cpu freq
hbus 0x00000003
cpu 0x00010001
start test memory access
ddr2 0x40000000
finish simple test
U-Boot 2009.08 (Oct 24 2017 - 11:44:23)

Freescale i.MX28 family
CPU:   297 MHz
BUS:   99 MHz
EMI:   130 MHz
GPMI:  24 MHz
I2C:   ready
DRAM:  128 MB
SFGEN: S25FL512S detected, total size 64 MB
A2F:   SPICOMM protocol v1.7, M3 firmware v1.8, FPGA design v1.1.0.0
A2F:   A2F firmware, version 1.8
```

```
A2F:   Device type: A2F060M3E-FG256
A2F:   MSS clock frequency: 40 MHz
A2F:   Fabric clock frequency: 20 MHz
A2F:   Fast delay calibration: 1330 cycles per 100uS
A2F:   eNVM: 128 KB (00000000 - 00020000)
A2F:   eSRAM: 16 KB (20000000 - 20004000)
A2F:   Extram start: 20002750
A2F:   Last reset cause: HARD
A2F:   VMON flags: 00000000
RUPG:  booting from image 0 (confirmed)
In:    serial
Out:   serial
Err:   serial
Net:   FEC0: 00:18:49:02:b2:06, FEC1: 00:18:49:02:b2:07
FEC0, FEC1
Hit any key to stop autoboot:  0
ChMM700
```

3. Echo current network settings:

```
ChMM700 printenv rmcpaddr netmask gatewayip
rmcpaddr=192.168.0.44
netmask=255.255.255.0
gatewayip=192.168.0.1
ChMM700
```

4. Change settings and commit to non-volatile storage:

```
ChMM700 setenv rmcpaddr 10.1.1.10
ChMM700 setenv netmask 255.255.0.0
ChMM700 setenv gatewayip 10.1.1.1
ChMM700 saveenv
Saving Environment to SPI Flash...
Erasing SPI flash...Writing to SPI flash...done
ChMM700
```

5. Boot the ChMM up to full operational state and log in as user “root”.

```
ChMM700 reset
resetting ...
U-Boot 2009.08 (Oct 24 2017 - 11:44:23)

Freescale i.MX28 family
CPU:   297 MHz
```

```
BUS:   99 MHz
EMI:   130 MHz
GPMI:  24 MHz
I2C:   ready
DRAM:  128 MB
SFGEN: S25FL512S detected, total size 64 MB
A2F:   SPICOMM protocol v1.7, M3 firmware v1.8, FPGA design v1.1.0.0
A2F:   A2F firmware, version 1.8
A2F:   Device type: A2F060M3E-FG256
A2F:   MSS clock frequency: 40 MHz
A2F:   Fabric clock frequency: 20 MHz
A2F:   Fast delay calibration: 1330 cycles per 100uS
A2F:   eNVM: 128 KB (00000000 - 00020000)
A2F:   eSRAM: 16 KB (20000000 - 20004000)
A2F:   Extram start: 20002750
A2F:   Last reset cause: SOFT
A2F:   VMON flags: 00000000
RUPG:  booting from image 1 (confirmed)
In:    serial
Out:   serial
Err:   serial
Net:   FEC0: 00:18:49:02:b2:06, FEC1: 00:18:49:02:b2:07
FEC0, FEC1
Hit any key to stop autoboot:  0
RUPG:  upgrade file is not present, skipping
Loading file 'uImage.1' to addr 0x42000000 with size 1541536 (0x001785a0)...
Loaded from UBIFS cache
Loading file 'rfs.1' to addr 0x46000000 with size 3092716 (0x002f30ec)...
Loaded from UBIFS cache
## Booting kernel from Legacy Image at 42000000 ...
   Image Name:   Linux-2.6.34.8-chmm700
   Created:      2017-10-24  16:44:21 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    1541472 Bytes =  1.5 MB
   Load Address: 40008000
   Entry Point:  40008000
   Verifying Checksum ... OK
## Loading init Ramdisk from Legacy Image at 46000000 ...
   Image Name:   ChMM700 RFS 1.2.0
   Created:      2017-10-24  16:39:38 UTC
```

```
   Image Type:   ARM Linux RAMDisk Image (gzip compressed)
   Data Size:    3092652 Bytes =  2.9 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...
…
ChMM700 login: root
```

6. Allow the ChMM to start up.
   Please note that the settings that were changed in the U-Boot firmware are not necessarily propagated to the Linux environment. The reason for this is that the Chassis Manager needs to maintain its own copy of the network configuration data in order to manage failover situations. If this is the first time the Chassis Manager has been booted, or if the Flash filesystems have been reset prior to bootup, then the Chassis Manager uses the network settings provided by U-Boot to set up this networking context (and thus the changes you made in U-Boot are propagated forward). Otherwise, the network configuration data is stored on the ChMM Flash file system in the following files:

   - *`/var/nvdata/ch1_param`* - parameters for channel 1
   - *`/var/nvdata/ch1_param_v2`* - additional parameters for channel 1
   - *`/var/nvdata/ch1_param_ipv6_v2`* - additional IPv6-related parameters for channel 1 (starting with Chassis Manager release 1.1.0)
   - *`/var/nvdata/ch2_param`* - parameters for channel 2
   - *`/var/nvdata/ch2_param_v2`* - additional parameters for channel 2
   - *`/var/nvdata/ch2_param_ipv6_v2`* - additional IPv6-related parameters for channel 2 (starting with Chassis Manager release 1.1.0)

   In this latter case, the following steps are required to configure the network settings in the Chassis Manager context.

   First, check to see if you are interacting with the active Chassis Manager.
   You only need to make changes on the active Chassis Manager as it updates the backup with the network configuration changes via the redundancy interface. Use the *`hwri`* command, and look for “Active” in the output of the command (see the underlined fragment of the output below). The *`hwri`* command reports the current state of the hardware redundancy signals on

the ChMM, exposed through SmartFusion customizable System-on-Chip (cSoC). The flags reported by the *hwri* command are identical for all three variants, though the actual register interface and bit layout is different among those variants. If you are not using the active ChMM, connect to the other ChMM device and repeat step 7:

```
# hwri
HWRI word: 3307
        1000h - Local Presence
        0100h - Remote Presence
        0002h - Local Healthy
        0200h - Remote Healthy
        0004h - Local Switchover Request
        2000h - Active
```

7. Get the current IP settings (for channel 1):

```
# clia getlanconfig 1
Pigeon Point Chassis Manager Command Line Interpreter

PING 192.168.1.253 (192.168.1.253): 56 data bytes
64 bytes from 192.168.1.253: seq=0 ttl=64 time=1.188 ms

--- 192.168.1.253 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.188/1.188/1.188 ms
Authentication Type Support: 0x17 ( None MD2 MD5 Straight Password/Key )
Authentication Type Enables:
    Callback level: 0x00
    User level: 0x17 ( "None" "MD2" "MD5" "Straight Password/Key" )
    Operator level: 0x17 ( "None" "MD2" "MD5" "Straight Password/Key" )
    Administrator level: 0x17 ( "None" "MD2" "MD5" "Straight Password/Key" )
    OEM level: 0x00
IP Address: 192.168.1.117
IP Address Source: Static Address (Manually Configured) (0x01)
MAC Address: 00:18:49:02:b2:06
Subnet Mask: 255.255.255.0
IPv4 Header Parameters: 0x40:0x40:0x10
Primary RMCP Port Number: 0x026f
Secondary RMCP Port Number: 0x0298
BMC-generated ARP Control: 0x02
    Enable BMC-generated ARP Response
Gratuitous ARP Interval: 2.0 seconds
```

```
Default Gateway Address: 192.168.1.253
Default Gateway MAC Address: 00:01:02:1e:13:04
Backup Gateway Address: 0.0.0.0
Backup Gateway MAC Address: 00:00:00:00:00:00
Community String: "public"
Number of Destinations: 16
Destination Type:
    DST Type # 1, Type: Acknowledged PET Trap Destination (0x80), ACK Timeout
/ Retry Interval: 3 seconds, Retries: 2
Destination Address:
    DST Addresses # 1, Address Format: IPv4 IP Address followed by DIX
ethernet / 802.3 MAC Address (0x00)
        Gateway: Default (0x00), Alerting IP: 192.168.1.253, Alerting MAC:
00:01:02:1e:13:04
802.1q VLAN ID: 0 (disabled)
VLAN priority: 0
Cipher Suite Entry count: 15
Supported Cipher Suite IDs: 0h, 1h, 2h, 3h, 4h, 5h, 6h, 7h, 8h, 9h, Ah, Bh,
Ch, Dh, Eh
Cipher Suite Privilege Levels:
    ID 00h, Priv.Level 'User'            (2); ID 01h, Priv.Level 'User'
(2);
    ID 02h, Priv.Level 'Administrator'   (4); ID 03h, Priv.Level 'OEM
Proprietary' (5);
    ID 04h, Priv.Level 'OEM Proprietary' (5); ID 05h, Priv.Level 'OEM
Proprietary' (5);
    ID 06h, Priv.Level 'User'            (2); ID 07h, Priv.Level
'Administrator'   (4);
    ID 08h, Priv.Level 'OEM Proprietary' (5); ID 09h, Priv.Level 'OEM
Proprietary' (5);
    ID 0Ah, Priv.Level 'OEM Proprietary' (5); ID 0Bh, Priv.Level
'Administrator'   (4);
    ID 0Ch, Priv.Level 'OEM Proprietary' (5); ID 0Dh, Priv.Level 'OEM
Proprietary' (5);
    ID 0Eh, Priv.Level 'OEM Proprietary' (5);
Destination Address VLAN TAGs:
    N/A
Bad password threshold: disabled
IPv6/IPv4 support: IPv6+IPv4, IPv6 for LAN Alerts (0x06)
IPv6/IPv4 Addressing Enables: 0x02 (enable IPv6 and IPv4)
IPv6 Header Static Traffic Class: 0x00
IPv6 Header Static Hop Limit: 64
IPv6 Header Flow Label: 0x00,0x00,0x00
```

```
IPv6 Status:
    Max Static Addresses: 4, Max Dynamic Addresses: 4; Addressing Supported:
0x02 (SLAAC)
IPv6 Static Addresses:
    N/A
IPv6 DHCPv6 Static DUID Storage Length: 3*16 bytes
IPv6 DHCPv6 Static DUIDs:
   0: 27 00 02 00 00 31 5A 48 50 4D 2E 33 2D 31 00 00
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      00 00 00 00 03 00 00 00
   1: 27 00 02 00 00 31 5A 48 50 4D 2E 33 2D 31 00 00
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      00 00 00 00 03 00 00 00
   2: 27 00 02 00 00 31 5A 48 50 4D 2E 33 2D 31 00 00
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      00 00 00 00 03 00 00 00
   3: 27 00 02 00 00 31 5A 48 50 4D 2E 33 2D 31 00 00
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      00 00 00 00 03 00 00 00
IPv6 Dynamic Addresses:
    0: fe80::218:49ff:fe02:b206/64: SLAAC: Active (0x00)
IPv6 DHCPv6 Dynamic DUID Storage Length: 3*16 bytes
IPv6 DHCPv6 Dynamic DUIDs:
   0: 27 00 02 00 00 31 5A 48 50 4D 2E 33 2D 31 00 00
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      00 00 00 00 03 01 00 00
   1: 27 00 02 00 00 31 5A 48 50 4D 2E 33 2D 31 00 00
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      00 00 00 00 03 01 00 00
   2: 27 00 02 00 00 31 5A 48 50 4D 2E 33 2D 31 00 00
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      00 00 00 00 03 01 00 00
   3: 27 00 02 00 00 31 5A 48 50 4D 2E 33 2D 31 00 00
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      00 00 00 00 03 01 00 00
IPv6 DHCPv6 Timing Configuration Support: Disabled (0)
IPv6 Router Address Configuration Support: Enable Static And Dynamic (3)
IPv6 Static Router 1
    IP Address: N/A
    MAC Address: N/A
```

```
    Prefix: ::/0
IPv6 Static Router 2
    IP Address: N/A
    MAC Address: N/A
    Prefix: ::/0
IPv6 Dynamic Router Info Sets: 0
IPv6 Dynamic Router Received Hop Limit: N/A
IPv6 ND/SLAAC Timing Configuration Support: Disabled (0)
```

8. Change the IP settings (for channel 1):

```
# clia setlanconfig 1 ip 10.1.1.10
Pigeon Point Chassis Manager Command Line Interpreter
IP set successfully
# clia setlanconfig 1 subnet_mask 255.255.0.0
Pigeon Point Chassis Manager Command Line Interpreter
Subnet Mask set successfully
# clia setlanconfig 1 dft_gw_ip 10.1.1.1
Pigeon Point Chassis Manager Command Line Interpreter
Default Gateway Address set successfully
```

If the second network interface is used for RMCP, get the current IP settings for channel 2:

```
# clia getlanconfig 2
Pigeon Point Chassis Manager Command Line Interpreter
...
```

Change the IP settings (for channel 2):

```
# clia setlanconfig 2 ip 10.2.1.10
Pigeon Point Chassis Manager Command Line Interpreter
IP set successfully
# clia setlanconfig 2 subnet_mask 255.255.0.0
Pigeon Point Chassis Manager Command Line Interpreter
Subnet Mask set successfully
# clia setlanconfig 2 dft_gw_ip 10.2.1.1
Pigeon Point Chassis Manager Command Line Interpreter
Default Gateway Address set successfully
```

3.4.6 Assigning VLAN IDs

The Chassis Manager supports Virtual LANs (VLANs), in accordance with version 2.0 of the IPMI specification. That is, RMCP/RMCP+ traffic to and from the Chassis Manager can be passed over a specific virtual LAN; the ID of that virtual LAN is stored in the LAN configuration parameters and is automatically added to the outgoing RMCP/RMCP+ packets (and stripped from the incoming packets) in accordance with the IEEE 802.1q specification. Separate VLAN IDs are supported for the two LAN channels (1 and 2) that can carry RMCP/RMCP+ traffic.

To assign a VLAN ID to a channel, use the IPMI "Set LAN Configuration Parameters" command, the CLI *setlanconfig* command or the CLI analog in the SNMP or WEB interface. Changing the VLAN ID may disrupt current RMCP/RMCP+ communication on the affected channel, so it should be used with caution. The VLAN ID is a 12-bit unsigned number, the value 0 indicates that virtual LANs are not used on the channel. Once set, the VLAN ID is stored persistently in the LAN configuration parameters on the ChMM Flash file system.

Example: Get the current VLAN ID settings for channel 1 (where no VLANs are currently in use):

```
# clia getlanconfig 1 vlan_id
Pigeon Point Chassis Manager Command Line Interpreter
802.1q VLAN ID: 0 (disabled)
```

Set the VLAN ID for channel 1 to a different value:

```
# clia setlanconfig 1 vlan_id enabled 5
Pigeon Point Chassis Manager Command Line Interpreter
VLAN ID set successfully
```

The IPMI 2.0 specification allows distinct VLAN IDs to be used for the outgoing SNMP traps (LAN alerts) generated by the Chassis Manager as a result of PEF processing. As allowed by the IPMI specification, the Chassis Manager does not support this extension and always uses the main VLAN ID that is assigned to the corresponding channel for such LAN alerts. Each LAN configuration parameter that defines a VLAN ID for a LAN alert destination is read-only in the Chassis Manager and identical to the VLAN ID for the corresponding LAN channel.

The default VLAN ID for a channel is usually 0 (VLANs disabled), but can be configured via the configuration parameters *DEFAULT_VLAN_ID* (for channel 1) and *DEFAULT_VLAN_ID2* (for channel 2). These defaults apply in the absence of the LAN configuration parameters on a ChMM, on a fresh ChMM. Once the LAN configuration parameters are defined and stored, the Set LAN Configuration Parameters mechanism should be used to change the VLAN ID.

3.4.7 Assigning IP addresses to the Chassis Manager via DHCP (version 4)

DHCP (Dynamic Host Configuration Protocol) and DHCP servers can be used to assign IP addresses to the Chassis Manager. This section focuses on using version 4 of this protocol (as opposed to DHCPv6) that deals with IPv4 addresses (as opposed to IPv6 addresses). The abbreviations DHCP and DHCPv4 are used in this section to designate DHCP version 4, and the abbreviations IP, IPv4 are used in this section to designate IP version 4.

The following types of IP addresses can be assigned via DHCP:

- RMCP accessible addresses (for one or both network interfaces)
- Private Chassis Manager addresses (for one or both network interfaces, for both Chassis Managers).

A total of six IP Chassis Manager-related addresses can be assigned via DHCP.

Two different DHCPv4 clients are currently available in the Chassis Manager. The new DHCPv4 client is available starting in release 1.2.0 and has the following differences from the legacy DHCPv4 client:

- The new client supports finite lease times, and performs renewal and rebinding of IP addresses on the DHCP server as defined by the DHCP protocol (unlike the legacy client, where the lease time is assumed to be always infinite).
- The new client operates also on the backup Chassis Manager; it is responsible for obtaining the IP addresses for the backup physical Chassis Manager instance (unlike the legacy client, where the active Chassis Manager operates as a DHCP proxy client for the backup Chassis Manager). This is to ensure that a request comes to the DHCP server from the same network interface on the client for which the address is requested (which is a DHCPv4 protocol requirement).
- The new client assigns an IP address to each specific interface as soon as it receives the address for that interface from the DHCP server (unlike the legacy client, where the active Chassis Manager collects all six IP addresses from the DHCP server before assigning any of them). The old behavior is not applicable in the new architecture because the DHCP client on the backup Chassis Manager operates independently of the DHCP client on the active Chassis Manager.
- The state of the client is passed over the redundancy interface; therefore, after a switchover, the newly active Chassis Manager knows that state, including the remaining timeout periods for all leases associated with the logical Chassis Manager. The state of the DHCP client for the logical Chassis Manager is therefore transparent for the DHCP server; the client does not restart the DHCP protocol state machine after the switchover. The IP address for the logical Chassis Manager moves over to the other Chassis Manager during a switchover, as in the static case.
- The IP address for the physical Chassis Manager that becomes active stays unchanged during the switchover. The physical Chassis Manager that lost the active role reboots and, after its activation as the backup, asks the DHCP server again for an address. It depends on the server whether it provides the same or a different IP address; the Client Identifier that the Chassis Manager sends to the DHCP server is the same, however.

The choice of which client to use for DHCPv4 client functionality (legacy vs. new) depends on the Chassis Manager configuration parameter *ENABLE_NEW_DHCPV4_CLIENT*; the default value is for this parameter is *TRUE*.

In some future release, multiple releases after release 1.2.0, support for the legacy DHCP client is likely to be removed from the Chassis Manager, together with the differentiating configuration parameter. Thereafter, only the new DHCP client will be included with the Chassis Manager in subsequent releases.

A specific IP address is designated by a particular value of the Client Identifier that is passed from the Chassis Manager to the DHCP server. All formats of Client Identifiers described below are supported by both legacy and new clients.

If the HPM.3 Client Identifier format is enabled (which is the default), the Client Identifier has the following standard format, defined in the HPM.3 specification:

Table 8: HPM.3 DHCP Client ID

Field	Length	Format	Content
Header	1	Binary	61 (3Dh)
Data Length	1	Binary	Length of remaining content (44, 2Ch)
Type	1	Binary	255 (FFh)
IAID	4	Binary	*IPMI LAN Channel Index* with range 0 through Ah and all other values reserved; see section 2.5.1 of HPM.3, DHCP-Assigned Platform Management Parameters Specification.
DUID	39	Binary	*DHCP Unique Identifier*. See Table 9: HPM.3 DHCP Unique Identifier format

The DHCP Unique Identifier (DUID) has the following format:

Table 9: HPM.3 DHCP Unique Identifier format

Field	Length	Format	Content
DUID format	2	Binary	DUID-EN identifier (2)
DUID header part 1	4	Binary	PICMG IANA ID (12634, 00315Ah)
DUID header part	5	ASCII	Text “HPM.3” identifier

2			(48504D2E33h)
DUID header part 3	2	ASCII	HPM.3 DUID version ("-1", 2D31h)
Chassis address type/length	1	Binary	*Chassis Address Type/Length Byte* field from the Chassis FRU Information (VITA 46.11 Address Table record)
Chassis address	20	Binary	*Chassis Address* field from the Chassis FRU Information (VITA46.11 Address Table record)
Chassis type	1	Binary	0 for VITA 46.11 chassis,
Slot location	4	Binary	See Table 10: Slot Location record for

Table 10: Slot Location record for VITA 46.11

OFFSET	LENGTH	DEFINITION
0	1	*Primary Site Type*. The Site Type of the Slot, for a Front Board or other Intelligent FRU that attaches directly to System IPMB.
1	1	*Primary Site Number*. The Site Number of the Slot, for a Front Board or other Intelligent FRU that attaches directly to System IPMB.
2	1	*Secondary Site Type*. Applicable when the Slot is a subsidiary site on the FRU designated by the *Primary Site Type* and *Primary Site Number*; identifies the Site Type of that Slot.
3	1	*Secondary Site Number*. Applicable when the Slot is a subsidiary site on the FRU designated by the *Primary Site Type* and *Primary Site Number*; identifies the Site Number of that Slot.

For the Chassis Manager addresses, the fields are assigned as follows:

- Primary Site Type = 3 (Dedicated Chassis Manager)

- Primary Site Number = 0 to request RMCP accessible addresses, 1 or 2 to request Private Chassis Manager addresses (the value 1 or 2 is the site number for the corresponding Chassis Manager in the Chassis Address Table)
- Secondary Site Type and Secondary Site Number = 0
- IPMI LAN Channel Index = 0 for the physical interface used by the first LAN channel and =1 for the physical interface used by the second LAN channel. The first physical interface is normally eth0 and the second physical interface is normally eth1, but this can be changed by the Chassis Manager configuration variables *RMCP_NET_ADAPTER* and *RMCP_NET_ADAPTER2*.

If the HPM.3 Client Identifier format is disabled, then in the default implementation, the Client Identifier is based on the Chassis Address string that is stored in the Chassis FRU Information, plus the Request Identifier (Request ID) byte at the end.

The Request ID byte has the following format:

- Request ID bits 7..4 – Chassis Manager number (0, 1, 2, with 0 for logical Chassis Manager)
- Request ID bits 3..0 – Interface number (0 for eth0, 1 for eth1)

If the Chassis Address string in the Chassis FRU Information is empty, a hard-coded string of 7 zero bytes is used. In that case, the Client Identifier values are implemented as follows:

Currently the Client Identifier values are hardcoded as follows:

```
00:00:00:00:00:00:00:10 – Chassis Manager 1, eth0
00:00:00:00:00:00:00:11 - Chassis Manager 1, eth1
00:00:00:00:00:00:00:20 - Chassis Manager 2, eth0
00:00:00:00:00:00:00:21 - Chassis Manager 2, eth1
00:00:00:00:00:00:00:00 - Logical Chassis Manager eth0
00:00:00:00:00:00:00:01 - Logical Chassis Manager eth1
```

However, these Client IDs can be redefined for a specific carrier inside the corresponding ChMM carrier-specific module (if the HPM.3 Client Identifier format is disabled).

To use this feature in the Chassis Manager, it is necessary to define the configuration parameter *USE_DHCP* in the Chassis Manager configuration file *`/etc/shelfman.conf`*, as follows:

```
USE_DHCP = TRUE
```

The configuration parameter *ENABLE_NEW_DHCPV4_CLIENT* determines whether the legacy or the new DHCP client is used. By default, this variable is set to *TRUE* which means the new client is used. To use the legacy DHCP client, set this variable explicitly to *FALSE* in the Chassis Manager configuration file, as follows:

```
ENABLE_NEW_DHCPV4_CLIENT = FALSE
```

By default, the Chassis Manager uses the first DHCP server that answers the *DHCPDISCOVER* request to assign the IP addresses. If more than one DHCP server is present in the network, the configuration parameter *DHCP_SERVER_ADDRESS* can be used in the Chassis Manager

configuration file *`/etc/shelfman.conf`* to specify the IP address of the DHCP server to be used, as follows:

```
DHCP_SERVER_ADDRESS = 192.168.1.50
```

In that case, only the specified DHCP server is used.

In addition, if the configuration parameter *DHCP_FOR_RMCP_ONLY* is set to *TRUE*, only the RMCP-accessible (Logical Chassis Manager) IP addresses are assigned via DHCP. In this case, the private ChMM IP addresses, if any, are left untouched.

The DHCP server should be configured to provide a unique IP address for each Client Identifier.Also, when the legacy client is in use, the lease time of each address must be set as "infinite" (time value 0xFFFFFFFF) to avoid IP address expiration. The new client correctly handles finite lease times.

If the configuration of the DHCP server changes over time, it may be necessary to restart the procedure of IP address assignment on the DHCP client for the Chassis Manager. The CLI command *`dhcp restart`* can be used for this purpose. Another CLI command, *`dhcp status`*, can be used to find out the current DHCP client status on the Chassis Manager (which IP addresses have already been retrieved and assigned). Please refer to the C*hassis Manager External Interface Reference* for the detailed information about these CLI commands.

In addition to providing an IP address related information, a DHCP server can be configured, via the TFTP Server Name and Bootfile Name options, to provide the name of a TFTP server and identify a bootfile on that server. For the benefit of other applications that may run on the ChMM in parallel with the Chassis Manager, the DHCP client, if it receives this additional information, stores it in the ChMM file system. A separate file is used for each of the 6 network interfaces processed by the DHCP client, according to the following rules:

`/tmp/tftp_info_0`	for the first network interface on the ChMM with the lower Hardware Address
`/tmp/tftp_info_1`	for the second network interface on the ChMM with the lower Hardware Address
`/tmp/tftp_info_2`	for the first network interface on the ChMM with the higher Hardware Address
`/tmp/tftp_info_3`	for the second network interface on the ChMM with the higher Hardware Address
/`tmp/tftp_info_4`	for the first RMCP-accessible interface
`/tmp/tftp_info_5`	for the second RMCP-accessible interface

The first four files above are stored on the corresponding ChMM only, the last two are stored on both redundant ChMMs.

Each file consists of two lines. The first line represents the TFTP server name and the second line represents the boot file name; for example:

```
# cat /tmp/tftp_info_2
```

```
TFTPSERVER="192.168.1.253"
BOOTFILE="/tftpboot/ssh_script_eth0_ShM2"
```

The example configuration file below shows how to configure the Linux DHCP server (DHCPD) to provide IP addresses, as well as TFTP server and bootfile information to the Chassis Manager. Fixed predefined addresses are used for that purpose.

This file should be located as *`/etc/dhcpd.conf`* on the system hosting the DHCP server. Other DHCP servers (such as those on non-Linux operating systems) are configured differently.

```
allow booting;
allow bootp;

option domain-name "tst";
option subnet-mask 255.255.255.0;
option domain-name-servers 192.168.1.100;
option ntp-servers 192.168.1.50;
option routers 192.168.1.253;
option vendor-class-identifier "nVent";

min-lease-time 4294967295;
default-lease-time 4294967295;

use-host-decl-names on;
ddns-update-style ad-hoc;

subnet 192.168.1.0 netmask 255.255.255.0
{
    host client00 {
        option dhcp-client-identifier 0:0:0:0:0:0:0:10;
        fixed-address 192.168.1.140;
        option tftp-server-name "192.168.1.253";
        option bootfile-name "/tftpboot/ssh_script_eth0_ShM1";
    }
    host client01 {
        option dhcp-client-identifier 0:0:0:0:0:0:0:11;
        fixed-address 192.168.1.141;
        option tftp-server-name "192.168.1.253";
        option bootfile-name "/tftpboot/ssh_script_eth1_ShM1";
    }
    host client02 {
        option dhcp-client-identifier 0:0:0:0:0:0:0:20;
        fixed-address 192.168.1.142;
```

```
        option tftp-server-name "192.168.1.253";
        option bootfile-name "/tftpboot/ssh_script_eth0_ShM2";
    }
    host client03 {
        option dhcp-client-identifier 0:0:0:0:0:0:0:21;
        fixed-address 192.168.1.143;
        option tftp-server-name "192.168.1.253";
        option bootfile-name "/tftpboot/ssh_script_eth1_ShM2";
    }
    host client04 {
        option dhcp-client-identifier 0:0:0:0:0:0:0:0;
        fixed-address 192.168.1.144;
    }
    host client05 {
        option dhcp-client-identifier 0:0:0:0:0:0:0:1;
        fixed-address 192.168.1.145;
    }
}
```

3.4.8 Using IPv6 for chassis-external access to the Chassis Manager

Starting with release 1.1.0, the Chassis Manager supports IPv6 in parallel with IPv4 for communication with the System Manager and other chassis-external entities, using RMCP or other application level protocols.

For configuration of IPv6-based services, the Chassis Manager supports new IPv6-related LAN configuration parameters, which are defined in the IPMI 2.0, rev 1.1 specification (parameters 50-80). These LAN configuration parameters, as well as the previously supported parameters, are stored persistently and are redundant (that is, changes to the parameters are passed from the active to the backup Chassis Manager over the redundancy interface and stored there). The parameters are stored in the files `/var/nvdata/ch1_param_ipv6_v2`, `/var/nvdata/ch2_param_ipv6_v2` (for LAN channels 1 and 2, respectively). From the user perspective, configuration can be done via the CLI, using the commands `setlanconfig` and `getlanconfig`.

The IPv6-based services in the Chassis Manager are designed as an addition, not replacement, of the existing IPv4-based services. So, it is not possible to make the Chassis Manager accessible only over IPv6 but not over IPv4.

IPv6-based services are always available for LAN channel 1, and for LAN channel 2, if the second network interface is used in the Active-Active mode. Use of IPv6 over a VLAN-based interface or over a bonded interface is also supported.

Also, IPv6 is not supported for inter-ChMM communication over the software redundancy interface; only IPv4 can be used for that communication.

It is possible to use IPv6 as the underlying protocol for LAN alert messages sent to remote destinations over the network. For that, the Chassis Manager supports an extended format for LAN configuration parameter 19 (Destination Addresses), which allows specifying an IPv6 address for a LAN alert destination. LAN alerts sent to such destinations use the IPv6 protocol and packet format.

The following IPv6-related LAN configuration parameters are essential for IPv6 operation on a LAN channel:

- IPv6/IPv4 Addressing Enables (51). This parameter must be set to the value 02h (Enabled IPv6 and IPv4 addressing simultaneously) to allow IPv6-based communication with the Chassis Manager over the channel. Setting this parameter to 00h (IPv6 addressing disabled) causes the Chassis Manager to stop listening to IPv6 addresses and sending IPv6-based LAN alerts. The parameter value 01h (Enabled IPv6 addressing only, IPv4 addressing is disabled) is not supported by the Chassis Manager, since it cannot operate in IPv6-only mode.
- IPv6 Static Addresses (57). For IPv6-based services, at least one static IPv6 address must be configured on the channel. Up to 4 addresses can be configured. RMCP service is available on any of these addresses. The Chassis Manager makes sure that an RMCP request received on a specific address is responded to from the same address. However, RMCP service is not available on the link-local IPv6 addresses that are automatically configured by the kernel (starting with the `fe80:` prefix). When static IPv6 addresses are assigned or removed, the Chassis Manager assigns the address to or removes it from the physical network interface of the ChMM via the corresponding ioctl() call.
- IPv6 Router Address Configuration Control (64). This parameter controls whether static IPv6 router addresses, assigned via corresponding LAN configuration parameters, are taken into consideration by the Chassis Manager. The value 03h of this parameter turns on the use of static routers, and the value 02h turns it off. Other values are not valid for the Chassis Manager; the Chassis Manager always honors dynamic router assignment performed via the IPv6 Router Discovery protocol.
- Static Router 1 IP Address (65), Prefix Length (67), Prefix Value (68); Static Router 2 IP Address (69), Prefix Length (71), Prefix Value (72). These are the configuration parameters for the two static routers that Chassis Manager can use if the parameter IPv6 Router Address Configuration Control turns on the use of static routers.

The following are the default values of the IPv6-related LAN parameters in the PPS provided RFS images:

- IPv6/IPv4 Addressing Enables = 02h (Enabled IPv6 and IPv4 addressing simultaneously.
- IPv6 Static Addresses (57) = not configured.
- IPv6 Router Address Configuration Control (64) = 03h (enable static and dynamic router address configuration).
- Static Router 1 IP Address (65), Prefix Length (67), Prefix Value (68); Static Router 2 IP Address (69), Prefix Length (71), Prefix Value (72) = not configured.

Note: The above parameters are reset to the default values, as assigned in the current RFS image, if the /var/nvdata files are erased, as would occur after re-initializing the file system or possibly after a reliable upgrade with, for example, the `--erase-all` *mode enabled.*

To configure the Chassis Manager with the default IPv6-related LAN parameter values for responding to RMCP traffic over IPv6, it is necessary at least to assign a static IPv6 address to the corresponding channel. For example, to assign the IPv6 address 3ffe:302:11:2::26, with the network part of 64 bits, as the first address (the address index starts from 0) to the LAN channel 1, the following command can be used:

```
# clia setlanconfig 1 ipv6_addresses 0 3ffe:302:11:2::26/64
```

Configuration of static routers is optional, depending on whether appropriate IPv6 routers exist and can be discovered dynamically (via the Router Discovery protocol) on the target local network. For example, if such dynamic discovery is not feasible or desirable, the following commands can be used to configure statically the default router with the address 3ffe:302:11:2::7f on the Chassis Manager:

```
# clia setlanconfig 1 ipv6_router1_ip 3ffe:302:11:2::7f
# clia setlanconfig 1 ipv6_router1_prefix_length 0
# clia setlanconfig 1 ipv6_router1_prefix_value ::
```

The values assigned via the `clia setlanconfig` command are stored persistently and survive a Chassis Manager reboot or switchover. In the case of a switchover, the static IPv6 addresses and other configuration information are failed over to the former backup Chassis Manager, which now becomes active, and RMCP communication can continue without interruption (as occurs with IPv4).

See the *Chassis Manager External Interfaces Reference* for further details on the IPv6-related LAN configuration parameters.

The Chassis Manager does not currently support obtaining IPv6 addresses and other information over the DHCP protocol (DHCPv6) from an appropriate server. This support may be implemented in future versions.

3.5 Configuring the FRU Information

This section describes configuring the Field Replaceable Unit (FRU) Information.

3.5.1 Accessing the Chassis FRU Information

According to the VITA 46.11 specification, the Chassis FRU Information should be redundant (at least two copies per chassis) and each copy may be represented by separate IPM Controllers, as FRU #1. Some VPX chassis adopt this approach fully. For such chassis, the default configuration file must be changed: the variable *LOCAL_SHELF_FRU* must be set to *FALSE*. This enables a chassis-wide search for potential sources of Chassis FRU Information on System IPMB.

On most VPX chassis implementing this approach, the Chassis FRU Information is accessed via two IPM Controllers with known IPMB addresses. In this case, it is possible to limit the search for Chassis FRU Information to the two pre-determined IPMB locations.

To do this, two configuration variables *SHELF_FRU_IPMB_SOURCE1* and *SHELF_FRU_IPMB_SOURCE2* must be defined in the configuration file */etc/shelfman.conf*. These variables are of the Number type and contain the IPMB addresses of the two designated IPM Controllers that represent Chassis FRU Information. For example, to limit the search for the Chassis FRU Information to IPM Controllers at 66h and 68h respectively, these variables should be defined as follows (note the use of "0x" prefix for hexadecimal addresses):

```
SHELF_FRU_IPMB_SOURCE1 = 0x66
SHELF_FRU_IPMB_SOURCE2 = 0x68
```

However, most known ChMM-based chassis provide two SEEPROMs that are connected to the Chassis Manager via the master-only I^2C interface, usually with each SEEPROM residing on its own bus behind an I^2C multiplexer.

Each of these two SEEPROMs stores a copy of the Chassis FRU Information, providing the needed redundancy. Some chassis do not provide even this type of storage; in that case it is possible to store the Chassis FRU Information on the ChMM itself, as a single Flash file */var/nvdata/chassis_fru_info*.

The redundant Chassis Managers each have their own copy of that Flash file, and synchronize them using the redundancy protocol, so even in that case some degree of redundancy is still preserved.

The Chassis Manager configuration (as represented in *shelfman.conf*) must be aligned with the mechanisms for accessing Chassis FRU Information that are provided by the chassis. The default configuration supports the approach where redundant Chassis FRU Information instances are represented by separate IPM Controllers as FRU #1 as well as the "two SEEPROMs" approach. The following key configuration variables are set as follows for that case:

```
LOCAL_SHELF_FRU = TRUE
SHELF_FRU_IN_EEPROM = TRUE
```

In this case, however, support for SEEPROMs must also be provided by the carrier-specific module in the Chassis Manager.

If none of the above approaches is supported by the chassis, and there are no sources of Chassis FRU Information represented by separate IPM Controllers, the system integrator must resort to the Flash file as storage for Chassis FRU Information. In that case, the following changes to the default configuration should be done:

- set the variable *SHELF_FRU_IN_EEPROM* to *FALSE*
- set the variable *MIN_SHELF_FRUS* to *1*

The last change is necessary because there is only one copy of the Chassis FRU Information on each ChMM. The variable *LOCAL_SHELF_FRU* must retain its default value of *TRUE*.

The following table summarizes the configuration variable settings that correspond to the various Chassis FRU Information source possibilities described above.

Table 11: Chassis FRU Information and configuration variable settings

SOURCE OF THE CHASSIS FRU INFORMATION	SETTINGS OF CONFIGURATION VARIABLES
Non-volatile storage (likely SEEPROMs), accessed via IPM Controllers on System IPMB	`LOCAL_SHELF_FRU = FALSE` `SHELF_FRU_IN_EEPROM =` does not matter `MIN_SHELF_FRUS =` minimum number of IPM Controllers on System IPMB providing the Chassis FRU Information (usually 2) Optionally: `SHELF_FRU_IPMB_SOURCE1 =` IPMB address of the first designated source `SHELF_FRU_IPMB_SOURCE2 =` IPMB address of the second designated source

SEEPROMs, accessed locally by the Chassis Manager	`LOCAL_SHELF_FRU = TRUE`
	`SHELF_FRU_IN_EEPROM = TRUE`
	`MIN_SHELF_FRUS =` the number of SEEPROMs providing the Chassis FRU Information (usually 2)

Flash file	`LOCAL_SHELF_FRU = TRUE`
	`SHELF_FRU_IN_EEPROM = FALSE`
	`MIN_SHELF_FRUS = 1`

3.5.2 Setting up the Chassis FRU Information

Since the contents of the Chassis FRU Information is crucial for successful management of the chassis, it is necessary to set up the Chassis FRU Information on a fresh chassis before starting the Chassis Manager on it. This procedure consists of the following steps:

- Creating a description of the chassis in a formalized text format (“INF format”).
- Compiling the text description using the FRU Information Compiler.
- Placing the binary image of the FRU Information into the appropriate storage.

The first two steps are documented separately in the user manual for the FRU Information Compiler. The last step is documented here and depends on where the Chassis FRU Information is stored.

The simplest case is if the Chassis FRU Information is stored on a Flash file on the ChMM. In that case, the binary image file should be downloaded on the ChMM via FTP and copied to the location *`/var/nvdata/chassis_fru_info`*.

The following log represents an example of the above process. It should be performed on an x86 Linux machine (NOT on the ChMM, itself!).

```
# mv chassis_fru.bin chassis_fru_info
# ftp 192.168.1.230
Connected to 192.168.1.230.
220 ChMM-230 FTP server (Version wu-2.6.2(1) Sun Dec 15 17:40:37 GMT 2002)
ready.
Name (192.168.1.230:serjio): ftp
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /var/nvdata
250 CWD command successful.
ftp> del chassis_fru_info
250 DELE command successful.
ftp> put chassis_fru_info
local: chassis_fru_info remote: chassis_fru_info
227 Entering Passive Mode (192,168,1,230,107,162)
150 Opening BINARY mode data connection for chassis_fru_info.
226 Transfer complete.
129 bytes sent in 5.1e-04 secs (2.3e+02 Kbytes/sec)
ftp> quit
221-You have transferred 129 bytes in 1 files.
221-Total traffic for this session was 640 bytes in 1 transfers.
221-Thank you for using the FTP service on ChMM-230.
221 Goodbye.
#
```

Another case is when the Chassis FRU Information is stored in SEEPROMs that are accessible to the Chassis Manager via the master-only I^2C bus. In that case, the binary file should be copied onto the ChMM and then written into the SEEPROMs. The utility *eepromw* can be used for that purpose.

The exact location of SEEPROMs on the master I^2C bus is carrier-specific, but typically they are located at address 0xA4 on channels 1 and 2 of the I^2C multiplexer, and the multiplexer itself resides at address 0xE0. In that case, the following commands (issued on the ChMM!) can be used to download the file */var/nvdata/chassis_fru_info* to the SEEPROMs. It is assumed that the file was compiled by the FRU Information Compiler and downloaded onto the ChMM, as in the previous example:

```
#eepromw –c 1 A4 /var/nvdata/chassis_fru_info
#eepromw –c 2 A4 /var/nvdata/chassis_fru_info
```

The general syntax for the *eepromw* utility is as follows:

eepromw [-b <multiplexer>] [–c <channel>] <eeprom-address> <file> [<count>]

where:

- *<multiplexer>* is the address of the I^2C multiplexer on the I^2C bus (default=0xE0),
- *<channel>* is the channel on the multiplexer to use (default=0),
- *<eeprom-address>* is the address of the target SEEPROM,
- *<file>* is the path to the file to write onto the target SEEPROM.
- *<count>* is how many bytes to write.

The optional parameter *<count>* specifies how many bytes to write. If this parameter is not specified, all the contents of the file are written into the target SEEPROM. The total number of bytes written to SEEPROM does not exceed the size of the SEEPROM.

The reciprocal *eepromr* utility allows the user to read the contents of an SEEPROM into a file on the Flash, and has the following parameters:

eepromr [-b <multiplexer>] [–c <channel>] <eeprom-address> <file> <count>

where:

- *<file>* is the path to the file where data are written from SEEPROM,
- *<count>* is how many bytes to read.

The remaining parameters are the same as for *eepromw*.

There is one caveat on using the utility *eepromw*. The Chassis Manager must not be running when the SEEPROM is updated, because the Chassis Manager can change the I^2C multiplexer settings trying to access an I^2C device behind the multiplexer while the update is in progress. This will cause the update to fail with high probability.

There are several ways to ensure that the Chassis Manager is not running on the ChMM. One is to turn off automatic start of the Chassis Manager by setting the U-Boot variable *start_rc2_daemons* to *n*. The other way is to terminate the Chassis Manager using the command *clia terminate*.

On some chassis, access to Chassis FRU Information SEEPROMs is also guarded by the ACTIVE# signal on the ChMM. In that case, make sure before running *eepromw* that the Hardware Redundancy Interface flags “Local Healthy” and “Active” are set for the local ChMM.

Use the *hwri* command to set flags mentioned above as well as “Presence” flag:

```
# hwri set presence healthy switchover
```

After that, make sure that the output of *hwri* (run without parameters) reports the “Active” flag set.

The remaining case, where the Chassis FRU Information resides on separate IPM Controllers inside the chassis, is completely chassis-specific and is beyond the scope of this document.

3.5.3 Setting up the Chassis FRU Information using the CLI

Because of the limitations and system dependencies of the approaches listed in the previous sections, another method is recommended to update the Chassis FRU Information. This method involves using the CLI command *frudataw* and it is applicable to all of the three cases considered in the previous section. This method requires that the Chassis FRU Information file to be placed on the active ChMM. After that, the following command can be used (possibly several times) to update the locations that contain the Chassis FRU Information:

clia frudataw <ipmc-address> <fru-id> <file>

where:

- *<ipmc-address>* is the address of the IPM Controller that contains the Chassis FRU Information;
- *<fru-id>* is the FRU device ID of that location;
- *<file>* is the name of the file on the ChMM that contains the new Chassis FRU Information image.

The Chassis Manager must be running when this command is issued. After the Chassis FRU Information is updated, the chassis must be completely restarted to accommodate the changes.

To update the Chassis FRU Information stored in a Flash file with the contents of the file *newdata.bin*, update a single location with the IPMB address 20h and FRU device ID 1, like this:

```
#clia frudataw 20 1 newdata.bin
```

To update the Chassis FRU Information stored in SEEPROMs with the contents of the file *newdata.bin*, use two commands to update locations with the IPMB address 20h and FRU device IDs 1 and 2, like this:

```
#clia frudataw 20 1 newdata.bin
#clia frudataw 20 2 newdata.bin
```

To update the Chassis FRU Information stored on separate IPM Controllers in the chassis, use several (usually two) commands with the corresponding IPMB addresses and FRU IDs. FRU ID is usually 1 in this case. Here is an example of updating the Chassis FRU Information in a chassis, where the Chassis FRU Information is stored on the IPM Controllers at 62h and 64h:

```
#clia frudataw 62 1 newdata.bin
#clia frudataw 64 1 newdata.bin
```

This method can be also used to update any other FRU Information repository in the chassis, if the IPMB address and the FRU device ID of that repository are known.

Note: The Chassis Manager doesn't support dynamic reconfiguration of the Chassis FRU Information. After any modifications of the ChassisFRU Information performed via CLI, RMCP, SNMP and other interfaces the whole chassis must be restarted to accommodate the changes.

3.5.4 Other FRU Information repositories

The Chassis Manager itself exposes at least one IPM Controller (the ChMC at IPMB address 20h). For most carriers, the Chassis Manager also exposes a "physical" IPM Controller that represents the resources of the carrier board and has an IPMB address derived from the physical address of the carrier. While the ChMC is exposed only by the active Chassis Manager and is subject to switchover, the "physical" IPM Controller is exposed separately by both active and backup Chassis Managers.

For both of these IPM Controllers, FRU Information is stored in Flash files on the ChMM:

`/var/nvdata/bmc-fru-information` for the ChMC

`/var/nvdata/shelfman-fru-information` for the "physical" IPM Controller

For some carriers, there may be additional FRUs represented by the ChMC.

The location of the FRU Information for these FRUs is carrier-specific.

Reading and writing these FRU Information repositories can be done via the IPMI commands "Read FRU Data", "Write FRU Data", addressed to the appropriate FRUs.

3.6 Configuring carrier and chassis attributes using HPDL

HPDL is a special description language used to describe the structure and device population of specific VITA 46.11 chassis, ChMM carriers and non-intelligent subsidiary FRUs managed by the Chassis Manager (such as Fan trays, PEMs, alarm panels). This language is called HPDL (which is short for Pigeon Point Hardware Platform Description Language).

A full description of the HPDL is available to chassis developers in a separate document (the *Pigeon Point HPDL Reference*).

HPDL descriptions are organized in units called HPDL modules. Each module corresponds to a single text file. One module can describe one or more carriers and/or chassis. The HPDL modules are compiled by a special compiler (`hpdlc`) and stored as binary data. The encoding of the binary data conforms to the ASN.1 Basic Encoding standard (ISO 8825). The Chassis Manager reads and interprets this data during initialization and obtains the following information from it:

- Population of the master-only I^2C bus on the carrier and chassis;
- Types and attributes of non-Intelligent FRUs managed by the Chassis Manager;
- Information about IPMI sensors exposed by the Chassis Manager;
- Associations between IPMI sensors and physical signals exposed by master-only I^2C devices.

Based on this information, the Chassis Manager configures master-only I^2C devices, creates FRU descriptors for non-intelligent FRUs, creates IPMI sensors and associates them with the corresponding FRUs.

In addition to HPDL modules, collections of SDRs are used to configure IPMI sensors on the carrier and in the chassis. These collections of SDRs are compiled with the SDR compiler, stored as binary data and retrieved by the Chassis Manager during initialization. The Chassis Manager uses these saved SDRs for the sensors described in HPDL that it creates during initialization.

3.6.1 Compiling HPDL definitions

HPDL definitions in text format are compiled by the HPDL compiler into a binary format that conforms to the ASN.1 basic encoding rules (ISO 8825) and stored in an output file. The HPDL compiler executable is available both for Linux (x86) and Windows operating systems (with the names *hpdlc* and *hpdlc.exe* respectively). The Linux version of the compiler can be run on various Linux distributions, including Red Hat, Mandriva Linux and Ubuntu.

The command line for the HPDL compiler has the following synopsis:

hpdlc [-d] [-v] <in-file> [<out-file>]

The parameter *<in-file>* specifies the input file that contains the HPDL descriptions in text format.

The optional parameter *<out-file>* specifies the output file; if this parameter is omitted, the name of the output file is based on the input file name with the suffix *.bin* appended at the end.

Option *-d* causes the HPDL compiler to output additional debugging information when compiling the input file.

Option *-v* causes the HPDL compiler to identify its version.

Error messages, if any, are output on the standard error stream of the HPDL compiler.

For example, the following command line compiles the input file *acb3.hpdl* into the binary file *acb3.hpdl.bin*:

```
>hpdlc acb3.hpdl
```

3.6.2 Compiling SDRs

Most of the attributes of sensors referenced in HPDL files are actually defined through Sensor Data Records (SDRs). The details of these SDRs are specified in text files and compiled to binary by a separate Python-based compiler. This compiler is available for both Linux and Windows operating systems; there is a separate *User Guide* for this compiler.

For example, the following command line compiles the input SDR file *acb3.sdr* into the binary SDR file *acb3.sdr.bin*:

```
>python sdrc.py acb3.sdr
```

3.6.3 Deploying HPDL data to the ChMM file system

The Chassis Manager first looks for HPDL data and SDRs in FRU Information areas (Chassis FRU Information for the chassis level data and carrier FRU Information for the ChMM carrier data). If the data is not found there, it is retrieved from the following files on the ChMM file system:

`/var/nvdata/carrier_data`	Binary HPDL definitions for the carrier
`/var/nvdata/carrier_sdrs`	Binary SDRs for the carrier sensors
`/var/nvdata/chassis_data`	Binary HPDL definitions for the chassis
`/var/nvdata/chassis_sdrs`	Binary SDRs for the chassis sensors

If the Chassis Manager cannot retrieve HPDL data from these locations, it uses the values of the following environment variables to locate the information:

CARRIER_HPDL	Binary HPDL definitions for the carrier
CARRIER_SDRS	Binary SDRs for the carrier sensors
CHASSIS_HPDL	Binary HPDL definitions for the chassis
CHASSIS_SDRS	Binary SDRs for the chassis sensors

Therefore, when HPDL definitions cannot be placed in some part of a FRU Information EEPROM, they are placed in the RFS and the environment variables listed above provide the locations inside the RFS (somewhere below the `/etc` directory).

To override the default HPDL definitions stored in the RFS, a user should copy the files generated by the HPDL and SDR compilers to the directory `/var/nvdata` on the ChMM file system by using, for example, FTP. The directory `/var/nvdata` is in the persistent part of the ChMM file system; that is, the file content persists across ChMM reboots.

3.6.4 Deploying HPDL data to FRU Information areas

FRU Information areas are the recommended deployment location for HPDL data and SDRs for the carrier and chassis (in carrier FRU Information and Chassis FRU Information, respectively). FRU Information for a specific FRU (Fan tray, Power Entry Module) can also be used; it provides FRU-specific HPDL data that amends definitions for that FRU in the chassis HPDL description. FRU Information is usually stored in EEPROM-type devices.

HPDL data and SDRs are stored in the FRU Information as a sequence of OEM-type multirecords, similar to the multirecords defined in the VITA specification. These multirecords have the following Manufacturer ID (IANA Enterprise Number) assigned to nVent, Schroff GmbH (formerly Pentair Technical Products) (00400Ah)† and the manufacturer-specific record type is 7 for HPDL data multirecords and 8 for SDR multirecords. The effective binary image for HPDL and SDR data is a

†This IANA Enterprise Number was originally assigned to Pigeon Point Systems, which has since been acquired by Pentair.

concatenation of the contents of all multirecords of the corresponding type that are present in a given FRU Information.

Since in many cases the capacity of EEPROM devices used to store the FRU Information is small, HPDL data and SDRs can be placed into the FRU Information in a compressed form. The GZIP compression algorithm is supported. Compressed data is detected automatically when retrieved by the Chassis Manager from the FRU Information and the data is automatically uncompressed in that case.

To support this deployment approach, several new features have been added to the Pigeon Point FRU Compiler and to the Pigeon Point Chassis Manager CLI (Command Line Interface); these features enable the user to incorporate compiled HPDL data and SDRs into the FRU Information.

3.6.4.1 Specifying the location of the Carrier FRU Information

The preferred location for the Carrier FRU Information is an EEPROM on the ChMM carrier. In addition, however, the Chassis Manager supports getting the Carrier FRU Information from a file on the ChMM file system; this can be useful if the ChMM carrier lacks an appropriate EEPROM.

However, if the Carrier FRU Information is stored in an EEPROM, the location and type of this EEPROM needs to be communicated to the Chassis Manager. Carrier HPDL data cannot be used for that because of a “chicken and egg” problem – in order to read carrier HPDL data, the Chassis Manager has to already know the location or the Carrier FRU Information.

So, the location of the Carrier FRU Information (if it is stored in an EEPROM) is specified as a carrier option *CARRIER_FRU_LOCATION* for the HPDL carrier (part of the string value of the Chassis Manager configuration parameter *CARRIER_OPTIONS*).

If the Carrier FRU Information is located in a file on the ChMM file system, the carrier option *CARRIER_FRU_LOCATION* must not be present.

The following syntax governs this carrier option:

```
CARRIER_FRU_LOCATION = <device-type>:<bus>:<address>[,<size>]
```

where

- *<device-type>* is the EEPROM device type (See Section 3 of the Pigeon Point HPDL Reference for the list of supported EEPROM devices). Typical values are *AT24C16*, *AT24LC256*, *ADM1026*.
- *<bus>* is the number of the master-only I^2C bus where the EEPROM is located (typically *0*)
- *<address>* is the 7-bit I^2C address of the EEPROM device on that bus, in hexadecimal
- *<size>* is an optional argument that specifies the size of the occupied portion of the EEPROM, that is, the number of bytes which the Chassis Manager should read to get the entire Carrier FRU Information. If this argument is not specified, the Chassis Manager reads the entire EEPROM.

For example, the following definition:

```
CARRIER_OPTIONS = "CARRIER_FRU_LOCATION = ADM1026:0:2E,2048"
```

specifies that the Carrier FRU Information should be retrieved from the SEEPROM of the ADM1026 device that is located on bus 0 at address 2Eh (7-bit), and that only the first 2048 bytes of this SEEPROM (which has the capacity of 8192 bytes) should be read.

3.6.4.2 Using the FRU Info Compiler to place HPDL and SDR data in FRU Information

The FRU Information Compiler is a Pigeon Point tool that generates a FRU Information image from its textual description. The image can then be stored in an appropriate EEPROM device using various techniques. The FRU Information Compiler implements special syntax that can be used to incorporate binary HPDL and SDR data into a FRU Information image that is being created.

For example, the following instructions add binary files *acb3.hpdl.bin.gz* and *acb3.sdr.bin.gz* (which are compressed HPDL and SDR files, respectively) to the current FRU Information image:

```
...
[nVent.HPDL Data]
File = acb3.hpdl.bin.gz
[nVent.HPDL SDR Data]
File = acb3.sdr.bin.gz
...
```

3.6.4.3 Using the Chassis Manager Command Line Interface

An extended version of the CLI command *frudataw* can be used to dynamically update any FRU Information in a chassis with HPDL data and SDRs. The corresponding binary files should be first downloaded on the ChMM.

The corresponding command line has the following syntax:

clia frudataw -d <ipmb-address> <fru-id> <in-file>

clia frudataw -s <ipmb-address> <fru-id> <in-file>

clia frudataw -d -c <ipmb-address> <fru-id>

clia frudataw -s -c <ipmb-address> <fru-id>

The parameters *<ipmb-address>* and *<fru-id>* specify the IPMB address and FRU device ID of the FRU Information to update. When updating HPDL data or SDRs in the Chassis FRU Information, use IPMB address 20h and FRU device ID 254, or the addresses of actual locations of the Chassis FRU Information.

The parameter *<in-file>* indicates the binary HPDL data or SDR file (which is possibly compressed) to be stored in the target FRU Information.

The option *-d* indicates that HPDL data is to be updated; the option *-s* indicates that SDRs data is to be updated.

The option *-c* removes (clears) the HPDL data or SDRs, respectively, from the target FRU Information.

For example, the following commands can be used to update the Chassis FRU Information with new versions of HPDL data and SDRs on a chassis where the Chassis FRU Information is stored in two redundant EEPROMs on the backplane, accessed locally by the Chassis Manager:

```
#clia frudataw -d 20 1 std14slot.hpdl.bin.gz
#clia frudataw -s 20 1 std14slot.sdr.bin.gz
#clia frudataw -d 20 2 std14slot.hpdl.bin.gz
#clia frudataw -s 20 2 std14slot.sdr.bin.gz
```

3.6.4.4 Placing HPDL data and SDRs in the FRU Information for specific FRUs

This section addresses placing HPDL data and SDRs into the FRU Information for a specific FRU (e.g. a fan tray). This information is used by the Chassis Manager to amend the attributes of that FRU from the attributes for that FRU that are specified in the HPDL chassis definition.

The following information about the FRU is supplied in the chassis definition and cannot be redefined by the HPDL data in the FRU:

- FRU site type and number
- Presence signal definition.

Device selection definitions can be redefined by the HPDL data in the FRU; however a problem arises when several slots exist in the chassis for the same type of the FRU, and addresses of devices of the FRU are different between different slots. In that case, it is not possible to define device location information on the FRU level; it is possible to do that only on the chassis level. This consideration limits the possibilities of redefining device information on the FRU level.

It is not necessary to have both HPDL data and SDRs in a given FRU Information area; only HPDL data or only SDRs may be present. For example, to change the power consumption of the FRU, it is sufficient to have only HPDL data. To change certain sensor attributes (e.g. the conversion formula, default threshold values or default hysteresis), it is sufficient to have only SDRs.

One problem with using HPDL data in a specific FRU is how to correlate the sensor numbers in the description to the actual sensor numbers. The sensor numbers are assigned within an IPM Controller, not within a specific FRU, so the same sensors on different FRUs of the same type have different numbers; and it is typically not known in advance in which slot a specific FRU is going to be placed. This problem is solved in the following way: the sensor numbers in the HPDL/SDR data for a specific FRU have relative numbers, and these numbers are correlated to the actual numbers in ascending order.

For example, assume that the replacement SDRs describe sensors 1, 2, 3 and 4, and the sensor numbers for the corresponding FRU in the HPDL chassis definition are 124, 208, 209 and 210. In that case, the replacement SDR for sensor 1 applies to sensor 124; the replacement SDR for sensor 2 applies to sensor 208, and so on.

3.7 Configuring the cooling management strategy

Different chassis vendors have different requirements for the cooling management strategy of the Chassis Manager. The VITA 46.11 specification contains several requirements related to cooling management that must be satisfied by all vendors (e.g. the specification requires the Chassis Manager to deactivate a FRU that is in a critical thermal condition). However, some vendors have their own vendor-specific requirements that need to be satisfied for specific chassis (e.g. "illuminate a certain LED when the thermal condition in the chassis becomes critical"). Until now, vendor-specific cooling management was implemented inside the Chassis Manager. With the advent of HPDL as a means to describe the architecture of carriers and chassis without additional coding, it became desirable to make cooling management modular, so that cooling management strategies can be added to the Chassis Manager without modification of the Chassis Manager code, and the proper cooling management strategy can be chosen dynamically.

To satisfy this goal, the following approach is supported in the Chassis Manager:

- Cooling management modules are implemented as Linux shared libraries (`*.so` files).
- A cooling management API is defined; this API fully defines the interface between the Chassis Manager and the cooling management module in the form of function calls.
- Support has been added to the Chassis Manager for dynamically loading the cooling management modules and binding to the functions that they export.

This approach is utilized only if HPDL is used to describe the carrier and the chassis. For non-HPDL-defined carriers and chassis, the cooling management strategy remains bound to carrier-specific code in the Chassis Manager.

3.7.1 Default cooling management strategy

With HPDL, a reasonable default cooling management strategy is provided. It is not necessary to configure the cooling management strategy if the default is satisfactory for a given carrier/chassis (which is likely the case for many carriers and chassis).

The default cooling management has the following features:

- Zoned cooling is supported; the Chassis Fan Geography record from the Chassis FRU Information is used to define cooling zones.
- In the normal cooling state (that is, when no thermal thresholds on any sensors are crossed), the Chassis Manager attempts to minimize the fan level, but at the same time prevent thermal alerts. It does that by adaptively choosing the lowest possible fan level that allows the chassis to avoid thermal alerts, for each fan, taking cooling zones into account.
- In the minor alert cooling state (non-critical thermal thresholds are crossed for one or more sensors) the Chassis Manager periodically increases the fan level for the fans that serve the cooling zone(s) where those thresholds have been crossed, until the fan level reaches its maximum or the thermal condition goes away.

- In the major alert cooling state (critical thermal thresholds are crossed for one or more sensors) the Chassis Manager sets the fan level to the maximum for the fans that serve the cooling zone(s) where those thresholds have been crossed.
- In the critical alert cooling state (non-recoverable thermal thresholds are crossed for one or more sensors) the Chassis Manager sets the fan level to maximum for the fans that serve the cooling zone(s) where those thresholds have been crossed. In addition, if the thermal condition is caused by a specific FRU, the FRU is deactivated. If the thermal alert is caused by a chassis-wide temperature sensor, all FRUs are deactivated. After a FRU is deactivated, it stays in the state M1 indefinitely and is not automatically powered back on when the thermal alert goes away; in that case an intervention by the System Manager or by an operator is needed to activate and power on the FRU.
- In addition, a fan management strategy is implemented that sets the fan level to the maximum for all fans in the cooling zone in the following cases:
  - If some fans are missing in the chassis, based on the fan population specified in the Address Table in the Chassis FRU Information, all fan trays that cool the same zone(s) as the missing ones, are set to maximum.
  - If one or more of the fan tachometer sensors have a major or critical threshold crossed (a fan is stopped or rotates too slowly), all fan trays that cool the same zone(s) as the underspeed fan(s) are set to maximum speed. The number of simultaneously underspeed fans that cause the remaining fans to go full speed is configurable via the configuration parameter *TURBO_MODE_MIN_FAN_FAILURES* (and the default value is 1). Setting this parameter to 0 turns off this feature.

3.7.2 Configuring a specific cooling management strategy

To configure a specific cooling management strategy, the following steps need to be taken:

1. Create a new cooling management strategy module (shared library) or choose one of the several libraries supplied with the Chassis Manager (in the RFS directory */lib*), that are based on cooling strategies from certain vendors.
2. If a new module is created, perform the following steps:
   - Implement the specific cooling management functionality, using the Chassis Manager Cooling Management API (which is described in a separate document).
   - Choose a name for the shared library that conforms to the following pattern: *libcooling_<xxx>.so*, where *<xxx>* is the name of the strategy, typically the vendor name.
   - Place the shared library either in the directory */lib* or in the directory */var/bin* on the ChMM.
3. Set the value of the configuration variable *COOLING_MANAGEMENT* in */etc/shelfman.conf* to the name of the strategy (*<xxx>* above) and restart the Chassis

Manager; thereafter, whenever it starts, the Chassis Manager loads and uses the designated shared library for cooling management.

3.8 Configuring local sensors

Note: This section applies only to chassis where HPDL is not used. With HPDL, sensor SDRs are defined as part of the carrier and chassis definition and do not require additional configuration from a file on the ChMM.

Local sensors on the ChMM can be configured when the ChMM is started. (This capability applies to sensors that are associated with either: 1) the Chassis Manager or 2) the physical IPMC that takes its System IPMB address from the hardware address of the ChMM carrier slot.) Only sensor attributes that are defined in Sensor Data Records (such as thresholds, hysteresis values, sensor name, linearization parameters, etc.) can be configured at this time. The Sensor Device Records (SDRs) defining these sensors are read from the file `/var/nvdata/user_sdr`.

This file must contain an array of binary SDRs that are compliant with the IPMI specification. However, these SDRs can contain only partial sensor definition, if only a subset of the attributes of the sensor need to be redefined (see below).

The Pigeon Point SDR compiler utility can be used to produce the binary SDRs from plain text human-readable text files. The SDR compiler can also decode binary SDR data and produce a human-readable text file from it. This utility is described in the *SDR Compiler User Guide*. The current version of the SDR compiler must be used.

In order to take advantage of this sensor configuration facility, you should install the SDR Compiler on either a Linux or a MS Windows system. Then you should create an SDR definition text file according to the format described in a *SDR Compiler User Guide*.

You can use standard text editors such as `vi` for Linux or Notepad for MS Windows. The standard extension for SDR definition text files is `.inf`. The SDR Compiler is a command-line utility. To produce a binary SDR definition file `user_sdr` from a text SDR definition file you should use the utility in the compilation mode. For example:

```
>python sdrc.py test.inf user_sdr
```

`user_sdr` is an optional name of an output file. If it is not indicated in the command line a binary file `test.bin` will be created by the SDR compiler.

The SDR Compiler is written in Python, so you also need a Python interpreter (version 2.3 or later) to run it. Python is available for free downloading, with support for both Windows and Linux at www.python.org.

After producing the binary SDR definition file, you should place it on your ChMM in the directory `/var/nvdata` under the name `user_sdr`. For example, you can use FTP:

```
>ftp 192.168.1.191
Connected to 192.168.1.191 (192.168.1.191).
220 chmm+120 FTP server (Version wu-2.6.2(1) Wed Oct 5 21:30:04 GMT 2005) ready.
```

```
Name (192.168.1.191:username):anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /var/nvdata
250 CWD command successful.
ftp> put user_sdr
local: user_sdr remote: user_sdr
227 Entering Passive Mode (192,168,1,191,246,195)
150 Opening BINARY mode data connection for user_sdr.
226 Transfer complete.
124 bytes sent in 8.5e-05 secs (7.6e+03 Kbytes/sec
ftp> quit
221-You have transferred 124 bytes in 1 files.
221-Total traffic for this session was 1165 bytes in 1 transfers.
221-Thank you for using the FTP service on chmm+120.
221 Goodbye.
```

A newly installed SDR definition file */var/nvdata/user_sdr* takes effect only after a restart of the Chassis Manager; the Chassis Manager log during that restart could look like the following example:

```
# daemon -f shelfman -lcs
# <I> 03:45:09.041   [152] Pigeon Point Chassis Manager ver. 1.2.0. Built on Oct 24 2017 15:23:14
<*> 03:45:09.055   [152] Limits: code=(10af4040:10ba95c0), end_data=10c355d4, start_stack=10c36e64, esp=10c36874, eip=10ba1838
<*> 03:45:09.063   [152] Stack limits: curr=800000, max=ffffffff
<*> 03:45:09.067   [152] Data limits: curr=ffffffff, max=ffffffff
<*> 03:45:09.090   [152] *** Lock log print buffer at 10c10e90 ***
<*> 03:45:09.094   [152] *** Pthread lock log print buffer at 10c14ed0 ***
<I> 03:45:09.380   [152] Enabling the CPLD Active bit workaround
<I> 03:45:09.435   [152] User SDR size = 124
<I> 03:45:09.453   [152] 20 # 120, Type = 1
<I> 03:45:09.467   [152] a0 # 2, Type = 1
…
```

If the SDR definition file */var/nvdata/user_sdr* is not present, the configurations of the local sensors are unmodified from the default established in the Chassis Manager, and the log looks like the following example:

```
# daemon -f shelfman -lcs
```

```
# <I> 17:11:28.004   [152] Pigeon Point Chassis Manager ver. 1.2.0. Built on Oct
24 2017 15:23:14
<*> 17:11:28.009   [522] Limits: code=(400000:529030), end_data=10062000,
start_stack=7fff7df0, esp=7fff7758, eip=2ab0d2e4
<*> 17:11:28.010   [522] Stack limits: curr=1ff000, max=7fffffff
<*> 17:11:28.010   [522] Data limits: curr=7fffffff, max=7fffffff
<*> 17:11:28.014   [522] *** Lock log print buffer at 1003c910 ***
<*> 17:11:28.014   [522] *** Pthread lock log print buffer at 10040940 ***
<W> 17:11:28.027   [522] Custom SDR initialization file is absent
…
```

It is important to understand that replacement SDRs must be closely coordinated with SDRs that are defined within the Chassis Manager. Therefore, additions or modifications to the set of replacement SDRs in a chassis should only be undertaken in close cooperation with the chassis supplier.

The following rules apply to SDRs that are used to configure local sensors (referenced as replacement SDRs, below):

- Every replacement SDR must be a Full Sensor Record (type 01). Compact Sensor Records (type 02) are not supported and are never used by the Chassis Manager for local sensors.
- The following fields are mandatory in every replacement SDR (note that this list is smaller than the list of mandatory fields for normal SDRs processed by the SDR compiler):
  - Sensor Owner ID
  - Sensor Number
  - Sensor Initialization
- If a field from a replacement SDR is used (as a result of the operation of other rules) to replace an attribute for a target sensor, and the field is not specified in the replacement SDR, a value zero (0) is used for this field.
- The tuple (Sensor Owner ID, Sensor Number) identifies the sensor that is to be configured. The Sensor Owner ID is a literal IPMB address; so for sensors on the physical IPM Controller two instances of the corresponding replacement SDR must be present, one each for the IPMB addresses associated with each of the two redundant dedicated ChMC slots.
- The fields Entity ID and Entity Instance, if specified in the replacement SDRs and different from (0,0), replace the corresponding attributes of the sensor.
- The Sensor Initialization field identifies the specific attributes to be redefined and indicates what fields in the rest of the replacement SDR are applicable. It is specified as a list of symbolic constants; the following constants are defined:
  - THRESHOLDS: indicates that the sensor thresholds are to be replaced with the corresponding threshold values specified in the replacement SDR

  - HYSTERESIS: indicates that the sensor hysteresis values are to be replaced with the corresponding hysteresis values specified in the replacement SDR
  - SENSOR_TYPE: indicates that the sensor type is to be replaced with the corresponding fields Sensor Type and Event/Reading Type from the replacement SDR
  - EVENTS: indicates that the event masks are to be replaced with the corresponding fields from the replacement SDR
- The field Sensor Capabilities cannot be replaced.
- If and only if the symbolic constant SENSOR_TYPE has been specified in the Sensor Initialization field, the fields Sensor Type and Event/Reading Type from the replacement SDR replace the corresponding attribute of the sensor (even if the values of these fields are 0).
- If and only if the symbolic constant EVENTS has been specified in the Sensor Initialization field, the following fields from the replacement SDR replace the corresponding attributes of the target sensor:
  - For Threshold-Based sensors:
    - Lower Threshold Reading Mask
    - Upper Threshold Reading Mask
    - Threshold Assertion Event Mask
    - Threshold Deassertion Event Mask
    - Settable Threshold Mask
    - Readable Threshold Mask
  - For Discrete sensors:
    - Assertion Event Mask
    - Deassertion Event Mask
    - Discrete Reading Mask
- The fields Sensor Units 1, Base Unit, Modifier Unit cannot currently be replaced.
- The following sensor attributes are replaced from the replacement SDR fields, if at least one of these fields is specified in the replacement SDR with a non-zero value:
  - Linearization
  - M
  - Tolerance
  - B
  - Accuracy
  - Accuracy exp
  - R exp

  - B exp
  - Analog Characteristic Flags
  - Nominal Reading
  - Normal Maximum
  - Normal Minimum
  - Sensor Maximum Reading
  - Sensor Minimum Reading
- If and only if the symbolic constant THRESHOLDS has been specified in the Sensor Initialization field, the fields listed below from replacement SDRs specify replacement threshold values for the target sensor. Note, however, that only thresholds supported by the sensor implementation can be redefined. (Some local sensors do not support all possible thresholds.) Here is a list of replaceable threshold types, all of which must be specified in the raw format (with the “0x” prefix):
  - Upper Non-Recoverable Threshold
  - Upper Critical Threshold
  - Lower Non-Critical Threshold
  - Lower Non-Recoverable Threshold
  - Lower Critical Threshold
  - Lower Non-Critical Threshold
    If and only if the symbolic constant HYSTERESIS has been specified in the Sensor Initialization field, the following fields from the replacement SDRs specify the replacement hysteresis values for the target sensor:
  - Positive Hysteresis
  - Negative Hysteresis
    Hysteresis values must be specified in the raw format (with the “0x” prefix).
- The field OEM from the replacement SDR always replaces the corresponding attribute of the target sensor (even if not specified).
- The field Id String replaces the target sensor name only if specified in the replacement SDR.

The example below illustrates a typical local sensor configuration text definition. It redefines thresholds for the two temperature sensors on the physical IPM Controller. It is assumed that the physical IPM Controllers have System IPMB addresses 10h and 12h in the target chassis.

```
[Full Sensor Record]
Owner Id = 0x10
Sensor Number = 2
Sensor Initialization = THRESHOLDS
Lower Non-Critical Threshold = 0xb0
```

```
Lower Critical Threshold = 0xc0
Lower Non-Recoverable Threshold = 0xd0
Upper Non-Critical Threshold = 0x40
Upper Critical Threshold = 0x48
Upper Non-Recoverable Threshold = 0x50
[Full Sensor Record]
Owner Id = 0x10
Sensor Number = 3
Sensor Initialization = THRESHOLDS
Lower Non-Critical Threshold = 0xb0
Lower Critical Threshold = 0xc0
Lower Non-Recoverable Threshold = 0xd0
Upper Non-Critical Threshold = 0x40
Upper Critical Threshold = 0x48
Upper Non-Recoverable Threshold = 0x50
[Full Sensor Record]
Owner Id = 0x12
Sensor Number = 2
Sensor Initialization = THRESHOLDS
Lower Non-Critical Threshold = 0xb0
Lower Critical Threshold = 0xc0
Lower Non-Recoverable Threshold = 0xd0
Upper Non-Critical Threshold = 0x40
Upper Critical Threshold = 0x48
Upper Non-Recoverable Threshold = 0x50
[Full Sensor Record]
Owner Id = 0x12
Sensor Number = 3
Sensor Initialization = THRESHOLDS
Lower Non-Critical Threshold = 0xb0
Lower Critical Threshold = 0xc0
Lower Non-Recoverable Threshold = 0xd0
Upper Non-Critical Threshold = 0x40
Upper Critical Threshold = 0x48
Upper Non-Recoverable Threshold = 0x50
```

3.9 Setting the Auxiliary Firmware Revision

The Auxiliary Firmware Revision can be set when the Chassis Manager is started. The Auxiliary Firmware Revision is reported by the “Get Device ID” command targeted to a physical ChMC (at the hardware-specified System IPMB address, versus the logical Chassis Manager at System IPMB

address 0x20) and is stored in a single Flash file */var/nvdata/aux-fw-revision*. If the file */var/nvdata/aux-fw-revision* is absent, the Auxiliary Firmware Revision is not defined.

According to IPMI v2.0 R1.0, Section 20.1 Get Device ID Command, the Auxiliary Firmware Revision is a 4-byte data item. The file */var/nvdata/aux-fw-revision* should contain the eight hexadecimal digits that represent the 4 bytes in question. No separators are allowed. The first two hexadecimal digits represent the most significant byte of the Auxiliary Firmware Revision.

For example, assume that the file */var/nvdata/aux-fw-revision* contains the string 'a0b1efcd'. When the Chassis Manager is started and the RMCPTA connection is established, it is possible to obtain the Auxiliary Firmware Revision via RMCP. The parameter *<IPMB-address>*, below, represents the IPMB-address of the alternative controller. Here is a dialogue with the Pigeon Point internal tool RMCPTA, but any RMCP client can be used to make this query:

```
RMCPTA{1}-> TargetFwd <IPMB-address>
RMCPTA{1}-> GetDeviceId
 Completion Code = 0x00 (OK)
 Device ID       = 0x08
 Device Revision = 0x0
 Device Mode     = normal operation ; Device SDR present
 Firmware Rev.   = 1.00
 IPMI Version    = 2.0
 Device Support  = Chassis; IPMB Req.Gen; IPMB Req.Rec; FRU; SEL; SDR; Sensor;
 Manufacturer ID = 0x0400A
 Product ID      = 0x0000
 AUX FW Rev.     = 0x01001C00 (01 00 1C 00)
```

3.10 Setting up the clock

When the system is brought up for the first time, the clock is not set and must be initialized. Initially the clock is set to January 1, 1970. The date can be accessed via the serial console.

```
# date
Thu Jan 1 03:16:30 UTC 1970
```

In order to change the date, you should type in the correct date using the date application. The format for the date command is MMDDhhmm[[CC]YY][.ss], where:

- MM - Month
- DD - Day
- hh - Hour (using 24 hour notation)
- mm - Minute
- ss - Second
- YY - Year (2-digit)
- CC - Century

For example:

```
# date 042916282006
Sat Apr 29 16:28:00 UTC 2006
```

To make the date persistent, you need to store it using the *hwclock* application.

```
# hwclock -w
```

In some cases, you might get the error message:

```
mktime: cannot convert RTC time to UNIX time
```

This error can be ignored. It is due to the original date being in an uninitialized state.

3.10.1 Obtaining date and time from a time server

It is possible to obtain the system date and time from a time server during system startup and synchronize it periodically thereafter. (This facility is critical if the ChMM carrier does not have an RTC battery.) There are two network time protocols that can be used for that purpose: NTP and RFC 868 (rdate). The specific protocol to be used is selected when configuring the ChMM.

To enable obtaining the network time via the NTP protocol, it is necessary to define the U-Boot variables *time_proto*, *time_server* and optionally the additional variable *timezone*. The variable *time_proto* determines the adjust time protocol (if this variable is undefined, by default the RFC 868 (rdate) protocol is used). This variable should be set to *ntp* to enable the NTP protocol. The usage of the other variables is identical their usage with RFC 868 (rdate) and is described below.

To enable obtaining the network time via the RFC 868 protocol (rdate) over TCP, it is necessary to define the U-Boot variable *time_server* and optionally the additional variable *timezone*. The variable *time_proto* should be left undefined or set to *rdate*.

The variable *time_server* contains the IP address of the time server that the Chassis Manager queries for the system time after startup. This server should support RFC 868 over TCP as required by the *rdate* utility or support NTP as required by the *ntpdate* utility. This variable is propagated to the Linux level as the environment variable *TIMESERVER*.

If this variable is set, the startup script */etc/netconfig* starts the script */etc/timesync* as a daemon, which runs in an endless loop and queries the time server with a default interval of 300 seconds. To change this interval, edit the script */etc/timesync* and change the value of the variable *INTERVAL*. If a query fails, the script starts polling the server more frequently (with the default interval of 30 seconds); this new interval is the value of the script variable *INTERVAL2*. When a subsequent query succeeds, the polling rate reverts back to *INTERVAL*. The *TIMESERVER* variable can be changed by the Chassis Manager if the Chassis Manager is configured to receive its RMCP IP address via DHCP. In that case, depending on the configuration of the DHCP server, the Chassis Manager can also receive the IP address of the NTP server via DHCP. It stores this address in the */tmp/timeserverip* file, which is used by the script */etc/timesync* to override the current value of the *TIMESERVER* variable.

Note that during each reboot of the Chassis Manager the clock is reset to January 1, 1970. Therefore, the system time stays incorrect until the script `/etc/timesync` succeeds in obtaining correct time from the time server.

The variable `timezone` contains the name of the current time zone followed by its offset from Greenwich Meridian Time (GMT). The offset is positive for time zones to the west of Greenwich and negative for time zones to the east of Greenwich. This variable is propagated to the Linux level as the environment variable *TZ*. The default value of this variable is *UTC*, i.e. Universal Coordinated Time which matches Greenwich time. Note that textual timestamps generated on a ChMM correspond to the local time zone. That is, these timestamps include the offset defined by the environment variable *TZ*.

The time sent by time servers is GMT time; if the time zone on the Chassis Manager is not set or not set correctly, the time obtained from the time server is interpreted incorrectly. The three-letter name of the time zone is not used by the Chassis Manager, but is propagated to set the Linux time zone. (For instance, if the time zone name XXX0 is used, the date command produces output like the following: "Thu Sep 9 21:24:24 XXX 2004".) Daylight saving time is not supported.

Here is an example of a time zone definition for US Eastern Time:

```
timezone = EST5
```

Here the digit 5 specifies that the time zone is 5 hours west of GMT. Any three letters can replace *EST*; they are used to identify the time zone in (for example) the Linux date command output.

For timezones with half-hour or quarter-hour differences from an integer offset (e.g. `UTC+05:30`), it is possible to use the `/etc/localtime` file. Due to limited RFS space, there is no time zone database on ChMM, but such file can be copied from a Linux system (e.g. `/usr/share/zoneinfo/Asia/Calcutta`) and installed on a ChMM as `/etc/localtime`. If this file is detected at ChMM startup, the *TZ* variable is not set, so date-related library calls use the time zone specified in `/etc/localtime`.

3.11 Setting up and using ChMM Power On Self Tests

The available Power On Self Test (POST) tests are built into U-Boot. The choice of the available POST tests that are actually executed is controlled by the dedicated U-Boot environment variables `post_normal` and `post_poweron`.

The value of the environment variable `post_normal` contains names of tests that are executed on each boot-up. These names are separated by space characters. These tests do not take much time and can be run on a regular basis.

The value of the environment variable `post_poweron` contains names of tests that are executed after power-on reset only (vs. on each boot-up). These names are separated by space characters.

As the POST tests are executed, the results are logged in a textual form in a dedicated area in SDRAM. Results for each particular test have the following form:

```
<4>POST <test name> [<test-specific output>] [PASSED|FAILED]
```

The POST framework provides interfaces for accessing the results of the POST tests in U-Boot and in Linux.

U-Boot supports a log show command, which can be used to access the POST test results. This command outputs the contents of the POST log buffer onto the serial console.

```
ChMM700 log show
<4>POST uart UART 0 test failed
<4>FAILED
<4>POST crc PASSED
```

In a POST-enabled configuration, the Linux kernel shares its internal message log buffer with the POST log buffer. This causes POST results to be automatically displayed on the serial console during the kernel bootstrap. Interactively, the user can access the kernel log buffer (and thus the POST test results) using the *dmesg* command implemented by *busybox*.

The defined names for POST tests are:

- *memory* (SDRAM tests, recommended for execution on the first boot-up after power-on)
- *uart* (UART verification, specific to the Au1550 processor, recommended for execution on each boot-up)
- *ethernet* (Ethernet MACs test, specific to the Au1550 processor, recommended for execution on each boot-up)
- *i2c* (Master-Only I^2C test, recommended for execution on each boot-up)

The names of the tests can be used in values of the *post_poweron* and *post_normal* environment variables.

The IPMI command “Get Self Test Results” directed to the logical Chassis Manager (IPMB address 20h) returns the results of the POST performed by U-Boot at the startup of the ChMM. If all tests have passed, the status code 0x55 is returned. If any tests have failed, the device-specific failure code 0x59 is returned. The third byte of the response contains the following bit mask in that case:

[7:5]	Reserved
[4]	1b = Ethernet test failed
[3]	1b = UART test failed
[2]	1b = U-Boot CRC test failed
[1]	1b = I^2C test failed
[0]	1b = Memory test failed.

3.12 Configuring external event handling

Sometimes there is a need for specific IPMI events to be processed by a user-defined application or script that is external to the Chassis Manager, but still executing on the ChMM. This is called

"external event handling". This feature allows extensions of Chassis Manager functionality by user programs or scripts executed on the ChMM.

External event handling is implemented via the PEF alerting mechanism. As an extension of this mechanism, all alerts with a destination channel set to the System Interface Channel (Fh) are delivered by the Chassis Manager to the standard input of a designated external event handler program. This program is specified via the configuration parameter *EXTERNAL_EVENT_HANDLER*.

3.12.1 Detailed steps to configure external event handling

To use external event handling the user should take the following steps:

1. Prepare a script or executable for external event handling and place it on the Chassis Manager Flash filesystem (for example, into the directory `/var/bin`).
2. Specify the location of the external event handler in the Chassis Manager configuration file(`/etc/shelfman.conf`) as the value of the configuration parameter *EXTERNAL_EVENT_HANDLER*, for example:
   `EXTERNAL_EVENT_HANDLER = /var/bin/ext_handler.sh`
   If the configuration parameter *EXTERNAL_EVENT_HANDLER* is not defined or is defined as an empty value, external event handling is disabled.
3. Start the Chassis Manager.
4. Enable PEF via the CLI, if not enabled yet:
   `# clia setpefconfig control 1`
5. Enable alerting, if not enabled yet:
   `# clia setpefconfig action_control 1`
6. Set alert strings if necessary:
   `# clia setpefconfig alert_string 1 "test string"`
   In the example above, *1* is the number of the string; to set multiple strings, use sequential string numbers *1*, *2*, etc. for subsequent strings.
7. Set at least one alert policy that sends alerts to the destination channel Fh (the IPMI System Interface):
   `# clia setpefconfig alert_policy 1 1 8 F 1 1`
   where the fields in the example have the following meanings (see the Pigeon Point Chassis Manager External Interface Reference document for detailed descriptions of the `setpefconfig` and other CLI commands):
   - *1* - number of alert policy table entry
   - *1* - policy number
   - *8* - policy (enabled, always send alert to this destination)
   - *F* - destination channel
   - *1* - destination selector

- *1* - alert string selector

8. Set event filters for relevant events. For example, the following event filter can be used to capture state transitions to state M0 of FRU #0 on IPMC 9Ch:
```
# clia setpefconfig event_filter 1 80 1 1 0 9C FF F0 FF FF FF FF 0F FF 0 0 0
0 FF FF 0
```
where:

- *1* - filter number
- *80* - filter configuration (enabled, software configurable filter)
- *1* - filter action - alert
- *1* - alert policy number
- *0* - event severity (unspecified)
- *9C* - Slave Address or Software ID from Event Message.
- *FF* - channel Number / LUN to match
- *F0* - type of sensor
- *FF* - sensor
- *FF* - event trigger
- *FF* - Event Data 1 Event Offset Mask
- *FF* - Event Data 1 Event Offset Mask
- *0F* - Event Data 1 AND Mask
- *FF* - Event Data 1 Compare 1
- *0* - Event Data 1 Compare 2
- *0* - Event Data 2 AND Mask
- *0* - Event Data 2 Compare 1
- *0* - Event Data 2 Compare 2
- *FF* - Event Data 3 AND Mask
- *FF* - Event Data 3 Compare 1
- *0* - Event Data 3 Compare 2

Note: when an event handler is placed in the directory `/var/bin` *as assumed in the example above, care must be taken to preserve the state of that directory across reliable firmware upgrades (see section 7.6 for details). Also note that the* `/var/bin` *directory is erased when the U-Boot variable* `flash_reset` *is set to* `y` *(see section 6.3 for details).*

3.12.2 External event handler operation

The external event handler can be a user-provided binary executable program or a shell script. The handler is invoked by the Chassis Manager when an event matching a relevant PEF filter occurs. During that invocation, a pipe is created between the Chassis Manager and the external event handler. This pipe serves as standard input for the external event handler.

For each event, the Chassis Manager submits a single line of text into the pipe in the following format:

```
raw="raw data" alert_string="alert string data"
```

The raw data are the 16 bytes of the SEL record, with the characters representing each byte separated by a colon (“:”) from the next byte. For example:

```
raw="36:3:2:27:df:49:0:20:0:4:f1:18:6f:a1:0:78" alert_string="test"
```

The external event handler should read its standard input line by line and process each line as a separate event.

There are possible two approaches to event processing in the external event handler:

- Read the standard input line by line and process incoming data. In that case, one instance of the external event handler is spawned in the beginning and handles all subsequent events.
- Read one line from the standard input, process the data on that line and exit. In that case, the Chassis Manager spawns a new instance of the external event handler for the next event.

Here is an example of a simple external event handler script:

```
#!/bin/sh
while read line; do
    raw=""
    eval $line;
    if [ ! -z "$raw" ]; then
        # do event processing here
        echo "event $raw!"
    fi
done
```

The following external event handler script example is more sophisticated. It powers down either IPMC or the entire chassis on Upper Non-Recoverable threshold crossing event from any voltage sensor, based on event source address. For sensors at BMC 20h the entire chassis is powered down, otherwise only the event source IPMC is powered down. The “raw” variable contains 16 bytes of a SEL entry defined in IPMI 2.0 specification and the script extracts source IPMB address, sensor type (02h for voltage), event direction and reading type (01h for asserted threshold events) and threshold type (lower half of the event data byte#1) from it.

```
#!/bin/sh
LOGGER="logger -t EVENTHAND -p 6 -s"
```

```
while read line; do
  raw=""
  eval $line;
  if [ "$raw" ]; then
    #$LOGGER "raw=$raw"
    source=`echo $raw | cut -d: -f 8`
    sentype=`echo $raw | cut -d: -f 11`
    direction=`echo $raw | cut -d: -f 13`
    threshold=`echo $raw | cut -d: -f 14 | cut -c2`
    $LOGGER "source=$source sentype=$sentype direction=$direction
threshold=$threshold"
    # voltage sensor type & asserted event & UNR threshold
    if [ "$sentype" = "2" -a "$direction" = "1" -a "$threshold" = "b" ]; then
      if [ "$source" = "20" ]; then
        $LOGGER "Powering down the chassis via IPMI:ChassisControl(power-down)"
        clia sendcmd 20 0 2 0
      else
        $LOGGER "Deactivating IPMC 0x$source via
PICMG:SetFruActivate(deactivate)"
        clia deactivate $source 0
      fi
    fi
  fi
done
```

3.13 Configuring the Platform Event Trap format

The Chassis Manager sends SNMP Traps as a result of event processing when an Alert action is initiated or as a result of the “Alert Immediate” command which is used to test Alert destinations. The Platform Event Trap format is controlled by the *PET_FORMAT* configuration variable. Currently there are three supported formats. All formats use the same enterprise OID = `iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).wired_for_management(3183).PET(1).version(1)`, but have different OIDs for the embedded variable(s). A separate MIB file for supported SNMP traps is available.

By default or with *PET_FORMAT* set to *0*, the Chassis Manager generates trap messages according to IPMI Platform Event Trap Format v1.0 specification. In this format, event data is packed as binary in a single variable with OID=`.1.3.6.1.4.1.3183.1.1.1`. A sample application *snmptc* is available for parsing this format.

If *PET_FORMAT* is set to *1*, the Chassis Manager generates trap messages in plain text format, packing event into single variable with OID=`1.3.6.1.4.1.3183.1.1.2` as an ASCII string. This format is convenient for visual monitoring without specialized tools.

If *PET_FORMAT* is set to *2*, Chassis Manager generates trap messages in multi-variable format, packing each field of the event (event number, timestamp, generator etc.) as a separate variable with OIDs in the `.1.3.6.1.4.1.3183.1.1.3.[1..13]` range. Processing traps in this format requires the MIB file.

It is possible to change the Platform Event Trap format dynamically using the *`clia setpefconfig pet_format <N>`* command with *<N>* = *0*..*2*. The command *`clia getpefconfig pet_format`* shows the currently used PET format.

Here is some sample output of the standard Linux *`snmptrapd`* tool for the supported formats:

PET_FORMAT = 0

```
    Received 111 bytes from 192.168.0.2
    192.168.0.2: Enterprise Specific Trap (2453248) Uptime: 191 days,
4:23:00.02,
        PPS-PET-MIB::ipmi-trap-data = Hex-STRING: BF CE 7A 60 AE 50 11 DC 00 80
00 50 C2 3F CD 24 00 01 12 BE 74 62 00 00 20 20 02 20 C1 00 00 00 FF FF 00 00 00
00 00 19 0A 40 00 00 00 00 C1
```

PET_FORMAT = 1

```
    Received 251 bytes from 192.168.0.2
    192.168.0.2: Enterprise Specific Trap (2453248) Uptime: 0:00:41.04,
        PPS-PET-MIB::ipmi-trap-text = STRING: "RecordID:0004;
RecordType:Platform Event; Time:Dec 19 17:11:43 2007; Generator:20 LUN0 Chan0;
SensorType:25; SensorNumber:C1; EventType:6F; EventDirection:asserted;
EventData: 00 FF FF"
```

PET_FORMAT = 2

```
    Received 296 bytes from 192.168.0.2
    192.168.0.2: Enterprise Specific Trap (2453248) Uptime: 489 days,
11:31:17.15,
        PPS-PET-MIB::ipmi-trap-record-id         = INTEGER: 4,
        PPS-PET-MIB::ipmi-trap-record-type       = INTEGER:
platformEvent(2),
        PPS-PET-MIB::ipmi-trap-timestamp         = INTEGER: 1198084292,
        PPS-PET-MIB::ipmi-trap-generator-address = INTEGER: bmc(32),
        PPS-PET-MIB::ipmi-trap-generator-lun     = INTEGER: lun0(0),
        PPS-PET-MIB::ipmi-trap-generator-channel = INTEGER: ipmb(0),
        PPS-PET-MIB::ipmi-trap-sensor-type       = INTEGER:
entityPresence(37),
        PPS-PET-MIB::ipmi-trap-sensor-number     = INTEGER: 193,
        PPS-PET-MIB::ipmi-trap-event-type        = INTEGER:
sensorSpecific(111),
```

```
        PPS-PET-MIB::ipmi-trap-event-direction   = INTEGER: asserted(0),
        PPS-PET-MIB::ipmi-trap-event-data        = Hex-STRING: 00 FF FF ,
        PPS-PET-MIB::ipmi-trap-entire-record     = Hex-STRING: 04 00 02 C4 50 69
47 20 00 04 25 C1 6F 00 FF FF
```

3.13.1 Parsed example of SNMP Trap

The following is a parsed version of sample SNMP Trap in the default (IPMI PET v1.0) format. The original event which is reported by this trap follows:

```
Event: at Tue Aug 5 16:29:21 2008; from:(0x20,0,0); sensor:(0x12,0x85);
event:0x6f(asserted): 0xC0 0x00 0x00
```

This trap describes a system-reconfigure event, which is generated during Chassis Manager startup and switchover.

In the example below, a bold hexadecimal represents the Tag of a field and an underlined hexadecimal represents the length of a field.

```
---message length 125-----------------------------
30 7B 02 01  00 04 06 70  75 62 6C 69  63 A4 6E 06
09 2B 06 01  04 01 98 6F  01 01 40 04  C0 A8 00 02
02 01 06 02  03 12 6F 00  43 04 61 A3  ED 13 30 4D
30 4B 06 0A  2B 06 01 04  01 98 6F 01  01 01 04 3D
1D 3A 30 76  62 F2 11 DD  00 80 00 50  C2 3F FB 56
00 07 13 ED  A3 61 00 00  20 20 01 20  85 00 00 C0
00 00 00 00  00 00 00 19  00 00 40 0A  00 00 80 4B
01 54 65 73  74 53 74 72  69 6E 67 00  C1
--------------------------------------------------
```

Here is a detailed description of the content of this trap:

30 7B - ASN.1 tag and remaining length;
02 01 00 - SNMP version field, len = 1, 0=SNMPv1;
04 06 70 75 62 6C 69 63 - community field, "public";
A4 6E - PDU Trap command tag and remaining length;
06 09 2B 06 01 04 01 98 6F 01 01 - Agent ID, Enterprise OID(.1.3.6.1.4.1.3183.1.1);
40 04 C0 A8 00 02 - Agent IPv4 (192.168.0.2);
02 01 06 - Generic Status, len=1, EnterpriseSpecific;
02 03 12 6F 00 - Specific Status, len=3, (SenType=12h, EventType=6Fh, EventOffset=00h, Direction=Asserted);
43 04 61 A3 ED 13 - Time in ticks (0.01sec), obviously wrong...;
30 4D - ASN.1 tag at variable bindings start and remaining length;
30 4B - First variable header and remaining length;
06 0A 2B 06 01 04 01 98 6F 01 01 01 - Variable OID(.1.3.6.1.4.1.3183.1.1.1);
04 3D - Variable data length, data follows (see Table 17-7 “PET Variable Bindings Field” of IPMI 2.0 specification);
1D 3A 30 76 62 F2 11 DD 00 80 00 50 C2 3F FB 56 - GUID;

00 07 - trap sequence;
13 ED A3 61 00 00 - Local Timestamp (Tue Aug 5 16:29:21 2008);
20 20 - trap source = event source = IPMI;
01 - Event Severity (Monitor);
20 - Sensor Device (BMC)
85 - Sensor Number;
00 – Entity (00h=unspecified)

00 – Entity Instance (00h=unspecified_
C0 00 00 00 00 00 00 00 - Event Data;
19 - Language Code (English);
00 00 40 0A - Manufacturer ID, MSB first (00400Ah[†] assigned to nVent, Schroff GmbH (formerly Pentair Technical Products));
00 00 - System ID, MSB first;
80 4B 01 - PET multirecord, ASCII, len=Bh, Text Alert String;
54 65 73 74 53 74 72 69 6E 67 00 - Null terminated Alert String "TestString";
C1 - No more fields;

3.14 Configuring system services

The Chassis Manager filesystem contains several network services that can be used for debug and upgrade purposes. The following table contains a list of ports used by the Chassis Manager with a brief description of the service configuration with each port:

[†]This IANA Enterprise Number was originally assigned to Pigeon Point Systems, which has since been acquired by Pentair.

Table 12: Network services

PORT NUMBER/TYPE	NETWORK SERVICE DESCRIPTION
21/tcp	*FTP server* Activated via the */etc/inetd.conf* file; to disable the FTP server, comment out the '*ftp*' line in that file. This service is based on *WU FTP* daemon, so the configuration is located in the */etc/ftp** files, mostly in */etc/ftpaccess*. The FTP server can be configured to listen to IPv6 sockets. To achieve that, it is necessary to replace 'tcp' with 'tcp6' in the corresponding line in the */etc/inetd.conf* file: **ftp stream tcp6 nowait root /bin/in.ftpd -a** This option is turned off by default, because the FTP server does not correctly show the peer IP address in the system logs in the case of an IPv6 connection.
22/tcp *22/tcp6*	*SSH server* Activated via the*/etc/inetd.conf* file; to disable the SSH server, comment out the '*ssh*' line in that file. Run *sshd --help* to see the list of supported arguments that can be specified. The SSH server listens to both IPv6 and IPv4 sockets, so SSH sessions can be established over both IPv6 and IPv4.
23/tcp	*Telnet server* Activated via the */etc/inetd.conf* file; to disable the Telnet server, comment out the '*telnet*' line in that file. Run *telnetd --help* to see the list of supported arguments that can be specified. The telnet server can be configured to listen to IPv6 sockets. To achieve that, it is necessary to replace 'tcp' with 'tcp6' in the corresponding line in the */etc/inetd.conf* file: **telnet stream tcp6 nowait root /bin/telnetd** This option is turned off by default, because the telnet server does not correctly show the peer IP address in the system logs in the case of an IPv6 connection.

PORT NUMBER/TYPE	NETWORK SERVICE DESCRIPTION
80/tcp	*HTTP server with the Chassis Manager Web interface* Activated by the */etc/rc.common* script if the U-Boot variable *start_rc2_daemons* is set to *y*; to disable the HTTP server, comment out the corresponding line in the script. The HTTP server supports the configuration file */etc/httpd.conf* file where it is possible to specify allowed and denied IP ranges and the access login and password, for example: A:192.16 # Allow any address that begins with 192.16. D:* # Deny from other IP connections /cgi-bin:username:password where password is generated via the *httpd –m password* command.
161/udp *161/udp6*	*SNMP server with Chassis Manager SNMP interface* Activated by the */etc/rc.common* script if the U-Boot variable *start_rc2_daemons* is set to *y*; to disable the SNMP server, comment out the corresponding line in the script. The SNMP server supports the configuration file */etc/snmpd.conf*. The validation criteria for incoming SNMP requests are hardcoded in this configuration file. For SNMP v1/v2c, the community string in an incoming request must be *public*. For SNMP version 3 requests, the user name in an incoming request must be *overlord*. These parameters can, however, be changed by editing the configuration file */etc/snmpd.conf*. The SNMP server handles requests over both IPv6 and IPv4. To disable IPv6 support, comment out the *agentaddress* and *com2sec6* directives in the configuration file.
514/tcp	*RSH server* Activated by the*/etc/inetd.conf* file; to disable the RSH server, comment out the '*shell*' line in that file. Run *in.rshd --help* to see list of supported arguments that can be specified.

PORT NUMBER/TYPE	NETWORK SERVICE DESCRIPTION
623/udp	*Chassis Manager RMCP interface* Activated by the */etc/rc.common* script if the U-Boot variable *start_rc2_daemons* is set to *y*; to disable the RMCP interface, comment out the corresponding line in the script. This interface is available only on the active ChMM and only if the Chassis Manager is operational. The Chassis Manager only accepts connections targeted to configured RMCP address(es).
1040/tcp	*Chassis Manager Redundancy interface* This is the inter-ChMM communication interface needed in redundant configuration. It only accepts connection targeted to the IP address specified as the configuration parameter *REDUNDANT_IP_ADDRESS* in the Chassis Manager configuration file.

3.15 Configuring the speed of I²C buses

Two of the I²C buses (with logical numbers 2 and 3) support both high speed (400 kHz) and normal speed (100 kHz) operation. By default, both buses operate in high-speed mode, but this can be changed by means of a Linux kernel startup parameter *i2c-mxs.speed*, which has the following syntax:

i2c-mxs.speed=<speed1>,<speed2>

where *<speed1>* and *<speed2>* can have values *100* or *400* and represent the speed of buses 2 and 3, respectively.

According to normal Linux kernel naming conventions, the name of the parameter is composed of two parts separated by the dot symbol: *i2c-mxs* corresponds to the I²C driver name, and *speed* is the parameter name within that driver.

For example, to set both buses to normal speed, the following parameter should be passed to the Linux kernel:

```
i2c-mxs.speed=100,100
```

The recommended method to pass this parameter to the kernel is to append it to the value of the U-Boot variable *bootargs_initrd*, for example, by typing the following at the U-Boot prompt:

```
ChMM700 setenv bootargs_initrd $(bootargs_initrd) i2c-mxs.speed=100,100
ChMM700 saveenv
```

On ChMM-700, to provide compatibility with designs based on previous generation of ShMM modules, bus mapping takes place on the OS level. For example, logical I²C bus 0 (represented by the device node /dev/i2c0) that goes to the ChMM-700R carrier and accommodates carrier-based I²C devices, is actually physical I²C bus 4. Logical I²C bus 1 (represented by the device node

/dev/i2c1) that accommodates on-ChMM I^{2}C devices, is actually physical I^{2}C bus 5. The details of the mapping are shown in the table below.

Table 13: OS level bus number mapping for ChMM-700

LOGICAL BUS NUMBER (/DEV/I2C<N>)	PHYSICAL- BUS NUMBER (ACCORDING TO THE CHMM-700R HARDWARE SPECIFICATION)
0	4
1	5
2	0
3	1
4	2
5	3

3.16 Configuring VPX Control/GPIO signals

In the Chassis Manager context the VPX signals *GDISCRETE1*, *NVMRO*, *SYSRESET** may be mapped to GPIO signals located on the ChMM-700 itself or on a dedicated I^{2}C GPIO device. The mapping is established in the Carrier HPDL file.

In order to configure the signals the user should add definitions of the signals to the Carrier HPDL file, and add expressions to the EXPRESSIONS sections, which map signal names (used externally) to specific signal definitions. A subset of these three signals may be defined; all defined signals should be configured as outputs.

Then, the user should compile the Carrier HPDL definitions by the HPDL compliler *hpdlc* and deploy the binary Carrier HPDL file to the ChMM carrier (section 3.6).

In the example below the signals *GDISCRETE1*, *NVMRO*, *SYSRESET** are mapped to the local GPIO_1, GPIO_2, GPIO_3 signals of the ChMM-700R, respectively.

```
SHM_GPIO {
…
        GDISCRETE1:     SHMM_GPIO_E1, out;
        NVMRO:          SHMM_GPIO_E2, out;
        SYSRESET:       SHMM_GPIO_E3, out;
     };

EXPRESSIONS {
```

```
…
        GDISCRETE1:  SHMM_GPIO.GDISCRETE1;
        NVMRO:       SHMM_GPIO.NVMRO;
        SYSRESET:    SHMM_GPIO.SYSRESET;
    };
```

To get access to the signals via the CLI interface, the CLI command *signal* can be used. This command allows a user to read the value of the specific signal or set the signal value to *0* or *1*.

```
# clia signal gdiscrete1
Pigeon Point Chassis Manager Command Line Interpreter
GDISCRETE1 == 0 (0x0)
# clia signal gdiscrete1 1
Pigeon Point Chassis Manager Command Line Interpreter
Operation complete
# clia signal gdiscrete1
Pigeon Point Chassis Manager Command Line Interpreter
GDISCRETE1 == 1 (0x1)
```

4 Using the Chassis Manager

This section introduces the overall operation of the Chassis Manager, including operation in a redundant (active/standby) pair.

4.1 In this section

This section contains the topics listed below. Just click on a topic to go to it.

- ChMM login
- Starting the Chassis Manager
- Starting SNMP and HTTP daemons manually
- Redundant operation
- Operation in the chassis with radial
- Automatic SEL truncation
- Cooling state sensors
- Deadlock detection
- I2C bus fault isolation
- Assignment of LAN configuration parameters to boards and modules
- ChMM tests available via the diagnostic infrastructure

4.2 ChMM login

Once the ChMM has been fully booted, you are prompted to login. You can log in as user root; the default password, as set in the factory-provided RFS images, is the empty password “”. We highly recommend that you change passwords during the configuration of the ChMM. If the password is forgotten, it can be reset as described in section 6.4. The following is a typical log-in:

```
chmm700 login: root
Password:
# ls /
bin             dev             etc             lib             mnt
proc            sbin            tmp             usr             var
##
```

4.3 Starting the Chassis Manager

The Chassis Manager software is implemented in the executable file *shelfman* in the directory */bin*. During normal use of the Chassis Manager it is invoked automatically by startup scripts. Typical users never need to invoke it manually as described here. Nevertheless, for the unusual circumstances in which manual invocation may be necessary, the details are described here.

The syntax of a Chassis Manager command line invocation is defined as follows:

shelfman [<options>] &

or

daemon –f shelfman [<options>]

The following options are recognized:

-h <address>

-v <verbosity>

-c <path>

-cs[-]

-w[-]

-wt

-l[c][s]

-g <ip_address>

-sf[-]

-port <port>

-ph[=<IPMB-addr1>,<IPMB-addr2>]

-i <command_line>.

A detailed description of these options is given below.

-h <address>

This option overrides the hardware address of the FRU site where the Chassis Manager resides. This hardware address is used as the IPMB address for the Chassis Manager (the address that is based on its hardware address, separate from 20h) and is treated as a hexadecimal number. This option can be used if no hardware address is automatically available for the Chassis Manager. If this option is not used, the Chassis Manager obtains the hardware address in a carrier-specific way; the IPMB address is the hardware address multiplied by 2.

-v <verbosity>

Set the initial debug verbosity mask (see section 3.3.2), both for the system log and for the console. If this option is not present, the values of the parameters *VERBOSITY* and *VERBOSITY_CONSOLE* in the file */etc/shelfman.conf* determine the initial debug verbosity mask.

-c <path>

Path to non-volatile configuration files. The default path is */var/nvdata*.

-cs[-]

Enforce (*-cs*) / disable (*-cs-*) checking Chassis FRU Information checksums. If a checksum is invalid, an error message is produced and the Chassis FRU Information is not used. If a checksum is valid, no message is produced. If the option *-cs* is not present in the command line, the value of

the parameter *VERIFY_SHELF_FRU_CHECKSUM* from the file */etc/shelfman.conf* determines Chassis Manager behavior in this area.

-w[-]| -wt

Enable (*-w*) /disable (*-w-*) the watchdog timer. If neither of the *–w-* or *-wt* options is present in the command line, the value of the parameter *WATCHDOG_ENABLED* from the file */etc/shelfman.conf* determines Chassis Manager actions in this area.

The *-wt* option puts the watchdog timer in test mode: that is, the actual timer is not turned on or strobed, but the strobing thread in the Chassis Manager still runs and a warning message is printed if the interval between subsequent strobes exceeds 500 ms. (In normal operation, if this interval exceeds two seconds, the Chassis Manager is reset.)

-l[c][s]

Set logging destination: use *s* to choose syslog and *c* to choose console. If this option is not specified, the values of the parameters *CONSOLE_LOGGING_ENABLED* and *SYSLOG_LOGGING_ENABLED* from the file */etc/shelfman.conf* determine Chassis Manager actions in this area.

-g <ip_address>

Set the default gateway IP address. This address is only used if no gateway address is set in the LAN Configuration Parameters for channel 1.

-sf[-]

The option *–sf* forces the Chassis Manager to use EEPROMs for Chassis FRU Information storage. The option *–sf-* forces the Chassis Manager not to use EEPROMs for Chassis FRU Information storage. If neither of these options is present in the command line, the value of the parameter *SHELF_FRU_IN_EEPROM* from the file */etc/shelfman.conf* determines Chassis Manager actions in this area.

-p <port>

This option sets the redundancy communication port. If this option is not present, the value of the parameter *REDUNDANCY_PORT* from the file */etc/shelfman.conf* determines the Chassis Manager actions in this area.

-ph[=<IPMB-addr1>,<IPMB-addr2>]

This option defines IPMB addresses of pseudo-hubs (by default, 0x82 and 0x84). Pseudo-hubs are virtual IPM Controllers that are created and emulated by the Chassis Manager and behave like IPM Controllers for switch modules. This option can be used for testing and when non-intelligent switch modules (non-compliant boards that do not implement an IPM Controller) are installed in the chassis.

-i <command_line>

This option defines the command line for the initialization script that is invoked by the active Chassis Manager during initialization. This initialization script can be used for platform-specific initialization. The HRI bits Local Healthy and Active are set when the command line is invoked and

the Chassis Manager waits for completion of the initialization script before progressing further. Note that the initialization script is run only at the startup of the active Chassis Manager, and is not called after a switchover. The corresponding command line can also be specified as the value of the configuration parameter *INITIALIZATION_SCRIPT* in the Chassis Manager configuration file.

To run the Chassis Manager from the command line, type the following:

daemon –f shelfman [<options>]

For example, here is a typical log of the Chassis Manager starting with options that cause it to route log information both to the console and to the log file and to use EEPROMs for Chassis FRU Information:

```
# daemon -f shelfman -sf- -lcs
Oct 24 17:49:29 chmm~110 daemon.info shelfman: [164]: Pigeon Point Chassis
Manager ver. 1.2.0. Built on Oct 24 2017 08:45:19
Oct 24 17:49:29 chmm~110 daemon.info shelfman: [164]: *** Lock log print buffer
at 0x22b220 ***
Oct 24 17:49:29 chmm~110 daemon.info shelfman: [164]: *** Pthread lock log print
buffer at 0x22fa40 ***
Oct 24 17:49:29 chmm~110 daemon.info shelfman: [164]: Read OTP bits 0x0202B206,
status=0
Oct 24 17:49:29 chmm~110 daemon.info shelfman: [164]: Reading configuration
file: /etc/shelfman.conf
Oct 24 17:49:29 chmm~110 daemon.info shelfman: [164]: No carrier-specific
configuration file found for "PPS700", using default configuration file
Oct 24 17:49:29 chmm~110 daemon.info shelfman: [164]: Shelfman: Using EEPROM for
Shelf FRU
Oct 24 17:49:29 chmm~110 daemon.info shelfman: [164]: Setting watchdog timeout
to 2 seconds returned 0
Oct 24 17:49:29 chmm~110 user.info kernel: [  175.410000] pps_wd: auto-strobbing
stopped
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: No Custom SDR
initialization file found, using defaults
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Carrier set to "PPS700"
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Successfully found a
callback table carrier_hpdl_callback_table at 0x1E9138
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: shm_crypto_init:
initialization succeeded
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: HPDL: Trying to load from
file /var/nvdata/carrier_data
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: HPDL: Trying to load from
file /etc/vendor/pps/vita_btc7_carrier.hpdlbin
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: DecryptData: using PPS AES
encryption
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Encrypted data found:
PPS0AES, 0 extra bytes
```

```
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Data decompression
success: uncompressed size 2240, original size 912
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Device : preserve_state=0,
double_read=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Device SHMM_GPIO:
preserve_state=0, double_read=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Device Multiplexor:
preserve_state=0, double_read=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Device Master_ADT7490:
preserve_state=0, double_read=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Device HardwareAddress:
preserve_state=0, double_read=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Device Temp:
preserve_state=0, double_read=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Device CARRIER_EEPROM:
preserve_state=0, double_read=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Device SHMM_ANALOG:
preserve_state=0, double_read=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: +++ IPMB Topology=0, cross
link=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: ShMM GPIO: Signal name
nBOARD_SEATED translated to 0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: ShMM GPIO: Signal name
HANDLE_CLOSED translated to 1
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: ShMM GPIO: Signal name
SLOT_ID translated to 2
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: ShMM GPIO: Signal name
nIPMB_A_PU_EN translated to 3
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: ShMM GPIO: Signal name
nIPMB_B_PU_EN translated to 4
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: ShMM GPIO: Signal name
nSYSRESET_OUT translated to 5
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: ShMM GPIO: Signal name
BLUE_LED translated to 6
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Opening local bus 0,
device i2c0, status=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Opening local bus 1,
device i2c1, status=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Opening local bus 2,
device i2c2, status=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Opening local bus 3,
device i2c3, status=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Opening local bus 4,
device i2c4, status=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Opening local bus 5,
device i2c5, status=0
```

```
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: HpdlCarrier: Configure
devices, active=0
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: cHpdlCarrierModule
HardwareInitializeCarrier called!
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: HPDL: Hardware address
received: 08
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Device GUID: {8B5F4748-
B8CA-11E7-0080-00184902B206}
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Netmask not specified in
two-adapter redundancy configuration: defaulting to 255.255.255.128
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Interface usb1 activated
successfully
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Input redundancy socket
successfully bound to: 192.168.0.110:1040
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Output redundancy socket
successfully bound to: 192.168.0.238:1041
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Active listening thread
created successfully
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [164]: Connecting to:
192.168.0.239:1040
Oct 24 17:49:31 chmm~110 daemon.info shelfman: [167]: Active listening thread
started
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: *** Running in Active mode
(connection to backup failed: -110) ***
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Redundancy protocol
initialized successfully
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Register
OEM_RUN_SHMM_DIAGNOSTICS: status=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Register
OEM_GET_SHMM_DIAGNOSTICS_RESULTS: status=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Register
OEM_GET_STATIC_DI_DATA: status=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Register
OEM_GET_DI_TEST_READINESS: status=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Register
OEM_GET_DI_START_TEST_RESERVATION_ID: status=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Register
OEM_SPECIFY_DI_TEST_PARAMETERS: status=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Register
OEM_START_DI_TEST: status=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Register
OEM_CANCEL_DI_TEST: status=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Register
OEM_GET_DI_TEST_STATUS: status=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Register
OEM_GET_DI_TEST_RESULTS_RESERVATION_ID: status=0
```

```
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Register
OEM_GET_DI_TEST_RESULTS: status=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: Calling
ipmf_init_open_ipmb
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: ipmf_ipmb_is_active_shmm:
set to 1
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: ipmf_init_open_ipmb:
cross_as_bus_index=0, active=1
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: ipmf_open_ipmb, bus 0,
handle=(nil), cross_as_bus_index=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: ipmf_open_ipmb, bus 1,
handle=(nil), cross_as_bus_index=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]:
channel_param_create_default called for channel 0, skip=0
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: pre: access for IPv6
file=-1
Oct 24 17:49:34 chmm~110 daemon.info shelfman: [164]: LAN : Legacy cipher suites
params storage filled
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: channel_set_param_file:
ch=0, offset=0, len=8, buffer=0x269c30
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: post: access for IPv6
file=-1
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]:
channel_param_create_default called for channel 1, skip=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: pre: access for IPv6
file=-1
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: channel_set_param_file:
ch=1, offset=0, len=408, buffer=0x269c40
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: post: access for IPv6
file=-1
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: channel_set_param_file:
ch=1, offset=441, len=459, buffer=0x269df9
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: channel_set_param_file:
write /var/nvdata/ch1_param_ipv6_v2; offset=441, len=459, recorded_len=408,
recorded_len2=441
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: channel_set_param_file:
write /var/nvdata/ch1_param_ipv6_v2; file_offset=0, wr_len=459, buf_offset=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]:
channel_param_create_default called for channel 2, skip=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: pre: access for IPv6
file=-1
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: channel_set_param_file:
ch=2, offset=0, len=408, buffer=0x269fc8
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: post: access for IPv6
file=-1
```

```
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: channel_set_param_file:
ch=2, offset=441, len=459, buffer=0x26a181
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: channel_set_param_file:
write /var/nvdata/ch2_param_ipv6_v2; offset=441, len=459, recorded_len=408,
recorded_len2=441
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: channel_set_param_file:
write /var/nvdata/ch2_param_ipv6_v2; file_offset=0, wr_len=459, buf_offset=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: HPM.3 ipmf_register_hpm3
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: HPM.3 registered
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: ipmf_open_ipmb, bus 0,
handle=0x269948, cross_as_bus_index=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: ipmf_open_ipmb, bus 1,
handle=0x269c20, cross_as_bus_index=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: ipmf_open_ipmb, bus 0,
handle=0x269948, cross_as_bus_index=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: ipmf_open_ipmb, bus 1,
handle=0x269c20, cross_as_bus_index=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Called
HpdlCarrierModuleHsHandleOpen
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: HPM.1: command handlers
are registered
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]:
reboot_reason_get_sensor_reading: reading: 00, asserted states: 0000
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]:
shm_fetch_reboot_reason(value=0) = 10
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Reboot reason sensor (129)
was registered, status=0, value=10
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]:
reboot_reason_get_sensor_reading: reading: 00, asserted states: 0400
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]:
reboot_reason_get_sensor_event_data: 0a, 00, 00
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Current bank sensor
registration successful
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Failed upgrade sensor
registration successful
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: HPDL: Trying to load from
file /var/nvdata/sdr_pool
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: HPDL: Trying to load from
file /var/nvdata/carrier_sdrs
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: HPDL: Trying to load from
file /etc/vendor/pps/vita_btc7_carrier.sdrs
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: DecryptData: using PPS AES
encryption
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Encrypted data found:
PPS0AES, 2 extra bytes
```

```
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Data decompression
success: uncompressed size 973, original size 398
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Max FRU device ID for 10h
= 0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]:
HpdlFru::RegisterControllerCallbacks: SA 10: updating name in SDR 0x26a208 to
Carrier_FRU
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]:
HpdlFru::RegisterControllerCallbacks need_ext_inactive=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Register LED
infrastructure, FRU 0: status = 0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 12 (this=0x2652e8), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
12 (this=0x2652e8): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
12 (this=0x2652e8): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,12)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 13 (this=0x265640), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
13 (this=0x265640): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
13 (this=0x265640): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,13)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 14 (this=0x265a10), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
14 (this=0x265a10): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
14 (this=0x265a10): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,14)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 15 (this=0x265dc8), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
15 (this=0x265dc8): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
15 (this=0x265dc8): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,15)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 16 (this=0x266170), pool=0x26b390, len=973
```

```
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
16 (this=0x266170): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
16 (this=0x266170): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,16)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 17 (this=0x266568), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
17 (this=0x266568): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
17 (this=0x266568): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,17)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 18 (this=0x266918), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
18 (this=0x266918): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
18 (this=0x266918): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,18)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 19 (this=0x266cf0), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
19 (this=0x266cf0): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
19 (this=0x266cf0): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,19)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 20 (this=0x2670c8), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
20 (this=0x2670c8): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
20 (this=0x2670c8): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,20)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 21 (this=0x2673f8), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
21 (this=0x2673f8): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
21 (this=0x2673f8): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,21)
registered, initial_update=0
```

```
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 22 (this=0x2677a0), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
22 (this=0x2677a0): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
22 (this=0x2677a0): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,22)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 23 (this=0x267b48), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
23 (this=0x267b48): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
23 (this=0x267b48): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,23)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 24 (this=0x267ef0), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
24 (this=0x267ef0): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
24 (this=0x267ef0): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,24)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 25 (this=0x268298), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
25 (this=0x268298): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
25 (this=0x268298): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,25)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 26 (this=0x268640), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
26 (this=0x268640): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
26 (this=0x268640): completed
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,26)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Setting up SDR for sensor:
10, 27 (this=0x2689e8), pool=0x26b390, len=973
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
27 (this=0x2689e8): FindSdr succeeded
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SetupSDR for sensor: 10,
27 (this=0x2689e8): completed
```

```
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Sensor (10,0,27)
registered, initial_update=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]:
HpdlFru::RegisterControllerCallbacks: fru=0x264ed8, exit, ok=1
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: ActivateHpdlPreloaded,
m_carrier_configured=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: HpdlCarrier: Configure
devices, active=1
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: ActivateHpdlPreloaded,
HPDL preloaded length=0, SDR preloaded length=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Registered
HpdlCarrierCmdMasterWriteRead
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Activating radial state:
maximum 96 links
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Activating radial states:
96 links
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Carrier version set to
01:20:1C:00
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: SEL: journaling ON,
compression ON
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: ChM SEL Activation
complete
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [169]: SEL truncation thread
started successfully
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [168]: Started SEL dedicated
write thread
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [168]: Creating new SEL
Journal...
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: PEF: Cannot open last
counters file /var/nvdata/pef.last, errno=2 (No such file or directory), using
defaults
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: PEF: External event
handler is not defined, PEF external handler thread will not start
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: PEF activated successfully
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: PEF initialized
successfully, System Event Sensor #133
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Chassis facility activated
successfully
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Chassis facility
initialized successfully
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Registering shelf FRU
notification: 0x40104c08
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [171]: PEF: starting PEF thread
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: channel_get_param_file:
ch=0, offset=0, len=8, buffer=0x28d3b0
```

```
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: channel_get_param_file:
ch=1, offset=0, len=408, buffer=0x28d3c0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: channel_get_param_file:
ch=2, offset=0, len=408, buffer=0x28d748
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: user_access_db: op 2,
encryption is supported; database is encrypted, offset=0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Session: session
descriptor size = 1120 bytes
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Session: session 3.4
legacy descriptor size = 996 bytes
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: dhcpv6_client_init called
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Msg: channel #0 is created
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Msg: channel #0 has
parameters
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Enabling channel 0
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Msg: channel #1 is created
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Msg: channel #1 has
parameters
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Enabling channel 1
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Msg: channel #2 is created
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Msg: channel #2 has
parameters
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: Enabling channel 2
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [164]: RMCPP : Legacy cipher
suites params storage filled
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [176]: Started local message
thread
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [178]: Started incoming message
thread, hc=1
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [178]: IPMB 0 attached
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [177]: Started local message
thread
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [178]: IPMB 1 attached
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [178]: Status change on IPMB 0:
flags = 183 (ONLINE,CONNECTED,ONL_CH,CON_CH)
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [178]: IPMB 0 went online
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [178]: Status change on IPMB 1:
flags = 183 (ONLINE,CONNECTED,ONL_CH,CON_CH)
Oct 24 17:49:35 chmm~110 daemon.info shelfman: [178]: IPMB 1 went online
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [179]: IPMC Stored write thread
started
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [175]: InitializeFRU for SA 0x10,
FRU 0: found=0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [175]: Controller 10, FRU 0:
State set to M1, prev=M0, cause=0, locked=0
```

```
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [175]: Sending FRU state, 10, 0:
last_open=0, state=1, prev_state=0, cause=0, locked=0, deact_locked=0,
activation_index=0

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [175]: Controller 10, FRU 0:
State set to M4, prev=M1, cause=0, locked=0

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [175]: Sending FRU state, 10, 0:
last_open=0, state=4, prev_state=1, cause=0, locked=0, deact_locked=0,
activation_index=1

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [175]:
ipmf_poll_threshold_sensor_with_hysteresis: assertion event at offset 2:
reading=-128, threshold=-128

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]:
shm_create_hpi_sys_event_sensor: sensor registered successfully, lun 3, number 0

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]:
session_create_session_audit_sensor: shm_cfg_disable_session_audit_events=0

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Starting power thread

Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [176]: IPMF: unknown command:
netfn 0x2c, command 0x00

Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [176]: IPMF: unknown command:
netfn 0x2c, command 0x00

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: sdrrep_full_ipmc_create:
called for 10

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: sdrrep_full_ipmc_create:
created new structure for 10

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: sdrrep_full_ipmc_create:
called for 20

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: sdrrep_full_ipmc_create:
created new structure for 20

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: SDR Repository:
broadcasting Get Device ID

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [189]: CLI: subscribing for
deadlocks, pid=189

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [189]: Registering facility 189
to the triggering watchdog

Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [182]: Tasklet GET_INFO for SA
0x20: Get PICMG Properties completion code = 0xc1

Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [182]: Tasklet GET_INFO for SA
0x20: Get PICMG Properties not supported on this IPMC

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [182]: Sending Get VSO
Capabilities to SA 0x20

Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [184]: Tasklet GET_INFO for SA
0x10: Get PICMG Properties completion code = 0xc1

Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [184]: Tasklet GET_INFO for SA
0x10: Get PICMG Properties not supported on this IPMC

Oct 24 17:49:36 chmm~110 daemon.info shelfman: [184]: Sending Get VSO
Capabilities to SA 0x10
```

```
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [184]: Tasklet GET_INFO for SA
0x10: Get PICMG Properties reported max FRU = 0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [186]: Initial indicator for SA
0x10: 28f5bcc (state new)
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [186]: Delete SDRs on ipmc state
NEW: SA=10
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [186]: Initial indicator for SA
0x20: 28f5bc7 (state new)
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [186]: Delete SDRs on ipmc state
NEW: SA=20
Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [176]: IPMF: unknown command:
netfn 0x2c, command 0x3e
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [186]: sdrrep: fetched new SDRs,
time=7
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [190]:
sdrrep_full_ipmc_sdrs_update_thread for 10
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [190]:
sdrrep_full_ipmc_sdrs_update_thread for 10: attempt 0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [191]:
sdrrep_full_ipmc_sdrs_update_thread for 20
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]:
shm_create_shelffru_redundancy_sensor: Shelf FRU redundancy sensor (3,1) created
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: shelf_fru_activate:
shm_create_shelffru_redundancy_sensor status=0: created
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Generating list of the
trusted Chassis FRU sources
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [192]: FRU data write thread
started successfully
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [191]:
sdrrep_full_ipmc_sdrs_update_thread for 20: attempt 0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [182]: Tasklet GET_INFO for SA
0x20: Get PICMG Properties reported max FRU = 32
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: FRU 1 on IPMC 0x20 was
added to the Shelf FRU trusted list
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Shmm Upgrade: Starting...
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Shmm Upgrade: Detecting
current bank.
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [195]: Shelf FRU: registering for
SDR Repository notifications!
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [195]: ***
shelf_fru_retrieve_fru_info: record ID=1(0001) type=0x12, op = 1 ***
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [184]: Channel mask=0x0, LAN
Params: start=0(0x0), rev=0; SOL Params: start=0(0x0), rev=0
Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [176]: IPMF: unknown command:
netfn 0x2c, command 0x3e
```

```
Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [176]: IPMF: unknown command:
netfn 0x2c, command 0x3e
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [177]: Hot Swap event: SA=0x10,
FRU 0, sensor_number = 0x00, M0->M1, cause 0, ignore=0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [177]: Hot Swap event ignored,
init incomplete: SA 0x10, FRU 0, event: 0->1
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [177]: Node busy returned,
message from 10 ignored, seq_no=9
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [182]: Channel mask=0x0, LAN
Params: start=0(0x0), rev=0; SOL Params: start=0(0x0), rev=0
Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [184]: Tasklet GET_INFO for SA
0x10: Get HPM.3 Capabilities not supported on this IPMC (cc=C1)
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [184]: Tasklet GET_INFO for SA
0x10: Get HPM.3 Capabilities reported rc=193, revision=0, LAN Params:
start=0(0x0), rev=0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [184]: Tasklet GET_INFO for SA
0x10: allocating 288 bytes, 1 FRUs, 288 bytes/FRU
Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [176]: IPMF: unknown command:
netfn 0x2c, command 0x3e
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [187]: ***
shelf_fru_retrieve_fru_info: record ID=1(0001) type=0x12, op = 1 ***
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [187]: ***
shelf_fru_retrieve_fru_info: record ID=1(0001) type=0x12, op = 4 ***
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [184]: sdrrep_full_ipmc_create:
called for 10
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [184]: sdrrep_full_ipmc_create:
found existing structure for 10
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [184]: Tasklet GET_INFO for SA
0x10: send Set Event Receiver (send_on_M7_exit=0)
Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [182]: Tasklet GET_INFO for SA
0x20: Get HPM.3 Capabilities not supported on this IPMC (cc=C1)
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [182]: Tasklet GET_INFO for SA
0x20: Get HPM.3 Capabilities reported rc=193, revision=0, LAN Params:
start=0(0x0), rev=0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [182]: Setting max FRU to 252 for
the BMC
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [182]: Tasklet GET_INFO for SA
0x20: allocating 72864 bytes, 253 FRUs, 288 bytes/FRU
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Shmm Upgrade: Current bank
id is 0.
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Shmm Upgrade: Restoring
upgrade status...
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Shmm Upgrade: Restored
status: action NONE, state DONE, pending ABSENT
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Shmm Upgrade: Current
status: action NONE, state DONE, pending ABSENT
```

```
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Shmm Upgrade: Processing
rupgrade status...
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Shmm Upgrade: Current
status: action NONE, state DONE, pending ABSENT
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [177]: Hot Swap event: SA=0x10,
FRU 0, sensor_number = 0x00, M0->M1, cause 0, ignore=0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [177]: State M1, SA 0x10, FRU 0,
op_state=M0, power_cycle=0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [186]: SDR Repository: dynamic
SDRs population has changed for SA 0x20, ind=28f5bc8
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [186]: Initial indicator for SA
0x20: 28f5bc8
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [186]: sdrrep: fetched new SDRs,
time=7
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: set_failed_upgrade_flag:
Status: ff00ff00
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: ***
ipmf_set_failed_upgrade_sensor called, upgrade_failed=0!
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [199]:
sdrrep_full_ipmc_sdrs_update_thread for 20
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [199]:
sdrrep_full_ipmc_sdrs_update_thread for 20: attempt 0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [177]: Hot Swap event: SA=0x10,
FRU 0, sensor_number = 0x00, M1->M4, cause 0, ignore=0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [204]: SA 0x10, FRU 0 is
ACTIVATING
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [204]: Tasklet ACTIVATE (10, 0):
is_local_address = 1
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [204]: Tasklet ACTIVATE (10, 0):
postponed until Shelf FRU Information is found
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [177]: Hot Swap event: SA=0x10,
FRU 0, sensor_number = 0x00, M1->M4, cause 0, ignore=0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [177]: Ignored Hot Swap event:
from 10, seq_no=12
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [205]: SA 0x10, FRU 0 is
ACTIVATING  (Postponed)
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [205]: Tasklet ACTIVATE (10, 0):
is_local_address = 1
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [205]: Tasklet ACTIVATE (10, 0):
postponed until Shelf FRU Information is found
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Controller 20, FRU 0:
State set to M1, prev=M0, cause=0, locked=0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Before init_fru_hs_state
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: set the FRU 1 initial
state
```

```
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Controller 20, FRU 1:
State set to M1, prev=M0, cause=0, locked=0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Shelfman: shelf_fru_init()
passed
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Registering shelf FRU
notification: 0xbfd44
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Registering shelf FRU
notification: 0x4007e1b4
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Carrier Activate() called,
m_carrier_configured=1
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Registering HPDL Shelf FRU
Notification: 0xa655c
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]: Registering shelf FRU
notification: 0xa655c
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]:
build_shmgr_fru_information: Time: mfg=1508823919, start=820443600
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [164]:
build_shmgr_fru_information: Final offset = 144
Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [200]: Tasklet GET_INFO for SA
0xa2: Get PICMG Properties completion code = 0xc1
Oct 24 17:49:36 chmm~110 daemon.warn shelfman: [200]: Tasklet GET_INFO for SA
0xa2: Get PICMG Properties not supported on this IPMC
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [200]: Sending Get VSO
Capabilities to SA 0xa2
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [200]: Tasklet GET_INFO for SA
0xa2: Get PICMG Properties reported max FRU = 1
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [177]: Hot Swap event: SA=0x20,
FRU 0, sensor_number = 0x00, M0->M1, cause 0, ignore=0
Oct 24 17:49:36 chmm~110 daemon.info shelfman: [177]: State M1, SA 0x20, FRU 0,
op_state=M0, power_cycle=0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [177]: Hot Swap event: SA=0x20,
FRU 1, sensor_number = 0x07, M0->M1, cause 0, ignore=0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [177]: State M1, SA 0x20, FRU 1,
op_state=M0, power_cycle=0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [200]: Channel mask=0x20, LAN
Params: start=192(0xc0), rev=1; SOL Params: start=192(0xc0), rev=1
Oct 24 17:49:37 chmm~110 daemon.warn shelfman: [200]: Tasklet GET_INFO for SA
0xa2: Get HPM.3 Capabilities not supported on this IPMC (cc=CC)
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [200]: Tasklet GET_INFO for SA
0xa2: Get HPM.3 Capabilities reported rc=204, revision=0, LAN Params:
start=0(0x0), rev=0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [200]: Tasklet GET_INFO for SA
0xA2: allocating 576 bytes, 2 FRUs, 288 bytes/FRU
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [200]: sdrrep_full_ipmc_create:
called for A2
```

```
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [200]: sdrrep_full_ipmc_create:
created new structure for A2
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [187]: ***
shelf_fru_retrieve_fru_info: record ID=2(0002) type=0x12, op = 1 ***
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [186]: SDR Repository: dynamic
SDRs population has changed for SA 0x20, ind=28f5bc9
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [186]: Initial indicator for SA
0x20: 28f5bc9
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [186]: Initial indicator for SA
0xa2: 0 (state new)
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [186]: Delete SDRs on ipmc state
NEW: SA=A2
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [186]: sdrrep: fetched new SDRs,
time=7
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [211]:
sdrrep_full_ipmc_sdrs_update_thread for A2
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [211]:
sdrrep_full_ipmc_sdrs_update_thread for A2: attempt 0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [209]:
sdrrep_full_ipmc_sdrs_update_thread for 20
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [209]:
sdrrep_full_ipmc_sdrs_update_thread for 20: attempt 0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [210]: Generating "on behalf"
hotswap event for 0xA2, state 1
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [210]: Hot Swap event: SA=0xa2,
FRU 0, sensor_number = 0x00, M0->M1, cause 0, ignore=0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [210]: State M1, SA 0xA2, FRU 0,
op_state=M0, power_cycle=0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [200]: Tasklet GET_INFO for SA
0xa2: send Set Event Receiver (send_on_M7_exit=0)
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [187]: ***
shelf_fru_retrieve_fru_info: record ID=3(0003) type=0x11, op = 1 ***
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [175]: Hot Swap event: SA=0xa2,
FRU 0, sensor_number = 0x00, M0->M1, cause 0, ignore=0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [175]: State M1, SA 0xA2, FRU 0,
op_state=M1, power_cycle=0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [175]: Ignored Hot Swap event:
from a2, seq_no=35
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [175]: Controller 20, FRU 0:
State set to M4, prev=M1, cause=0, locked=0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [177]: Hot Swap event: SA=0x20,
FRU 0, sensor_number = 0x00, M1->M4, cause 0, ignore=0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [175]: Controller 20, FRU 1:
State set to M4, prev=M1, cause=0, locked=0
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [216]: SA 0x20, FRU 0 is
ACTIVATING
```

```
Oct 24 17:49:37 chmm~110 daemon.info shelfman: [175]: Hot Swap event: SA=0xa2,
FRU 1, sensor_number = 0x15, M0->M1, cause 0, ignore=0

Oct 24 17:49:37 chmm~110 daemon.info shelfman: [175]: State M1, SA 0xA2, FRU 1,
op_state=M0, power_cycle=0

Oct 24 17:49:37 chmm~110 daemon.info shelfman: [177]: Hot Swap event: SA=0x20,
FRU 1, sensor_number = 0x07, M1->M4, cause 0, ignore=0

Oct 24 17:49:37 chmm~110 daemon.info shelfman: [164]:
build_shmm_fru_information: Time: mfg=1388696400, start=820443600

Oct 24 17:49:37 chmm~110 daemon.info shelfman: [164]:
build_shmm_fru_information: Final offset = 176

Oct 24 17:49:37 chmm~110 daemon.info shelfman: [217]: SA 0x20, FRU 1 is
ACTIVATING

Oct 24 17:49:37 chmm~110 daemon.info shelfman: [164]: Chassis Manager:
ChMM/Chassis Manager FRU Information created

Oct 24 17:49:37 chmm~110 daemon.info shelfman: [164]: Shelfman: Running

Oct 24 17:49:37 chmm~110 daemon.info shelfman: [182]: sdrrep_full_ipmc_create:
called for 20

Oct 24 17:49:37 chmm~110 daemon.info shelfman: [182]: sdrrep_full_ipmc_create:
found existing structure for 20

Oct 24 17:49:37 chmm~110 daemon.info shelfman: [187]: ***
shelf_fru_retrieve_fru_info: record ID=4(0004) type=0x12, op = 1 ***

Oct 24 17:49:37 chmm~110 daemon.info shelfman: [186]: sdrrep: Skipping SA 0xA2,
read already in progress

Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: ***
shelf_fru_retrieve_fru_info: record ID=5(0005) type=0x11, op = 1 ***

Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
cHpdlCarrier::GetShelfFruInventoryAreaInfo called

Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: *** FRU Inventory Area
Info [20#1], size = 32768, access = 0, status = 0

Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 2: Get Device Locator Record ID completion code = 0xd3

Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 2: Get Device Locator Record ID completion code = 0xd3

Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 2: Get Device Locator Record ID completion code = 0xd3

Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 2: 3 attempts to read Type 11h/12h SDR failed

Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 3: Get Device Locator Record ID completion code = 0xd3

Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 3: Get Device Locator Record ID completion code = 0xd3

Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 3: Get Device Locator Record ID completion code = 0xd3

Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 3: 3 attempts to read Type 11h/12h SDR failed
```

```
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 4: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 4: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 4: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 4: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 5: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Added IPMC 0x20, FRU 1 to
the FRU list, treated as Shelf FRU Info storage, size = 32768 (data size = 624)
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: 1 Equal Shelf FRUs have
been detected
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: !!! Allocating and
populating shelf_fru_info_global, count=1
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: !!! Setting
shelf_fru_info_found_flag = 1
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
shelffru_redundancy_set_new_state: setting state for Shelf FRU redundancy to 2,
additional=0
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Calling shelf FRU
notifications: count=1, op=1
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Mark shelf FRU
notification for call: 0x40104c08
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Mark shelf FRU
notification for call: 0xbfd44
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Mark shelf FRU
notification for call: 0x4007e1b4
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Mark shelf FRU
notification for call: 0xa655c
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Calling shelf FRU
notification (marked): 0x40104c08
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 5: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Calling shelf FRU
notification (marked): 0xbfd44
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 5: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 5: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 6: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Lan apply cfg parameters:
HW addr 0x10, cross 0, use_second 0, adapter eth0 (), adapter2 eth1 ()
```

```
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Lan apply cfg parameters:
IP 7501A8C0, GW FD01A8C0, MASK 00FFFFFF
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_set_param_file:
ch=1, offset=12, len=1, buffer=0x28d3cc
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_set_param_file:
ch=1, offset=8, len=4, buffer=0x28d3c8
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 6: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 6: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 6: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [175]:
shelffru_redundancy_get_sensor_event_data: 31, FF, 00
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_set_param_file:
ch=1, offset=32, len=4, buffer=0x28d3e0
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 7: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_set_param_file:
ch=1, offset=19, len=4, buffer=0x28d3d3
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 7: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_get_param_file:
ch=1, offset=8, len=892, buffer=0x28d3c8
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_get_param_file:
read /var/nvdata/ch1_param_ipv6; offset=8, len=892, recorded_len=408,
recorded_len2=441
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_get_param_file:
read /var/nvdata/ch1_param_ipv6; file_offset=0, rd_len=33, buf_offset=400
Oct 24 17:49:38 chmm~110 daemon.err shelfman: [187]: channel_get_param_file:
open(/var/nvdata/ch1_param_ipv6) failed (2)
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_get_param_file:
read /var/nvdata/ch1_param_ipv6_v2; offset=8, len=892, recorded_len=408,
recorded_len2=441
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 7: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_get_param_file:
read /var/nvdata/ch1_param_ipv6_v2; file_offset=0, rd_len=459, buf_offset=433
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: LAN: channel 1 address
192.168.1.117:623
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 7: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: LAN: setting RMCP IP
address to 192.168.1.117 on eth0
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: lan_channel_start: setting
flags 1043 to eth0
```

```
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: LAN channel 1: set
flags=0x1043 on 'eth0'
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: LAN: setting RMCP subnet
mask to 255.255.255.0
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: keep up on eth0 returns 0
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: lo, family=17, flags=10049
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: bond0, family=17, flags=1402
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: eth0, family=17, flags=11043
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: eth1, family=17, flags=11043
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: usb0, family=17, flags=11043
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: usb1, family=17, flags=11043
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: lo, family=2, flags=10049
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: eth0, family=2, flags=11043
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: eth1, family=2, flags=11043
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: usb0, family=2, flags=11043
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: usb1, family=2, flags=11043
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: lo, family=10, flags=10049
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: eth0, family=10, flags=11043
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: eth1, family=10, flags=11043
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: usb0, family=10, flags=11043
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
dhcp_restore_alias_interfaces: usb1, family=10, flags=11043
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_set_param_file:
ch=1, offset=8, len=892, buffer=0x28d3c8
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 8: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_set_param_file:
write /var/nvdata/ch1_param_ipv6; offset=8, len=892, recorded_len=408,
recorded_len2=441
```

```
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_set_param_file:
write /var/nvdata/ch1_param_ipv6; file_offset=0, wr_len=33, buf_offset=400
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_set_param_file:
write /var/nvdata/ch1_param_ipv6_v2; offset=8, len=892, recorded_len=408,
recorded_len2=441
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: channel_set_param_file:
write /var/nvdata/ch1_param_ipv6_v2; file_offset=0, wr_len=459, buf_offset=433
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: rmcp_channel_start:
cd=0x221368, head=0x221368, head->next=(nil), cd->next=(nil), primary=00,
secondary=00, head->p=00, head->s=00
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [221]: RMCP: starting server
thread for 192.168.1.117:623
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [221]: RMCP: started server
thread for 192.168.1.117:623 (0x221368), sockfd=29, sockfd6=30, terminate=0
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [221]: Registering facility 221
to the triggering watchdog
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
channel_param_start_channel_on_backup: channel 1, propagate 0
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Calling shelf FRU
notification (marked): 0x4007e1b4
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
ipmc_shelffru_found_callback: quantity = 1
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: LAN Config Parameters data
in the Shelf FRU: 0 bytes
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I002 : ipmb A support=1h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I002 : ipmb B support=1h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I002 : ipmb A max freq=1h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I002 : ipmb B max freq=0h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I001 Support for IPMB-B
successfully determined as 1h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : VITA 46.11 address
table contents, total 13 records
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [211]: sdrrep: finished reading
SDRs for SA 0xA2: added 26, preserved 0, deleted 0, time=9 seconds (1726 msec in
thread, 1 reserves, 123 msgs: 1664 msec)
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #000  addr=8h
site=1h  type3h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #001  addr=9h
site=2h  type3h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #002  addr=10h
site=1h  typeC0h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #003  addr=10h
site=1h  type2h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #004  addr=10h
site=1h  type4h
```

```
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #005  addr=10h
site=2h  type4h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #006  addr=41h
site=1h  type0h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #007  addr=42h
site=2h  type0h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #008  addr=43h
site=3h  type0h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #009  addr=44h
site=4h  type0h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #010  addr=45h
site=5h  type0h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #011  addr=46h
site=6h  type0h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I003 : #012  addr=51h
site=7h  type0h
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: I004 : Address table found
and parsed
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]:
ipmc_shelffru_found_callback: before invoking Zoned cooling
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 2, Site Type = 0, Site Number: 1
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 2, Site Type = 0, Site Number: 3
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 2, Site Type = 0, Site Number: 5
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 2, Site Type = 0, Site Number: 7
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 2, Site Type = 0, Site Number: 9
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 2, Site Type = 0, Site Number: 11
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 2, Site Type = 0, Site Number: 13
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 2, Site Type = 0, Site Number: 15
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 3, Site Type = 0, Site Number: 2
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 3, Site Type = 0, Site Number: 4
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 3, Site Type = 0, Site Number: 6
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 3, Site Type = 0, Site Number: 8
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-
to-FRU mapping: IPMB = 0x20, FRU Id = 3, Site Type = 0, Site Number: 10
```

```
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-to-FRU mapping: IPMB = 0x20, FRU Id = 3, Site Type = 0, Site Number: 12
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-to-FRU mapping: IPMB = 0x20, FRU Id = 3, Site Type = 0, Site Number: 14
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-to-FRU mapping: IPMB = 0x20, FRU Id = 3, Site Type = 0, Site Number: 16
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-to-FRU mapping: IPMB = 0x20, FRU Id = 3, Site Type = 3, Site Number: 1
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Cooling: Registered Fan-to-FRU mapping: IPMB = 0x20, FRU Id = 3, Site Type = 3, Site Number: 2
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: ipmc_shelffru_found_callback: before loading HPM.3
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [187]: Shelf FRU Info found: failed to get HPM.3 DHCP Configuration record (inst=0) from Shelf FRU (res = -61)
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Found Power Distribution record: instance=0, size=49
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Allowance for FRU Activation Readiness = 10 seconds, start=42949554
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Found Power Management record, format version=0
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [216]: sdrrep_full_ipmc_create: called for 20
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [216]: sdrrep_full_ipmc_create: found existing structure for 20
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [187]: Retrieving initial power budget
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [216]: Tasklet ACTIVATE (20, 0): is_local_address = 1
Oct 24 17:49:38 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO for SA 0x20, FRU 8: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [217]: Tasklet ACTIVATE (20, 1): is_local_address = 1
Oct 24 17:49:38 chmm~110 daemon.info shelfman: [217]: Tasklet ACTIVATE (20, 1): postponed until Shelf FRU Information is found
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO for SA 0x20, FRU 8: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO for SA 0x20, FRU 8: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO for SA 0x20, FRU 9: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO for SA 0x20, FRU 9: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO for SA 0x20, FRU 9: Get Device Locator Record ID completion code = 0xd3
```

```
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 9: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 10: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 10: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]:
[ipmc_get_shelffru_ekey_info] Entered
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: Reading AXIe Board P2P
info from Shelf FRU: res=0
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [180]: Cooling: Default cooling
thread has been terminated...
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 10: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 10: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]:
ipmc_shelffru_found_callback: before register SDR callback for FRUs and sensors:
list unlocked
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]:
ipmc_shelffru_found_callback: before register SDR callback for FRUs and sensors:
list locked
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]:
ipmc_shelffru_found_callback: before starting AXIE power management: 0
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [226]: Zoned Cooling query thread
started successfully
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]:
ipmc_shelffru_found_callback: before start activation
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41514c88
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41514d08
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41514d88
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41514e08
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41514e88
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41514c88 to the internal event queue
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41514d08 to the internal event queue
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41514d88 to the internal event queue
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41514e08 to the internal event queue
```

```
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41514e88 to the internal event queue
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 4 on SA 0x10
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: Cooling update sensor
reading: state changed from UNKNOWN to NORMAL for sensor { 0x10, 4, 0: reading
00 C0 00; reading2 00 00 00 }
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41514c88
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 12 on SA 0x10
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41514c88
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41514f08
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41514c88 to the internal event queue
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41514f08 to the internal event queue
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [228]: CONFIGURING 0x20 # 0
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]:
ipmc_shelffru_found_callback: after start deactivation
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 11: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]:
ipmc_shelffru_found_callback: after ipmc_list_unlock
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [227]: SA 0x10, FRU 0 is
ACTIVATING  (Postponed)
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [227]: Tasklet ACTIVATE (10, 0):
is_local_address = 1
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [227]: +ipmc_zoned_cooling:
allocated an internal event 0x4260a028
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [227]: +ipmc_zoned_cooling:
allocated an internal event 0x4260a0a8
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [227]: +ipmc_zoned_cooling:
allocated an internal event 0x4260a128
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [227]: +ipmc_zoned_cooling:
allocated an internal event 0x4260a1a8
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [227]: +ipmc_zoned_cooling:
allocated an internal event 0x4260a228
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [227]: +ipmc_zoned_cooling: moved
event 0x4260a028 to the internal event queue
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [227]: +ipmc_zoned_cooling: moved
event 0x4260a0a8 to the internal event queue
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [227]: +ipmc_zoned_cooling: moved
event 0x4260a128 to the internal event queue
```

```
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [227]: +ipmc_zoned_cooling: moved
event 0x4260a1a8 to the internal event queue
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: Calling shelf FRU
notification (marked): 0xa655c
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: HPDL-Carrier: Shelf FRU
found
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: HPDL Chassis data in the
Shelf FRU: 0 bytes
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: HPDL: Trying to load from
file /var/nvdata/chassis_data
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: HPDL: Trying to load from
file /etc/vendor/pps/pps700_chassis.hpdlbin
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [227]: +ipmc_zoned_cooling: moved
event 0x4260a228 to the internal event queue
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [229]: SA 0x20, FRU 1 is
ACTIVATING  (Postponed)
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [229]: Tasklet ACTIVATE (20, 1):
is_local_address = 1
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [229]: Tasklet ACTIVATE:
postponed until special order
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [181]: Power not applied  SA
0x20, FRU 0: this is a VSO FRU
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [230]: CONFIGURING 0x10 # 0
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [181]: Power not applied  SA
0x10, FRU 0: this is a VSO FRU
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: DecryptData: using PPS AES
encryption
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: Encrypted data found:
PPS0AES, 13 extra bytes
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: Cooling update sensor
reading: state changed from UNKNOWN to NORMAL for sensor { 0x10, 12, 0: reading
17 C0 C0; reading2 00 00 00 }
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41514d08
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 20 on SA 0x10
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 11: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: Data decompression
success: uncompressed size 1427, original size 531
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: HPDL: Trying to load from
file /var/nvdata/chassis_sdrs
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: HPDL: Trying to load from
file /etc/vendor/pps/pps700_chassis.sdrs
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: DecryptData: using PPS AES
encryption
```

```
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: Encrypted data found:
PPS0AES, 8 extra bytes
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: Data decompression
success: uncompressed size 118, original size 88
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: Device ShelfFRU1_Device:
preserve_state=0, double_read=0
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: Cooling update sensor
reading: state changed from UNKNOWN to NORMAL for sensor { 0x10, 20, 0: reading
14 C0 C0; reading2 00 00 00 }
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41514d88
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 21 on SA 0x10
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: Cooling update sensor
reading: state changed from UNKNOWN to NORMAL for sensor { 0x10, 21, 0: reading
13 C0 C0; reading2 00 00 00 }
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41514e08
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 22 on SA 0x10
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 11: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 11: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 12: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:39 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 12: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: Cooling update sensor
reading: state changed from UNKNOWN to NORMAL for sensor { 0x10, 22, 0: reading
80 C0 C2; reading2 00 00 00 }
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: HPDL-Carrier: ignore_hw=0
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: cHpdlChassis::Init: Max
FRU device ID for 20h = 2
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]:
HpdlFru::RegisterControllerCallbacks: SA 20 FRU 1: updating name in SDR 0x26aa50
to ShelfFRU1
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41514e88
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 4 on SA 0x20
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [225]: Cooling update sensor
reading: state changed from UNKNOWN to NORMAL for sensor { 0x20, 4, 0: reading
00 C0 00; reading2 00 00 00 }
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: UnRegister LED
infrastructure, FRU 1: status = 0
```

```
Oct 24 17:49:39 chmm~110 daemon.info shelfman: [187]: Register LED
infrastructure, FRU 1: status = 0
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: InitializeFRU for SA 0x20,
FRU 1: found=0 (in RegisterControllerCallbacks)
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]:
HpdlFru::RegisterControllerCallbacks: fru=0x415153f8, exit, ok=1
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: Register FRU FanTray1 on
controller 20, FRU 2
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: Setting up SDR for sensor:
20, 240 (this=0x41516490), pool=0x41514d08, len=118
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: SetupSDR for sensor: 20,
240 (this=0x41516490): FindSdr succeeded
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: SetupSDR for sensor: 20,
240 (this=0x41516490): completed
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: Setting up SDR for sensor:
20, 241 (this=0x41516818), pool=0x41514d08, len=118
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: SetupSDR for sensor: 20,
241 (this=0x41516818): FindSdr succeeded
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: SetupSDR for sensor: 20,
241 (this=0x41516818): completed
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41514c88
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 10 on SA 0x20
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 12: Get Device Locator Record ID completion code = 0xd3
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 12: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 13: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 13: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: Cooling update sensor
reading: state changed from UNKNOWN to NORMAL for sensor { 0x20, 10, 0: reading
00 C0 00; reading2 00 00 00 }
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41514f08
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 4 on SA 0x10
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: InitializeFRU for SA 0x20,
FRU 2: found=0
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 13: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 13: 3 attempts to read Type 11h/12h SDR failed
```

```
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x4260a028
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 12 on SA 0x10
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 14: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 14: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: Register LED
infrastructure, FRU 2: status = 0
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 14: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2: offset 0, count 8,
cc=00, returned 8
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2: New HPDL
data=(nil), length=0; SDR data=(nil), length=0
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 14: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 15: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x4260a0a8
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 20 on SA 0x10
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x4260a128
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 21 on SA 0x10
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: Set fan level on FRU 2
(0x415156a8): new=8, cur=1, new threshold mask 0
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 15: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x4260a1a8
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 22 on SA 0x10
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 15: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 15: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x4260a228
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 16: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 16: Get Device Locator Record ID completion code = 0xc9
```

```
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 16: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 16: 3 attempts to read Type 11h/12h SDR failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 17: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 17: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: Cooling thread: state
changed from UNKNOWN to NORMAL for sensor { 0xA2, 4, 0: reading 00 C0 00;
reading2 00 00 00 }

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 17: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 17: 3 attempts to read Type 11h/12h SDR failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 18: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 18: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: Cooling thread: state
changed from UNKNOWN to NORMAL for sensor { 0xA2, 8, 0: reading 12 C0 C0;
reading2 00 00 00 }

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 18: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 18: 3 attempts to read Type 11h/12h SDR failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 19: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 19: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 19: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 19: 3 attempts to read Type 11h/12h SDR failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 20: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 20: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 20: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 20: 3 attempts to read Type 11h/12h SDR failed

Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: Sending fan state, 20, 2:
last_open=0, fan level=8 (0), state=0, prev_state=0, cause=0, locked=0,
deact_locked=0, activation_index=0

Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 1: raw:
```

```
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 2: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 3: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 4: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 5: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: Cooling thread: state
changed from UNKNOWN to NORMAL for sensor { 0xA2, 9, 0: reading 91 C0 C0;
reading2 00 00 00 }
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 6: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 7: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 8: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 9: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 10: raw:
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 21: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [225]: Cooling thread: state
changed from UNKNOWN to NORMAL for sensor { 0xA2, 19, 0: reading 91 C0 C0;
reading2 00 00 00 }
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 21: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 21: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 21: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 11: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 12: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 13: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 14: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 15: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: FRU 2, fan level 16: raw:
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: Controller 20, FRU 2:
State set to M1, prev=M0, cause=0, locked=0
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: HPDL Sensor: Setting
initial update delay 15 s for the tacho: SA 0x20, LUN 0, sensor 240
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: Sensor (20,0,240)
registered, initial_update=1
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: HPDL Sensor: Setting
initial update delay 15 s for the tacho: SA 0x20, LUN 0, sensor 241
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: Sensor (20,0,241)
registered, initial_update=1
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: CarrierHPDL:
alarm_configuration_type = 0
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: HPDL-Carrier AlarmInit:
alarm_conf=0
```

```
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [177]: Hot Swap event: SA=0x20,
FRU 2, sensor_number = 0x0d, M0->M1, cause 0, ignore=0

Oct 24 17:49:40 chmm~110 daemon.info shelfman: [177]: State M1, SA 0x20, FRU 2,
op_state=M0, power_cycle=0

Oct 24 17:49:40 chmm~110 daemon.info shelfman: [231]: HPDL Chassis:
SetAdaptiveThreshold thread started...

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [187]: Resolve for
"TelcoAlarmMinor" against "BTC_700": FindQualifiedObject failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [187]: Resolve for
"TelcoAlarmMajor" against "BTC_700": FindQualifiedObject failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 22: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 22: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [187]: Resolve for
"TelcoAlarmCritical" against "BTC_700": FindQualifiedObject failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [187]: Resolve for
"TelcoAlarmPower" against "BTC_700": FindQualifiedObject failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [187]: Resolve for
"TelcoAlarmMinorLed" against "BTC_700": FindQualifiedObject failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [187]: Resolve for
"TelcoAlarmMajorLed" against "BTC_700": FindQualifiedObject failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [187]: Resolve for
"TelcoAlarmCriticalLed" against "BTC_700": FindQualifiedObject failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [187]: Resolve for
"TelcoAlarmMinorClear" against "BTC_700": FindQualifiedObject failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [187]: Resolve for
"TelcoAlarmMajorClear" against "BTC_700": FindQualifiedObject failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [187]: Resolve for
"TelcoAlarmCutoff" against "BTC_700": FindQualifiedObject failed

Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: Same-owner=0

Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: +++ Setting radial
descriptor for pseudo-radial

Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]:
carrier_common_pseudo_radial_init: local=10, other=12

Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: IPMB link mapping record
not found in Shelf FRU info, assuming bused IPMB

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [187]: Shelfman: failed to
initialize pseudo-radial IPMB, assuming bused IPMB, error: 0xFD(-3)

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 22: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 22: 3 attempts to read Type 11h/12h SDR failed

Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 23: Get Device Locator Record ID completion code = 0xc9
```

```
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 23: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: Sending FRU state, 20, 1:
last_open=0, state=4, prev_state=1, cause=0, locked=0, deact_locked=1,
activation_index=2
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 23: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:40 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 23: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [186]: SDR Repository: dynamic
SDRs population has changed for SA 0x20, ind=28f5bd3
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [186]: Initial indicator for SA
0x20: 28f5bd3
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41525830
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41517e58
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41517ed8
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41517f58
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41517fd8
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41525830 to the internal event queue
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41517e58 to the internal event queue
Oct 24 17:49:40 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41517ed8 to the internal event queue
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [233]:
sdrrep_full_ipmc_sdrs_update_thread for 20
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [233]:
sdrrep_full_ipmc_sdrs_update_thread for 20: attempt 0
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41517f58 to the internal event queue
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41517fd8 to the internal event queue
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [234]: CONFIGURING 0x10 # 0
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [233]: Delete SDR in
sdrrep_full_ipmc_sdrs_update_thread: SA=20, sens=-117
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x41518058
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling:
allocated an internal event 0x415180d8
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x41518058 to the internal event queue
```

```
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: +ipmc_zoned_cooling: moved
event 0x415180d8 to the internal event queue
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [181]: Power not applied  SA
0x10, FRU 0: this is a VSO FRU
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [236]: CONFIGURING 0x20 # 0
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [181]: Power not applied  SA
0x20, FRU 0: this is a VSO FRU
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [233]: Delete SDR in
sdrrep_full_ipmc_sdrs_update_thread: SA=20, sens=-118
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [233]: Delete SDR in
sdrrep_full_ipmc_sdrs_update_thread: SA=20, sens=-119
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [233]: Delete SDR in
sdrrep_full_ipmc_sdrs_update_thread: SA=20, sens=-120
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [233]: Delete SDR in
sdrrep_full_ipmc_sdrs_update_thread: SA=20, sens=-121
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [233]: Delete SDR in
sdrrep_full_ipmc_sdrs_update_thread: SA=20, sens=-123
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [233]: Delete SDR in
sdrrep_full_ipmc_sdrs_update_thread: SA=20, sens=-124
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [233]: Delete SDR in
sdrrep_full_ipmc_sdrs_update_thread: SA=20, sens=-128
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [233]: Cooling: Registered Fan
Tray at IPMB = 0x20, FRU Id = 2, Entity = {0x1E,0x60}
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [237]: SA 0x20, FRU 1 is
ACTIVATING  (Postponed)
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [237]: Tasklet ACTIVATE (20, 1):
is_local_address = 1
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [237]: Tasklet ACTIVATE:
evaluating ipmc_hpm3_run_proxy(SA=20, FRU 1 IPMB-L=00), proxy_run=0
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [238]: CONFIGURING 0x20 # 1
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 24: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [181]: Power not applied  SA
0x20, FRU 1: this is a VSO FRU
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [186]: sdrrep: fetched new SDRs,
time=11
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 24: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: !!! shelf_fru_info_update,
forced=0
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: !!! Entry: SA=20, FRU=1,
invalid=0, shelf_fru=1
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: *** Calling for Shelf FRU
Info update within retrieve, op=1, count=1, status=0 ***
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: ***
shelf_fru_retrieve_fru_info: record ID=4(0004) type=0x12, op = 4 ***
```

```
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: ***
shelf_fru_retrieve_fru_info: record ID=5(0005) type=0x11, op = 4 ***

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: IPMC 0x20, FRU 1 is in the
list already, exiting

Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 24: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 24: 3 attempts to read Type 11h/12h SDR failed

Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 25: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: ***
shelf_fru_retrieve_fru_info: record ID=5(0005) type=0x11, op = 2 ***

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [175]: Controller 20, FRU 2:
State set to M4, prev=M1, cause=0, locked=0

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: ***
shelf_fru_retrieve_fru_info: record ID=5(0005) type=0x11, op = 1 ***

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: *** Controller was removed
and then added again, so reread FRU (SA=20, FRU ID=1)***

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: *** FRU Inventory Area
Info [20#1], size = 8192, access = 0, status = 0

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [177]: Hot Swap event: SA=0x20,
FRU 2, sensor_number = 0x0d, M1->M4, cause 0, ignore=0

Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 25: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [175]: HpdlFruActivate: 20, 2

Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 25: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 25: 3 attempts to read Type 11h/12h SDR failed

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [240]: SA 0x20, FRU 2 is
ACTIVATING

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [240]: Tasklet ACTIVATE (20, 2):
is_local_address = 1

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [240]: Tasklet ACTIVATE:
evaluating ipmc_hpm3_run_proxy(SA=20, FRU 2 IPMB-L=00), proxy_run=0

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [241]: CONFIGURING 0x20 # 2

Oct 24 17:49:41 chmm~110 daemon.info shelfman: [181]: Power not applied  SA
0x20, FRU 2: this is a VSO FRU

Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 26: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 26: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 26: Get Device Locator Record ID completion code = 0xc9

Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 26: 3 attempts to read Type 11h/12h SDR failed
```

```
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 27: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 27: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 27: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 27: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 28: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 28: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 28: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 28: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 29: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: Zoned cooling: Sending Get
Fan Level to pft=0x4260a028, SA=20, fru=2, VSO=1
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 29: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: Cooling: Decreasing speed
for Fan Tray at IPMB = 0x20, FRU Id = 2, from 8 to 7 level
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: Hpdl Cooling Management: 1
fan FRUs present, expected 2
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: Maximizing 0x20, 3 fan
speed: fan is absent
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: Hpdl Cooling Management:
some fan FRUs are not active: total 2, active 1
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 4 on SA 0x10
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 29: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 29: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41525830
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 12 on SA 0x10
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 30: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41517e58
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 20 on SA 0x10
```

```
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [221]: Resolve for
"Eth0PhysicalLinkDetected" against "BTC_700": FindQualifiedObject failed
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 30: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41517ed8
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 21 on SA 0x10
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 30: Get Device Locator Record ID completion code = 0xc9
Oct 24 17:49:41 chmm~110 daemon.warn shelfman: [182]: Tasklet Tasklet GET_INFO
for SA 0x20, FRU 30: 3 attempts to read Type 11h/12h SDR failed
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: FRU Info 20#1: removing
locked entry, updated=0
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: DELETING FRU 0x20, 1
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: Added IPMC 0x20, FRU 1 to
the FRU list, treated as Shelf FRU Info storage, size = 8192 (data size = 624)
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: !!! shelf_fru_info_update,
forced=0
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: !!! Entry: SA=20, FRU=1,
invalid=0, shelf_fru=1
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [187]: *** Calling for Shelf FRU
Info update within retrieve, op=1, count=1, status=0 ***
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41517f58
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 22 on SA 0x10
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41517fd8
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 4 on SA 0x20
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41518058
Oct 24 17:49:41 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 10 on SA 0x20
…
Oct 24 17:49:51 chmm~110 daemon.info shelfman: [177]: Ignored Hot Swap event:
from 20, seq_no=9
Oct 24 17:49:51 chmm~110 daemon.info shelfman: [243]: CONFIGURING 0x20 # 1
Oct 24 17:49:51 chmm~110 daemon.info shelfman: [181]: Power not applied  SA
0x20, FRU 1: this is a VSO FRU
Oct 24 17:49:51 chmm~110 daemon.info shelfman: [242]: CONFIGURING 0x20 # 0
Oct 24 17:49:51 chmm~110 daemon.info shelfman: [181]: Power not applied  SA
0x20, FRU 0: this is a VSO FRU
Oct 24 17:49:51 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x41501d48
```

```
Oct 24 17:49:51 chmm~110 daemon.info shelfman: [225]: Zoned Cooling: reread
sensor 16 on SA 0x20

Oct 24 17:49:51 chmm~110 daemon.info shelfman: [244]: CONFIGURING 0x20 # 2

Oct 24 17:49:51 chmm~110 daemon.info shelfman: [181]: Power not applied  SA
0x20, FRU 2: this is a VSO FRU

Oct 24 17:49:51 chmm~110 daemon.info shelfman: [225]: +ipmc_zoned_cooling:
releasing internal event 0x415011d0

Oct 24 17:49:52 chmm~110 daemon.info shelfman: [225]: Zoned cooling: Sending Get
Fan Level to pft=0x4260a028, SA=20, fru=2, VSO=1

Oct 24 17:49:52 chmm~110 daemon.info shelfman: [225]: Cooling: Decreasing speed
for Fan Tray at IPMB = 0x20, FRU Id = 2, from 8 to 7 level

Oct 24 17:49:52 chmm~110 daemon.info shelfman: [225]: Zoned cooling: Sending Get
Fan Level to pft=0x4260a028, SA=20, fru=2, VSO=1

Oct 24 17:49:52 chmm~110 daemon.info shelfman: [225]: Zoned cooling: Sending Set
Fan Level to pft=0x4260a028, SA=20, fru=2, VSO=1

Oct 24 17:49:52 chmm~110 daemon.info shelfman: [176]: Set fan level on FRU 2
(0x415156a8): new=7, cur=8, new threshold mask 0

Oct 24 17:49:52 chmm~110 daemon.info shelfman: [231]: HPDL Chassis: Updating
adaptive thresholds for FT 2: cur=7, old=255

Oct 24 17:49:52 chmm~110 daemon.info shelfman: [176]: Sending fan state, 20, 2:
last_open=0, fan level=7 (8), state=4, prev_state=1, cause=0, locked=0,
deact_locked=1, activation_index=3

Oct 24 17:49:52 chmm~110 daemon.info shelfman: [225]: Hpdl Cooling Management: 1
fan FRUs present, expected 2

Oct 24 17:49:52 chmm~110 daemon.info shelfman: [225]: Maximizing 0x20, 3 fan
speed: fan is absent

Oct 24 17:49:52 chmm~110 daemon.info shelfman: [225]: Hpdl Cooling Management:
some fan FRUs are not active: total 2, active 1

Oct 24 17:49:55 chmm~110 daemon.info shelfman: [172]: Resetting initial_update
flag for the sensor: SA 0x20, FRU 2, LUN 0, sensor 241 (postponed)

Oct 24 17:49:55 chmm~110 daemon.info shelfman: [172]: Resetting initial_update
flag for the sensor: SA 0x20, FRU 2, LUN 0, sensor 240 (postponed)

Oct 24 17:49:56 chmm~110 daemon.info shelfman: [249]: ***
shelf_fru_check_timer_sync called

Oct 24 17:49:56 chmm~110 daemon.info shelfman: [249]: Timeout for obtaining
Chassis FRU information has expired, check operation called

Oct 24 17:49:56 chmm~110 daemon.info shelfman: [249]: 1 Shelf FRUs have been
detected, 20 FRU 1 was NOT updated, 20 FRU 2 was NOT updated

Oct 24 17:49:56 chmm~110 daemon.info shelfman: [249]:
shelffru_redundancy_set_new_state: setting state for Shelf FRU redundancy to 2,
additional=0

Oct 24 17:50:22 chmm~110 daemon.info shelfman: [225]: Zoned cooling: Sending Get
Fan Level to pft=0x4260a028, SA=20, fru=2, VSO=1

Oct 24 17:50:22 chmm~110 daemon.info shelfman: [225]: Cooling: Decreasing speed
for Fan Tray at IPMB = 0x20, FRU Id = 2, from 7 to 6 level
```

```
Oct 24 17:50:22 chmm~110 daemon.info shelfman: [225]: Zoned cooling: Sending Get
Fan Level to pft=0x4260a028, SA=20, fru=2, VSO=1
Oct 24 17:50:22 chmm~110 daemon.info shelfman: [225]: Zoned cooling: Sending Set
Fan Level to pft=0x4260a028, SA=20, fru=2, VSO=1
Oct 24 17:50:22 chmm~110 daemon.info shelfman: [176]: Set fan level on FRU 2
(0x415156a8): new=6, cur=7, new threshold mask 0
Oct 24 17:50:22 chmm~110 daemon.info shelfman: [231]: HPDL Chassis: Updating
adaptive thresholds for FT 2: cur=6, old=7
Oct 24 17:50:22 chmm~110 daemon.info shelfman: [176]: Sending fan state, 20, 2:
last_open=0, fan level=6 (7), state=4, prev_state=1, cause=0, locked=0,
deact_locked=1, activation_index=3
<I> 17:21:12.263  [1292] Maximizing 0x20, 2 fan speed: fan is not active
<I> 17:21:12.264  [1292] Zoned Cooling: reread sensor 4 on SA 0x10
<I> 17:21:12.271  [1246] build_shmgr_fru_information: Time: mfg=1429025718,
start=820440000
<I> 17:21:12.273  [1303] CONFIGURING 0x82 # 0
<I> 17:21:12.275  [1263] Power not applied  SA 0x82, FRU 0: this is a VSO FRU
<I> 17:21:12.277  [1304] CONFIGURING 0x82 # 1
<I> 17:21:12.279  [1263] Power not applied  SA 0x82, FRU 1: this is a VSO FRU
<I> 17:21:12.282  [1263] Power not applied  SA 0x10, FRU 0: this is a VSO FRU
<I> 17:21:12.294  [1246] build_shmgr_fru_information: Final offset = 144
<I> 17:21:12.328  [1292] Zoned Cooling: reread sensor 10 on SA 0x10
<I> 17:21:12.360  [1292] Zoned Cooling: reread sensor 11 on SA 0x10
<I> 17:21:12.395  [1292] Zoned Cooling: reread sensor 12 on SA 0x10
<I> 17:21:12.430  [1264] SDR Repository: dynamic SDRs population has changed for
SA 0x20, ind=1467
<I> 17:21:12.431  [1264] Initial indicator for SA 0x20: 1467
<I> 17:21:12.433  [1311] sdrrep_full_ipmc_sdrs_update_thread for 20
<I> 17:21:12.434  [1311] sdrrep_full_ipmc_sdrs_update_thread for 20: attempt 0
<I> 17:21:12.441  [1311] Cooling: Registered Fan Tray at IPMB = 0x20, FRU Id =
2, Entity = {0x1E,0x60}
<I> 17:21:12.422  [1292] Zoned Cooling: reread sensor 4 on SA 0x20
<I> 17:21:12.499  [1292] Zoned Cooling: reread sensor 10 on SA 0x20
<I> 17:21:12.550  [1264] Initial indicator for SA 0x82: 0 (state new)
<I> 17:21:12.552  [1264] sdrrep: fetched new SDRs, time=6
<I> 17:21:12.563  [1313] sdrrep_full_ipmc_sdrs_update_thread for 82
<I> 17:21:12.565  [1313] sdrrep_full_ipmc_sdrs_update_thread for 82: attempt 0
<I> 17:21:12.614  [1292] Cooling thread: state changed from UNKNOWN to NORMAL
for sensor { 0x20, 16, 0: reading 00 C0 00; reading2 00 00 00 }
<I> 17:21:12.616  [1292] Zoned cooling: Sending Get Fan Level to pft=0x25eb98,
SA=20, fru=2, VSO=1
<W> 17:21:12.627  [1313] SDR Repository: failed to send Get Dev SDR to SA 0x82
(res = -62)
```

```
<I> 17:21:12.650  [1257] Hot Swap event: SA=0x82, FRU 0, sensor_number = 0x00,
M1->M4, cause 0, ignore=0
<I> 17:21:12.655  [1317] CONFIGURING 0x82 # 0
<I> 17:21:12.660  [1292] Cooling: Decreasing speed for Fan Tray at IPMB = 0x20,
FRU Id = 2, from 8 to 7 level
<I> 17:21:12.664  [1292] Hpdl Cooling Management: some fan FRUs are not active:
total 1, active 0
<I> 17:21:12.665  [1292] Maximizing 0x20, 2 fan speed: fan is not active
<I> 17:21:12.669  [1263] Power not applied  SA 0x82, FRU 0: this is a VSO FRU
<I> 17:21:12.729  [1246] build_shmm_fru_information: Time: mfg=1388692800,
start=820440000
<I> 17:21:12.731  [1246] build_shmm_fru_information: Final offset = 176
<I> 17:21:12.731  [1246] Chassis Manager: ChMM/Chassis Manager FRU Information
created
<I> 17:21:12.738  [1257] Hot Swap event: SA=0x82, FRU 1, sensor_number = 0x15,
M1->M4, cause 0, ignore=0
<I> 17:21:12.745  [1319] CONFIGURING 0x82 # 1
<I> 17:21:12.750  [1263] Power not applied  SA 0x82, FRU 1: this is a VSO FRU
<I> 17:21:12.754  [1246] Shelfman: Running
```

4.4 Starting SNMP and HTTP daemons manually

During normal operation the configuration variable *start_rc2_daemons* is set to *y* and daemons *snmpd*, *httpd* and *shelfman* are launched automatically during system startup by the *rc.common* script. Advanced users that choose to disable *start_rc2_daemons* variable can start service daemons manually with these commands:

SNMP daemon:

```
# daemon -f snmpd -Lsd -c /etc/snmpd.conf
```

HTTP daemon:

```
# daemon -f /bin/httpd -h /usr/httpd/html
```

4.5 Redundant operation

The active Chassis Manager exposes the ChMC device (address 20h) on IPMB, manages IPMB and the IPM Controllers and interacts with the System Manager over RMCP and other chassis-external interfaces. It maintains an open TCP connection with the backup Chassis Manager. It communicates all changes in the state of the managed objects to the backup Chassis Manager.

The backup Chassis Manager does not expose the ChMC on IPMB (but may expose itself as a physical controller on IPMB), does not actively manage IPMB and IPMcontrollers, nor interact with the System Manager via the chassis-external interfaces (with one exception noted below). Instead, it maintains the state of the managed objects in its own memory (volatile and non-volatile) and updates the state as directed by the active Chassis Manager. These state updates are not applied immediately, but are cached on the backup Chassis Manager.

The backup Chassis Manager may become active as the result of a switchover. Two types of switchover are defined:

- Cooperative switchover: the active and backup Chassis Managers negotiate the transfer of responsibilities from the active to the backup Chassis Manager; this mode is supported via the CLI `switchover` command issued on the active or backup Chassis Manager.
- Forced switchover: the backup Chassis Manager determines that the active Chassis Manager is no longer alive or healthy, and forcefully takes on the responsibilities of the active Chassis Manager. Forced switchover can also be initiated via the CLI `switchover` command with the option `–force`, addressed to the backup Chassis Manager.

As discussed briefly in section 2.4.3, redundant ChMMs share three Hardware Redundancy Interface (HRI) data items between them (for six items, total); these items together indicate, for each ChMM, the health, presence and switchover request state of the ChMM itself (referenced as the local state) and its peer (referenced as remote state). Each ChMM variant has a different hardware implementation of these signals.

At the software level, each of the Chassis Managers can access a register that provides a view of these data items. The register is implemented in the SmartFusion cSoC that is part of each ChMM. For details of the HRI implementation, please see the *Hardware Architecture Specification*. In the remainder of this section, these HRI data items are referenced informally as the Healthy, Presence and Switchover Request bits, each with Local and Remote variants. In addition, the set of HRI data items also includes an Active bit, which is 1 on the active Chassis Manager.

The backup Chassis Manager recognizes the departure of the active Chassis Manager when the Remote Healthy or Remote Presence bit becomes 0. The Remote Presence bit monitors the presence of the peer Chassis Manager. This bit changing to 0 indicates that the peer Chassis Manager has been reset or removed from the chassis.

WARNING: Only the ChMM carrier board hosting the backup Chassis Manager should be removed from a chassis. If the ChMM carrier board hosting the active Chassis Manager needs to be removed the user should first issue the CLI `switchover` command on the active or backup Chassis Manager, and then remove the ChMM carrier board after the switchover has completed.

The Remote Healthy bit is set by the peer Chassis Manager during initialization; this bit changing to 0 means that the remote Chassis Manager has become not healthy (typically, has been powered off or reset).

Another situation that needs some action from the backup Chassis Manager is when the TCP connection between the Chassis Managers gets closed. This happens when either the communication link between the two Chassis Managers is broken, or the `shelfman` process on the active Chassis Manager terminates, in a voluntary or involuntary way, or due to a software exception.

Also, since the keepalive option is enabled on the TCP connection, it closes shortly after the active ChMM is switched off or reset. In the case of Chassis Manager termination, it is possible that the TCP connection is closed before the Remote Healthy bit becomes 0. So, in order to determine why

the TCP connection closed, the backup Chassis Manager samples the state of the Remote Healthy bit immediately and, if it is still 1, samples it again after some delay. If the Remote Healthy bit becomes 0, the backup Chassis Manager concludes that the active Chassis Manager is dead. In that case it initiates a switchover and assumes the active Chassis Manager role.

Otherwise, if the Remote Healthy bit retains the value of 1, the backup Chassis Manager concludes that the communication link between the Chassis Managers is broken. In that case, no switchover is initiated; instead the backup Chassis Manager repeatedly reinitializes itself and tries to establish a connection with the active Chassis Manager, until the communication link is restored. Re-initialization is achieved by rebooting the ChMM and automatically restarting the Chassis Manager after the reboot. Special logic in the Chassis Manager guarantees that it does not try to become active at startup if the peer Chassis Manager is already active.

The Chassis Manager uses a watchdog timer to protect against becoming unresponsive due to infinite loops or other software bugs. In the event the watchdog timer on the active Chassis Manager triggers, that ChMM is reset, causing the Remote Healthy bit on the backup ChMM to become 0 and triggering a switchover.

In addition to the scenarios described above, the active Chassis Manager monitors the state of the Local Healthy and Active bits. If either of these bits becomes 0 on the active Chassis Manager, and the backup Chassis Manager is present, the active Chassis Manager initiates a reboot of the ChMM, in this way initiating a switchover to the backup Chassis Manager. This allows third-party applications on the ChMM to participate in monitoring health of the whole ChMM and initiate a switchover if necessary. Also, this behavior allows the former active ChMM to reinitialize itself as a backup in the case of a forced switchover from the backup Chassis Manager.

However, the behavior of a standalone Chassis Manager (one without a backup) in the case of the loss of either Local Healthy or Active bit is different.

In the case of the Local Healthy bit, the behavior is configurable and depends on the value of the configuration variable *EXIT_IF_HEALTHY_LOST_IN_STANDALONE_MODE*. By default, the value is *FALSE*, and in that case the Chassis Manager restores the Local Healthy bit and continues the operation. However if the value is *TRUE*, the standalone Chassis Manager reinitializes itself by rebooting the ChMM.

After a switchover, the former backup, now active, Chassis Manager performs additional initialization, applies cached state changes and collects any necessary further information from the IPM Controllers on IPMB. The new active Chassis Manager then exposes the ChMC device (address 20h) on IPMB, and assumes the IP address that was used for RMCP and other chassis-external interactions between the formerly active Chassis Manager and the System Manager. Since the RMCP session information is propagated from the active Chassis Manager to the backup Chassis Manager, RMCP sessions survive the switchover. For the System Manager using RMCP, the switchover is transparent.

The switchover is also transparent for the Web interface and for the SNMP interface (provided that they use the redundancy IP address that is switched over). Command-line interface sessions, since they are initiated locally on the target Chassis Manager, do not survive a switchover and need to

be re-established again on the newly active Chassis Manager. The command-line interface support on the backup Chassis Manager is limited, but it does allow the backup Chassis Manager to request a switchover using the `switchover` command.

The formerly active Chassis Manager after the switchover can cease to exist or reinitialize itself as the backup Chassis Manager. To reinitialize as the backup Chassis Manager the formerly active ChMM must reboot the operating system.

4.5.1 Initialization of the redundant Chassis Managers

When starting, the Chassis Manager tries to determine its role (active or backup) and establish a TCP connection with its peer Chassis Manager. It does this using the following algorithm:

- The Chassis Manager starts listening for incoming TCP connection requests from its peer Chassis Manager.
- If either of the Remote Healthy and Remote Present bits is 0, the Chassis Manager assumes that there is no peer Chassis Manager and assumes the active role.
- Otherwise, the Chassis Manager issues a connection request, trying to establish a TCP connection to its peer. If the connection request fails, the Chassis Manager assumes the active role.
- If the connection request succeeds and there is no pending incoming connection request, the Chassis Manager assumes that an active Chassis Manager already exists in the chassis, and assumes the backup role.
- If both: 1) the outgoing connection request succeeds and 2) an incoming connection request is pending, the situation is symmetric between the two Chassis Managers and one of the Chassis Managers should assume the active role and the other should assume the backup role. The decision is based on the value of the Chassis Manager geographic address.

After the connection is established, the active Chassis Manager performs the initial synchronization with the backup Chassis Manager, sending it all the appropriate redundant data. During the initial synchronization, the backup Chassis Manager verifies that the active Chassis Manager uses the same carrier-specific module. If the carrier-specific modules are different, further data propagation and collaboration between the active and the backup Chassis Managers is not possible. Therefore, in that case, the backup Chassis Manager does not proceed with initialization, but reboots and reestablishes the connection in a loop, expecting the active Chassis Manager with the correct carrier-specific module to appear. The only exception to this is that in order to facilitate upgrades from non-HPDL to HPDL-based configurations, the HPDL-based backup Chassis Manager can coexist with a non-HPDL active Chassis Manager, but only when a reliable upgrade operation is in progress.

4.5.2 Redundancy and HWRI State sensor

This discrete sensor is associated with the physical Chassis Manager, has number 128, name “HWRI State” and OEM event/reading type code DEh. The sensor uses the ChMM-700 HRI programmatic interface to obtain the values of the HRI bits.

The state mask reflects the high-level redundancy state (normal state) of the ChMM, as well as various low-level redundancy-related exceptions. Several bits of the exception group can be set in the mask, along with one of the normal state bits. The meanings of the bits in the sensor state mask are given below:

High-level redundancy state bits (normal state):

[0] - The current Chassis Manager is Active with no Backup.

[1] - The current Chassis Manager is Active, with a Backup.

[2] - The current Chassis Manager is a Backup.

Low-level exception state bits (should be treated as low-level errors if set):

[3] - The Chassis Manager is Active with a Backup but Backup is unavailable for switchover.

[4] - The Chassis Manager is a Backup but the remote presence bit is not set.

[5] - The Chassis Manager is a Backup but the remote active bit is not set.

[6] - The Chassis Manager is a Backup but the HRI Active bit is set.

[7] - The Chassis Manager is Active with a Backup but the remote presence bit is not set.

[8] - The Chassis Manager is Active with a Backup but the remote healthy bit is not set.

[9] - The Chassis Manager is Active with a Backup but the HRI Active bit is not set.

[10] - The local presence bit is not set for the current Chassis Manager.

[11] - The Chassis Manager is Active with no Backup, but the remote healthy bit is set.

[12] - The Chassis Manager is Active with no Backup but the remote active bit is set.

[13] - The Chassis Manager is Active but senses the HRI Active bit set on other ChMM.

The current reading and state mask of the sensor can be retrieved in many ways, e.g. with the CLI command `sensordata`.

The sensor also generates events when the state mask changes. The format of the event message is given in the following table:

Table 14: Format of event message

BYTE	DATA FIELD
1	*Event Message Rev* = 04h (IPMI 1.5 and IPMI 2.0)
2	*Sensor Type* = DEh (Redundancy and HWRI state)
3	*Sensor Number* = 80h
4	*Event Direction* [7] = 0b (Assertion) *Event Type* [6:0] = 6Fh (General Availability)

5	*Event Data 1:* [7:4] = 7h (previous state and severity in Event Data 2, OEM code in Event Data 3) [3:0] = The highest new offset: 00h - The current Chassis Manager is Active with no Backup. 01h - The current Chassis Manager is Active, with a Backup. 02h - The current Chassis Manager is a Backup. 03h - The Chassis Manager is Active with a Backup but Backup is unavailable for switchover. 04h - The Chassis Manager is a Backup but the remote presence bit is not set. 05h - The Chassis Manager is a Backup but the remote active bit is not set. 06h - The Chassis Manager is a Backup but the HRI Active bit is set. 07h - The Chassis Manager is Active with a Backup but the remote presence bit is not set. 08h - The Chassis Manager is Active with a Backup but the remote healthy bit is not set. 09h - The Chassis Manager is Active with a Backup but the HRI Active bit is not set. 0Ah - The local presence bit is not set for the current Chassis Manager. 0Bh - The Chassis Manager is Active with no Backup, but the remote healthy bit is set. 0Ch - The Chassis Manager is Active with no Backup but the remote active bit is set. 0Dh – The Chassis Manager is Active but senses the HRI Active bit set on other ChMM.
6	*Event Data 2:* [7:4] = Event Severity [3:0] = The highest previous offset
7	*Event Data 3:* On the ChMM-700, this byte should be ignored; its value is always 00h

4.5.3 Reboot Reason sensor

Both active and standby Chassis Managers also expose a sensor that indicates the reason for the last reboot. This discrete sensor is associated with the physical Chassis Manager, has number 129, name “Reboot Reason” and OEM event/reading type code DDh.

The state mask for the sensor indicates the cause of the last reboot; its bits have the following meanings:

[0] - The reboot reason is either a Chassis Manager crash or unknown.

[1] - The reboot was caused by a switchover operation.

[2] - The reboot was caused by a forced switchover operation.

[3] - The reboot was caused by the CLI command *`terminate`*.

[4] - The reboot was caused by loss of the HEALTHY bit.

[5] - The reboot was caused by loss of the ACTIVE bit.

[6] - The reboot of the Backup ChMM happened because the redundancy connection was broken but the Active ChMM was still alive.

[7] - The reboot happened due to an error during the Chassis Manager startup.

[8] - The reboot was caused by the ChMM hardware watchdog.

[9] – The reboot was initiated by software (a *`reboot()`* system call).

[10] – The ChMM has been power cycled.

[11] – The reboot happened because of an SDRAM error.

[12] – The reboot was caused by the IPMI “Cold Reset” command.

[13] – The reboot was caused by I2C Isolation switchover

[14] – The reboot was caused by a local System IPMB failure

The sensor reading byte is always 0 and does not have any meaning.

Note: in case of a Chassis Manager reboot initiated by software, this sensor may report two different reasons based on system shutdown speed. If the system shuts down within the default setting of 2 seconds (i.e. while the watchdog timer has not expired), the reboot reason will be a reboot() call (state mask = 9). Otherwise, the reboot reason will be the hardware watchdog. To avoid this non-deterministic result, it is possible to increase the `WATCHDOG_TIMEOUT` *configuration variable value, for example* `WATCHDOG_TIMEOUT` *could be increased to 5 seconds.*

In addition to the state mask reported by the reboot reason sensor, information about recent reboots is stored in the file *`/var/nvdata/reboot_reason.log`* on the ChMM. The typical log file is shown below:

```
04/10/2014 04:55:35 The reboot was caused by power operation
```

```
04/10/2014 04:41:04 The reboot was caused by power operation
04/09/2014 23:32:47 The reboot was caused by power operation
04/08/2014 05:14:29 The reboot was caused by WATCHDOG
04/07/2014 00:10:43 The reboot was caused by Chassis Manager abnormal
termination: signal = 11(Segmentation fault), pid = 139
04/06/2014 02:11:59 The reboot was caused by power operation
```

Each line in the file contains a timestamp and text description of a ChMM reboot. The text description normally corresponds to the description of the reboot reason sensor state shown above. However, if the corresponding reboot reason is 0 (“The reboot reason is either a Chassis Manager crash or unknown”) and it is known that a crash has occurred, the log message notes that and adds information about the signal that caused the crash, along with the Chassis Manager process identifier.

4.6 Operation in the chassis with radial System IPMB

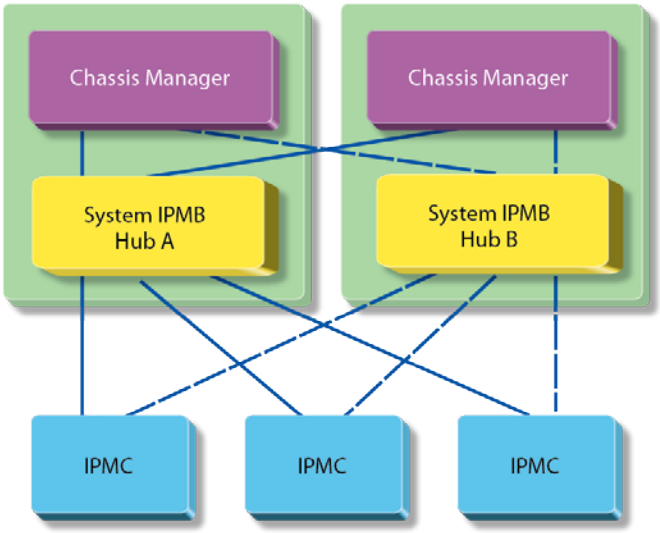
Some chassis implement a radial System IPMB instead of, or in addition to, a bused System IPMB. For a radial System IPMB, there is a distinct logical bus for each implemented IPMB (IPMB-A and IPMB-B) of each radial IPMB segment. The reliability of the radial System IPMB is improved compared to a bused System IPMB since each radial segement is separately accessible and permanently isolated from the other radial segments.

Two different topology types are supported for radial System IPMB: dual star and redundant dual star. The implemented topology is determined by the design of the ChMM carrier.

With the dual star topology, only one IPMB hub exists on each ChMM carrier. This hub connects the ChMM to radial links on either IPMB-A or IPMB-B, depending on the physical position of the ChMM carrier in the chassis. This topology has the following properties:

- Provides equivalent connectivity to the dual star topology that ATCA defines for the Gigabit Ethernet Base Interface
- Includes two cross-links that connect an IPMB (either A or B) port on one ChMM to the radial hub on the other ChMM carrier
- Requires two installed Chassis Managers for dual redundant operation of System IPMB
- Establishes two distinct and unconnected fault zones for the two Chassis Managers

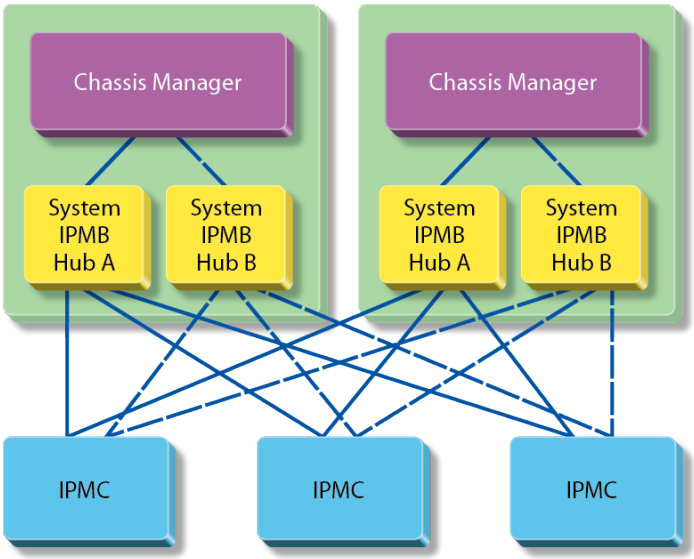
Figure 4: Dual star radial System IPMB topology



With the redundant dual star topology, each ChMM carrier has two System IPMB hubs, with each hub connecting one of the ChMM IPMB ports (A or B) to corresponding radial links on the backplane. This topology has the following properties:

- Supports dual redundant operation of System IPMB even with the only one Chassis Manager installed
- Does not need or include System IPMB cross-links between the two ChMM carriers

Figure 5: Redundant dual star radial System IPMB topology



The IPMB topology is always specified in HPDL and no IPMB Topology record in the Carrier FRU Information is needed or supported. The IPMB topology clause has the following syntax.

```
<ipmb_topology_clause> ::= "IPMB_TOPOLOGY" <bus_type>
       ["," <cross_link_num>] ";"
<bus_type> ::= "BUSED" | "DUAL_STAR" | "REDUNDANT_DUAL_STAR"
<cross_link_num> ::= 0 | 1["," "NOBUS"]
```

The value *BUSED* of the *<bus_type>* token indicates a bused System IPMB carrier. The default value is *BUSED*. For hybrid System IPMB designs, i.e. carrier designs that support both a radial System IPMB and a bused System IPMB, the radial System IPMB topology is specified. For carrier designs that only support a bused System IPMB topology the IPMB Topology clause is optional and does not need to be included.

The value *DUAL_STAR* corresponds to the “non-redundant dual star” radial topology (one Chassis Manager implements all the IPMB-A radial links, the other Chassis Manager implements all the IPMB-B radial links).

The value *REDUNDANT_DUAL_STAR* corresponds to the “redundant dual-star” radial topology where each Chassis Manager implements both IPMB-A and IPMB-B radial links.

The *<cross_link_num>* specifies the link number used for communication with the peer Chassis Manager in a radial configuration and is used only with the Redundant Dual Star topology. The default value is *0*, which means that no cross link exists and the Chassis Managers communicate with each other over the bused System IPMB. Currently, only link number *1* can be specified as the crosslink; as a result, the only legal values for *<cross_link_num>* are *0* and *1*.

If a cross link exists (i.e. *<cross_link_num>* is *1*), communication between redundant Chassis Managers goes over the cross link, even if a bused System IPMB also exists, as would be the case for a hybrid System IPMB topology. The keyword *NOBUS* can be specified if *<cross_link_num>* is *1*. This indicates a bused System IPMB is not physically implemented in the chassis. In this case, the active Chassis Manager does not attempt to access the bused System IPMB but emulates the bused System IPMB logically over the cross link. The bused IPMB link sensor (”IPMB LINK”) in this case monitors the state of the cross link.

4.6.1 Upgrading the routing element SmartFusion cSoC(s)

The routing element SmartFusion cSoC(s) can be programmed/upgraded using the *ripmb_prog* utility, which is included as part of the standard RFS image. In addition to programming the SmartFusion cSoC(s), this utility can be used to obtain the version of the currently-installed image(s).

IMPORTANT: before running *ripmb_prog* to program/upgrade the routing element SmartFusion cSoC(s), terminate the Chassis Manager by issuing a *clia terminate* command. Upon completion of the SmartFusion cSoC programming the Chassis Manager must be re-initialized by issuing a *reboot* command. Failing to do so may lead to SmartFusion cSoC access errors and

unpredictable behavior. If *ripmb_prog* is invoked only to obtain the version of the currently-installed image, terminating the Chassis Manager is not necessary.

To program a new SmartFusion cSoC image to the routing element SmartFusion cSoC(s), issue the following command:

ripmb_prog <image_name>

where *<image_name>* is the name of the upgrade image (with a ".dat" extension). The SmartFusion cSoC upgrade image file includes a SmartFusion A2F060 image concatenated with a SmartFusion A2F200 image. Each individual image includes all the programmable content needed for the A2F060 device or the A2F200 device (including FPGA fabric and Cortex-M3 firmware). The *ripmb_prog* utility will determine the SmartFusion device type (A2F060 or A2F200) as well as the number of SmartFusion devices that are connected to the JTAG chain and program the appropriate image from the concatenated SmartFusion cSoC image file to each device.

IMPORTANT: all routing element SmartFusion cSoCs on the ChMM Carrier must be programmed with the same version of the SmartFusion cSoC image file. If the upgrade procedure is interrupted at any point, it must be re-started to ensure all SmartFusion cSoCs are completely programmed. When the programming is successful, the following message(s) are displayed by the *ripmb_prog* utility (depending on the number of SmartFusion devices to be programmed):

```
Image file: radial.dat
Determining the number of devices in the JTAG scan chain: 2
Programming device #1:.........................................SUCCESS
Programming device #2: ........................................SUCCESS
```

To obtain the version of the currently-installed routing element image, issue the following command:

ripmb_prog -V

This command displays the version of the routing element SmartFusion cSoC firmware and FPGA design in the following format: *<firmware_version>/<fpga_version>*, for example: **1.8/1.1.0.0**. If this command displays **"VERSION MISMATCH"**, the SmartFusion cSoCs must be re-programmed with the desired image.

4.7 Automatic SEL truncation

The System Event Log (SEL) on the Chassis Manager stores events from all IPM Controllers and FRUs in the chassis and can easily exceed its maximum capacity. To prevent overflows, the Chassis Manager can automatically truncate the SEL, removing the oldest entries as SEL approaches its limits.

The automatic truncation algorithm works as follows. The two configuration parameters are defined: *SEL_HIGH_WATERMARK* and *SEL_LOW_WATERMARK*. The first parameter relates to the percentage of free entries in the SEL, and the second one relates to the percentage of occupied entries in the SEL. When the percentage of free SEL entries falls below *SEL_HIGH_WATERMARK*, the truncation thread wakes up and starts deleting the oldest entries from the SEL using the IPMI

command “Delete SEL Entry”. The thread works until the percentage of occupied entries in the SEL falls below *SEL_LOW_WATERMARK*; then it stops and stays dormant until the percentage of free SEL entries falls below the *SEL_HIGH_WATERMARK* again.

This algorithm is enabled by default, but can be turned off by setting the configuration variable *SYSTEM_MANAGER_TRUNCATES_SEL* to *TRUE*. In that case, automatic truncation is turned off, but the Chassis Manager assists the System Manager by indicating the current state of the SEL via the sensor SEL State of the type Event Logging Enabled, as follows (in accordance with the IPMI 2.0 specification):

- The reading returned by that sensor is equal to the percentage of the SEL that is occupied (0 to 100);
- The sensor assumes the state SEL Almost Full (5h) when the percentage of free entries in the SEL falls below the high watermark (the value of the configuration parameter *SEL_HIGH_WATERMARK*). An event is generated in that case and placed in the SEL; Event Data Byte 3 of the event contains the percentage of the SEL that is occupied. No event if generated for this state until the occupancy of the SEL falls below the low watermark; this is done to prevent multiple events if SEL occupancy oscillates around the high watermark.
- A custom state (6h) is defined for the state when the SEL occupancy is below the low watermark (the percentage of occupied entries falls below the value of the configuration parameter *SEL_LOW_WATERMARK*). There is no event associated with this state; the System Manager can detect this state by polling the sensor value and its asserted states.

Using this information, the System Manager performs truncation of the SEL, presumably using the IPMI command “Delete SEL Entry”, sending it over the RMCP session.

4.8 Cooling state sensors

The cooling management strategy in the Chassis Manager is configurable and also depends on the currently chosen carrier. Cooling management strategy configuration is described in section 3.7. Irrespective of the current cooling management strategy, however, several sensors are defined that expose the current cooling state of the chassis to the System Manager. These sensors belong to the logical Chassis Manager (IPMB address 20h) and are defined in the following table:

Table 15: Cooling state sensors

Sensor number	Sensor name	Event/Reading type	Sensor type	Offsets
135	FT Oper. Status	Redundancy (Bh)	Management Subsystem Health (28h)	00h = Full Redundancy = all fan trays defined in the Address Table are operational 01h = Redundancy Lost = some of the fan trays defined in the Address Table are

				missing or non-operational. With the HPDL default cooling management strategy this causes the fan level for all remaining fan trays to be set to their maximum
136	Cooling State	Severity (7)	Management Subsystem Health (28h)	00h = transition to OK. The cooling state is Normal 01h = transition to Non-Critical from OK. The cooling state is now Minor Alert, the previous cooling state was Normal. 02h = transition to Critical from less severe. The cooling state is now Major Alert, the previous cooling state was either Normal or Minor Alert. 04h = transition to Non-Critical from more severe. The cooling state is now Minor Alert, the previous cooling state was either Major or Critical Alert. 05h = transition to Critical from Non-recoverable. The current cooling state is Major Alert, the previous cooling state was Critical Alert. 06h = transition to Non-recoverable. The current cooling state is now Critical Alert.
137	Fans State	Severity (7)	Management Subsystem Health (28h)	00h = transition to OK. The fans state is Normal (no thresholds are crossed on fan tachometer sensors) 01h = transition to Non-Critical from OK. The fans

				state is now Minor Alert (non-critical thresholds are crossed for some tachometer sensors), the previous fans state was Normal. 02h = transition to Critical from less severe. The fans state is now Major Alert (critical thresholds are crossed for some tachometer sensors), the previous fans state was either Normal or Minor Alert. 04h = transition to Non-Critical from more severe. The fans state is now Minor Alert, the previous fans state was either Major Alert or Critical Alert (non-recoverable thresholds are crossed for some tachometer sensors). 05h = transition to Critical from Non-recoverable. The current fans state is Major Alert, the previous fans state was Critical Alert. 06h = transition to Non-recoverable. The current fans state is now Critical Alert.

4.9 Deadlock detection

The Chassis Manager contains built-in facilities for detecting internal deadlocks. A deadlock happens if two or more threads in the Chassis Manager application acquire locks in an incorrect order, so that one thread owns the lock requested by the other thread, and vice versa. As a result, the Chassis Manager becomes unresponsive on its external interfaces, though it still continues to run. Any deadlock is a result of a serious programming bug and should not happen during normal operation. The deadlock detection facilities, however, add an additional layer of protection and ensure that should a deadlock happen during normal operation of the Chassis Manager, a switchover to the backup Chassis Manager takes place and the Chassis Manager continues operation without interruption.

There are two deadlock detection facilities: one that is triggered by watchdog timer expiration and one that directly detects incorrect locking patterns in the Chassis Manager.

The watchdog-based deadlock detection facility monitors the operation of the most important interface-oriented subsystems of the Chassis Manager, which now include the CLI processing thread and the RMCP processing threads. Normally, if a deadlock takes place anywhere in the Chassis Manager, one or more of these threads is affected immediately or very soon as a result of deadlock propagation.

The threads monitored by the watchdog-based deadlock detection facility must periodically strobe the internal watchdog timer. The strobe period is set separately for each type of monitored thread and is 30 seconds for the CLI thread and 3 seconds for the RMCP threads. If one of the monitored threads fails to strobe the internal watchdog timer on timely basis, the Chassis Manager does the following:

- issues a warning message to the console and/or to the syslog file.
- collects information about current usage of locks in the Chassis Manager.
- writes this information to a Flash file on the ChMM (*/var/nvdata/dumplog.bin*), where it survives reboots and can be subsequently analyzed by the analysis tool *dumplog*.
- if one of the monitored threads fails to do a timely strobe of the internal watchdog timer three times in a row, stops strobing the hardware watchdog timer, which causes a reset of the ChMM and a switchover to the backup Chassis Manager.

In this scenario, output similar to the following appears on the console:

```
<W> 17:25:49.368   [218] [ipmf_watchdog_thread_proc] 239 facility has failed,
don't confirm the watchdog
<I> 17:25:49.760   [218] Write 132232 bytes to the '/var/nvdata/dumplog.bin', 0
seconds, lock offset 131848
<W> 17:25:50.068   [218] [ipmf_watchdog_thread_proc] 239 facility has failed,
don't confirm the watchdog
<W> 17:25:50.378   [218] [ipmf_watchdog_thread_proc] 239 facility has failed,
don't confirm the watchdog
<W> 17:25:50.688   [218] [ipmf_watchdog_thread_proc] 239 facility has failed,
don't confirm the watchdog
<W> 17:25:50.998   [218] [ipmf_watchdog_thread_proc] 239 facility has failed,
don't confirm the watchdog
```

For postmortem analysis of a deadlock, the utility *dumplog* (supplied in the Chassis Manager RFS) can be used. The following command line invokes this utility:

```
# dumplog –f /var/nvdata/dumplog.bin
```

The result is produced on standard output in a human-readable format. It includes information about which locks are owned by which threads, and a per-thread history of lock acquisition and release, for each currently active Chassis Manager thread. The output of the command looks something like this:

```
*** 24, Log for pthread with pid = 263, 18:17 ***
```

```
Jun 22 17:23:27 2007: Succeeded 2ac34530; pid=263, file="sdrrep.c", line=4288
Jun 22 17:23:27 2007: Released 2ac34530; pid=263, file="sdrrep.c", line=3715
Jun 22 17:23:27 2007: Acquiring 2ac34ad0; pid=263, file="ipmc.c", line=387
Jun 22 17:23:27 2007: Succeeded 2ac34ad0; pid=263, file="ipmc.c", line=387
Jun 22 17:23:27 2007: Released 2ac34ad0; pid=263, file="ipmc.c", line=484
Jun 22 17:23:27 2007: Acquiring 7d5ff740; pid=263, file="msg.c", line=775
Jun 22 17:23:27 2007: Succeeded 7d5ff740; pid=263, file="msg.c", line=775
Jun 22 17:23:27 2007: Acquiring 2acc1f80; pid=263, file="ipmf.c", line=1360
Jun 22 17:23:27 2007: Succeeded 2acc1f80; pid=263, file="ipmf.c", line=1360
Jun 22 17:23:27 2007: Acquiring 2ac344a0; pid=263, file="msg.c", line=924
Jun 22 17:23:27 2007: Succeeded 2ac344a0; pid=263, file="msg.c", line=924
Jun 22 17:23:27 2007: Released 2ac344a0; pid=263, file="msg.c", line=932
Jun 22 17:23:27 2007: Acquiring 2ac344a0; pid=263, file="msg.c", line=867
Jun 22 17:23:27 2007: Succeeded 2ac344a0; pid=263, file="msg.c", line=867
Jun 22 17:23:27 2007: Released 2ac344a0; pid=263, file="msg.c", line=876
Jun 22 17:23:27 2007: Acquiring 2acbe5ec; pid=263, file="ipmf_ipmb.c", line=1649
Jun 22 17:23:27 2007: Succeeded 2acbe5ec; pid=263, file="ipmf_ipmb.c", line=1649
Jun 22 17:23:27 2007: Released 2acbe5ec; pid=263, file="ipmf_ipmb.c", line=1656
Jun 22 17:23:27 2007: Acquiring 2acc1de0; pid=263, file="ipmf_ipmb_buffe",
line=98
Jun 22 17:23:27 2007: Succeeded 2acc1de0; pid=263, file="ipmf_ipmb_buffe",
line=98
Jun 22 17:23:27 2007: Released 2acc1de0; pid=263, file="ipmf_ipmb_buffe",
line=129
Jun 22 17:23:27 2007: Released 2acc1f80; pid=263, file="ipmf.c", line=1533
Jun 22 17:23:27 2007: CondWait Before 7d5ff740; pid=263, file="msg.c", line=790
Jun 22 17:23:27 2007: CondWait After 7d5ff740; pid=263, file="msg.c", line=790
……
Jun 22 17:23:18 2007: Succeeded 2ac34970; pid=216, file="shelffru.c", line=3171
Jun 22 17:23:18 2007: Released 2ac34970; pid=216, file="shelffru.c", line=3196
Jun 22 17:23:18 2007: Acquiring 2ac34970; pid=216, file="shelffru.c", line=3171
Jun 22 17:23:18 2007: Succeeded 2ac34970; pid=216, file="shelffru.c", line=3171
Jun 22 17:23:18 2007: Released 2ac34970; pid=216, file="shelffru.c", line=3196
Total of 32 locks
2ac34530: owner=0, waiting=0
2ac34ad0: owner=0, waiting=0
2acc1f80: owner=0, waiting=0
2ac344a0: owner=0, waiting=0
2acbe5ec: owner=0, waiting=0
2acc1de0: owner=0, waiting=0
2ac34950: owner=0, waiting=0
```

```
2ac36230: owner=0, waiting=0
2ac34b90: owner=0, waiting=0
2acbe620: owner=0, waiting=0
2ac34df0: owner=0, waiting=0
2ac34970: owner=0, waiting=0
1004bd60: owner=0, waiting=0
2ac36350: owner=0, waiting=0
10051070: owner=0, waiting=0
2ac34450: owner=0, waiting=0
2acc1e28: owner=0, waiting=0
2acc1e40: owner=0, waiting=0
10090890: owner=0, waiting=0
2ac344f0: owner=0, waiting=0
2acbdb98: owner=0, waiting=0
10091fa8: owner=0, waiting=0
10091c10: owner=0, waiting=0
100923e8: owner=0, waiting=0
10092840: owner=0, waiting=0
1008d7a4: owner=0, waiting=0
7efffa80: owner=0, waiting=0
2ac34430: owner=0, waiting=0
2ac36310: owner=0, waiting=0
2ac33d98: owner=0, waiting=0
1008d81c: owner=0, waiting=0
10094c08: owner=0, waiting=0
```

Deadlock detection shall be turned off if the value of the configuration variable *DETECT_DEADLOCKS* is *FALSE*.

4.10 I^2C bus fault isolation

On some carriers, ChMM GPIOs can be used to reset an I^2C multiplexer or I^2C switch that is implemented on an I^2C bus. By resetting the I^2C multiplexer or switch, it is possible to implement an algorithm in the Chassis Manager to isolate faulty devices on subordinate I^2C buses that are "behind" that multiplexer/switch, so that access to devices on other I^2C buses can still occur successfully.

This feature allows more than one I^2C multiplexer or switch to be resettable. These multiplexers or switches can also be located on multiple I^2C buses. The implementation applies only to HPDL-based chassis. The information about reset support and the number of the ChMM GPIO used to reset the device must be contained in the HPDL device description of the corresponding multiplexer or switch.

The typical HPDL description of an I^{2}C multiplexer supporting reset looks as follows:

```
DEVICE Multiplexor {
    PCA9545: 5: 0x70;
    SIGNALS {
      Channels: Channels, "6", "7", "8", "9", "Reset=8", "SwitchoverThreshold=10;
      #MuxIsolated: IsolatedChannelMask;
      MuxIslCh1: Channel1_Isolated;
      MuxIslCh2: Channel2_Isolated;
      MuxIslCh3: Channel3_Isolated;
      MuxIslCh4: Channel4_Isolated;
    };
  };
```

A configuration parameter for the signal “Channels” in the form “Reset=<n>” specifies the number of the ChMM GPIO used to reset the device, This GPIO is chosen by the designers of the ChMM carrier, For distinct multiplexers, different GPIOs must be used.

Signals with internal names in the form “Channel<n>_Isolated” report the isolation state of a specific channel of the multiplexer or switch (starting from 1). The value of the signal is 1 if the channel is currently isolated by the isolation algorithm and 0 otherwise. These signals can be exposed as IPMI sensors using normal HPDL mechanisms.

The signal “IsolatedChannelMask” reports the bitmask that has 1s for currently isolated channels (bit 0 of the bitmask corresponds to the first channel, bit 1 – to the second channel, etc.).

In this mode, the isolation algorithm operates as follows:

- When the error ETIMEDOUT is detected by the Linux I^{2}C subsystem while trying to access a device on a subordinate bus, this bus is recorded as faulty in the Chassis Manager and the multiplexer/switch is held in reset via the corresponding GPIO. This operation causes all the channels to be deselected on the multiplexer/switch, which allows the Chassis Manager to access devices on the main master-only I^{2}C bus even if one of the subordinate buses has a fault.
- The multiplexer/switch channel that caused the error is noted and values of the signals “Channel<n>_Isolated” and “IsolatedChannelMask” are modified accordingly.
- The multiplexer/switch is re-enabled via the corresponding GPIO the next time some part of the Chassis Manager tries to access it. This allows the Chassis Manager to access devices on the subordinate buses that are still functional.
- When an attempt is made to access the faulty subordinate bus (by writing the channel number or mask for the faulty bus into the multiplexer/switch), this operation is not permitted for a certain (configurable) period of time after the failure. During this period, the

faulty bus is not enabled; an error is instead returned for the corresponding write operation. This time period is specified by the configuration parameter *ISOLATE_MUX_IGNORE_TIME*.

- The algorithm knows which value in the multiplexer/switch register corresponds to the faulty bus, because it monitors all write accesses to the multiplexer/switch registers and remembers the last value before the occurrence of the fault.

- When the “ignore time” expires, an attempt is made to access the faulty bus and check whether the fault is still present. If the attempt fails, the “ignore time” is reset to its original value. If the attempt succeeds, then the recorded fault condition for that bus is removed and operation continues normally.
- With this approach, it is also possible to initiate a switchover to the backup Chassis Manager if the main I^2C bus remains inaccessible after the resetting the I^2C switch/multiplexer. If the configuration parameter "SwitchoverThreshold=<N>" on the Channel signal is defined as a non-zero value, it specifies the maximum consecutive number of attempts to reset the switch/multiplexer. If after that number of attempts the main I^2C bus is still inaccessible, a switchover to the backup Chassis Manager is initiated and the active Chassis Manager is rebooted.

4.11 Assignment of LAN configuration parameters to boards and modules

In chassis where some of the boards and intelligent modules implement LAN IPMI channels, there may be a need for a centralized repository of LAN configuration parameters (mainly the IP addresses, subnet masks and gateway addresses) for these boards and modules. This repository can be maintained on the Pigeon Point Chassis Manager. The parameters are stored statically on the ChMM or in the Chassis FRU Information. Also the Chassis Manager supports obtaining parameters from a DHCP server using the DHCP protocol.

One use for such LAN IPMI channels, once configured, is to support serial over LAN sessions via the management controllers on the boards or modules. This use involves a significant number of additional IPMI parameters. In the current Chassis Manager, only the following three parameters are supported: IP address, subnet mask and default gateway address. Other parameters (for instance, MAC addresses) may be added in future releases of the Chassis Manager.

Note: this facility is superceded by the PICMG HPM.3 specification, which specifies a standard way to assign LAN configuration parameters to boards and modules in AdvancedTCA shelves using DHCP. This functionality will likely be removed from the Chassis Manager at some point, when a mapping of HPM.3 to VPX systems becomes available.

4.11.1 Structure and composition of the supported parameters

The parameters are specified on a per-slot basis. For VITA 46.11 modules, the slot number is the board physical slot number (site number). For each slot, several parameter sets may be defined. Each parameter set contains parameters (currently the IP address, subnet mask and default

gateway address) for one LAN channel. Parameter sets for a specific slot are ordered; they are assigned to a board/module in the order in which the Chassis Manager polls LAN channels of a board/module (see the definition of the configuration variable *BOARD_LAN_PARAMETERS_CHANNEL_LIST*).

In general, parameters sets are retrieved by the Chassis Manager dynamically on demand. During initialization, after the Chassis FRU Information is found and the Chassis Address Table is known, the Chassis Manager retrieves parameter sets for all currently active boards and modules (in state M4) and performs the assignment. After initialization, when a new board or module installed in the chassis reaches state M4, the Chassis Manager retrieves the corresponding parameter sets for it and performs the assignment. This assignment is performed synchronously.

4.11.2 Obtaining LAN configuration parameters on the Chassis Manager level

There are three ways to obtain LAN configuration parameters on the Chassis Manager level:

1. Read from a flash file on the ChMM.
   After the Chassis Manager starts up, it retrieves the LAN configuration parameters from a file on the ChMM */var/nvdata/lan_config_params*. This is the simplest approach to retrieve the parameters.
   The file has a text format. Each line defines one parameter set and has the following format:
   `<physical-slot> "=" <ip-address> "," <subnet-mask> "," <default-gateway-ip-address>`
   Comment lines starting with # are allowed and ignored. Also, the rest of the line after a # character is treated as a comment and ignored. Empty lines are allowed and ignored, too.
   For each slot, the file can define several parameter sets. All parameter sets defined for the same slot must be placed in the file together and have the same values to the left of the equal sign. They are assigned to LAN channels on the corresponding boards or modules that require such assignments.

2. Read from the Chassis FRU Information
   In this case, there is a set of custom multirecords with the following Manufacturer ID (IANA Enterprise Number) assigned to nVent, Schroff GmbH (formerly Pentair Technical Products) 00400Ah[†] and Record ID = 0Ch. The concatenation of the data in these multirecords (optionally compressed with *gzip*) represents the parameter data in the same format as for the text file above. Management of parameter sets is also performed in the same way as for the previous case.
   An example of a LAN parameters configuration file is given below:

```
# Parameters for a module in physical slot 1
1 = 192.168.1.101, 255.255.255.0, 192.168.1.1
1 = 192.168.1.102, 255.255.255.0, 192.168.1.1
1 = 192.168.1.103, 255.255.255.0, 192.168.1.1

# Parameters for a module in physical slot 2
```

[†]This IANA Enterprise Number was originally assigned to Pigeon Point Systems, which has since been acquired by Pentair.

```
2 = 192.168.1.121, 255.255.255.0, 192.168.1.1
2 = 192.168.1.122, 255.255.255.0, 192.168.1.1
2 = 192.168.1.123, 255.255.255.0, 192.168.1.1
…
```

4.11.2.1 Using the FRU Info Compiler to place LAN configuration data in the Chassis FRU Information

The FRU Information Compiler implements special syntax that can be used to incorporate a LAN configuration parameters file into a FRU Information image that is being created.

For example, the following instructions add a compressed LAN configuration parameters file *lan_config_params.gz* to the current FRU Information image:

```
[nVent.Board/AMC LAN Configuration]
File = lan_config_params.gz
```

4.11.2.2 Using the Chassis Manager Command Line Interface to place LAN configuration parameters in the Chassis FRU Information

An extended version of the CLI command *frudataw* can be used to dynamically update Chassis FRU Information with LAN configuration information. The corresponding configuration file (optionally compressed) should first be downloaded to the ChMM.

The corresponding command line has the following syntax:

clia frudataw -l <ipmb-address> <fru-id> <in-file>

clia frudataw -l -c <ipmb-address> <fru-id>

The parameters *<ipmb-address>* and *<fru-id>* specify the IPMB address and FRU device ID of the FRU Information to update. For the Chassis FRU Information, use IPMB address 20h and FRU device ID 254.

The parameter *<in-file>* indicates the LAN configuration parameters file (which is possibly compressed) to be stored in the target FRU Information.

Option *-c* removes (clears) the LAN configuration data from the target FRU Information.

For example, the following command can be used to update the Chassis FRU Information with a new version of the LAN configuration parameters file:

```
#clia frudataw -l 20 254 lan_config_params.gz
```

4.11.2.3 Configuring a Linux DHCP server for obtaining LAN configuration parameters

The */etc/dhcpd.conf* file on the DHCP server should have a record with a unique DHCP client identifier string for each channel of each FRU for which the LAN configuration parameters need to be assigned. Unless otherwise documented explicitly for certain specific types of ATCA chassis, the format of the client identifier is as follows:

```
<dhcp-client-identifier> ::= <chassis-address in bytes> : <2F + board physical
number> : <FRU ID> : <channel number>
```

The example below is for a chassis with 2 board slots (with IPMB addresses 82h, 84h) and the chassis address “XXX” (in ASCII):

```
############## Board LAN Configuration Params #############

    # 0x82,0 0

    host board1_fru0_ch0 {
        option dhcp-client-identifier 58:58:58:30:0:0;
        fixed-address 192.168.1.202;
        option routers 192.168.1.253;
        option subnet-mask 255.255.255.0;
    }

    host board1_fru0_ch1 {
        option dhcp-client-identifier 58:58:58:30:0:1;
        fixed-address 192.168.1.202;
        option routers 192.168.1.253;
        option subnet-mask 255.255.255.0;
    }

    host board1_fru0_ch2 {
        option dhcp-client-identifier 58:58:58:30:0:2;
        fixed-address 192.168.1.202;
        option routers 192.168.1.253;
        option subnet-mask 255.255.255.0;
    }

    host board1_fru0_ch3 {
        option dhcp-client-identifier 58:58:58:30:0:3;
        fixed-address 192.168.1.202;
        option routers 192.168.1.253;
        option subnet-mask 255.255.255.0;
    }

    host board1_fru0_ch4 {
        option dhcp-client-identifier 58:58:58:30:0:4;
        fixed-address 192.168.1.202;
        option routers 192.168.1.253;
        option subnet-mask 255.255.255.0;
    }
```

```
    # 0x84,0 0

host board2_fru0_ch0 {
    option dhcp-client-identifier 58:58:58:31:0:0;
    fixed-address 192.168.1.203;
    option routers 192.168.1.253;
    option subnet-mask 255.255.255.0;
}

host board2_fru0_ch1 {
    option dhcp-client-identifier 58:58:58:31:0:1;
    fixed-address 192.168.1.203;
    option routers 192.168.1.253;
    option subnet-mask 255.255.255.0;
}

host board2_fru0_ch2 {
    option dhcp-client-identifier 58:58:58:31:0:2;
    fixed-address 192.168.1.203;
    option routers 192.168.1.253;
    option subnet-mask 255.255.255.0;
}

host board2_fru0_ch3 {
    option dhcp-client-identifier 58:58:58:31:0:3;
    fixed-address 192.168.1.203;
    option routers 192.168.1.253;
    option subnet-mask 255.255.255.0;
}

host board2_fru0_ch4 {
    option dhcp-client-identifier 58:58:58:31:0:4;
    fixed-address 192.168.1.203;
    option routers 192.168.1.253;
    option subnet-mask 255.255.255.0;
}
```

4.11.3 Dispatching LAN configuration parameters to boards

For each board or module, the Chassis Manager detects whether it supports LAN channels and expects the configuration parameters for them to be assigned by the Chassis Manager. If this is the case for a particular board or module, the Chassis Manager sends it the configuration parameters from the next parameter set for the corresponding physical slot. Chassis Manager does this for all boards or modules in states M4-M5 during startup (after the Chassis FRU Information has been found). After the initialization, Chassis Manager performs this procedure during board or module activation (when the module reaches state M4). The Chassis Manager maintains internal records to avoid sending configuration parameters to the same board or module twice.

To detect what LAN channels a specificboard or module has, the Chassis Manager sends it the command “Get Channel Info”, with a channel number picked sequentially from the *BOARD_LAN_PARAMETERS_CHANNEL_LIST*. If a board or module responds with the Channel Medium Type = 802.3 LAN for some channel, the Chassis Manager sends the command “Get LAN Configuration Parameters (IP Address Source)” for that channel. If the IP Address source = 3 (address loaded by BIOS or system software), the Chassis Manager assigns configuration parameters for this channel. Commands are sent on System IPMB.

Parameter sets are obtained in order as they are needed for a specific slot. If a parameter set cannot be obtained (for instance, if parameter sets are exhausted for a specific slot), configuration parameters are not assigned.

The Chassis Manager sends the configuration parameters to a board or module as a set of “Set LAN Configuration Parameters” commands, over System IPMB; however the channel parameter in these commands is set to channel N. For modules, commands are wrapped in “Send Message” commands. The following batch of “Set LAN Configuration Parameters” commands is sent:

- Channel = N, Parameter = Set In Progress (0), data = set in progress
- Channel = N, Parameter = IP Address (3), data = board IP address
- Channel = N, Parameter = Subnet Mask (6), data = board subnet mask
- Channel = N, Parameter = Default Gateway (12), data = board default gateway IP address
- Channel = N, Parameter = Set In Progress (0), data = commit write
- Channel = N, Parameter = Set In Progress (0), data = set complete

4.12 ChMM tests available via the diagnostic infrastructure

A Diagnostic Initiators (DI) facility has been implemented in the Chassis Manager. Currently, nVent OEM commands are used for access to the diagnostic initiators; they are documented in the *Pigeon Point Chassis Manager External Interface Reference*. The overall infrastructure is documented in the *IPM Controller Diagnostic Initiator Architecture* document.

The Diagnostic Initiators facility is implemented on the physical Chassis Manager and provides access to several run-time tests for the ChMM. All tests are associated with Diagnostic Initiator 0. Cancellation is not supported for any of the tests. Parameters for the tests are described in the

following table. The results string for most of these tests has the form “Status: XX”, where XX is a two-digit hexadecimal test-specific reason code (“00” if the test completes successfully).

For the I^2C test, the results string in the case of success is a list of 2-character hexadecimal addresses for I^2C devices present on the bus. For the IPMB test, the results string (in the case of both success and failure) is the list of extended IPMB state and error flags reported by the driver.

Some of the tests are not implemented on ChMM-700 because they don’t make sense for that platform; this is noted explicitly in the table below. However, to ensure compatible allocation of test identifiers between the Chassis Manager and the Shelf Manager, the corresponding tests are still described as present on the ChMM-700; when invoked, such tests report failure, with the textual result string: “This test is not implemented”.

Table 16: ChMM tests implemented in the diagnostic infrastructure

Test name	Test ID	Parameters (Name, Type, Description)	Test description
“UART INT” (Internal UART Loopback Test). Not implemented on ChMM-700.	0	“uart”: INT32: UART ID to test Acceptable values are 0 and 1. Default value is 0.	The UART RX and TX channels are internally looped back while the test is in progress, which may result in data loss for external communication on the affected UART. For example, if UART #0 is the system console, some console output may be lost or corrupted.
“UART EXT” (External UART Loopback Test). Not implemented on ChMM-700.	1	“uart”: INT32: UART ID to test Acceptable values are 0 and 1. Default value is 0.	This test requires manual interaction with the ChMM carrier to loop back the UART RX and TX pins.
“ETHER INT” (Internal Ethernet Loopback Test)	2	“mac”: INT32: MAC ID to test Acceptable values are 0 and 1. Default value is 0.	The RX and TX channels are internally looped back while the test is in progress, which may result in data loss and/or a broken network connection.

Test name	Test ID	Parameters (Name, Type, Description)	Test description
"ETHER PHY" (Ethernet PHY Loopback Test)	3	"mac": INT32: MAC ID to test Acceptable values are 0 and 1. Default value is 0.	The RX and TX channels are internally looped back within the PHY while the test is in progress, which may result in data loss and/or a broken network connection.
"ETHER ARP" (Ethernet ARP Test)	4	"mac": INT32: MAC ID to test Acceptable values are 0 and 1. Default value is 0. "ip": INT32: IP Address to resolve during the test. "timeout": INT32: Timeout value (in seconds) for the test.	The test sends an ARP request with the specified IP address and waits for a reply in the specified timeout range.
"FLASH CRC" (Flash Checksum Test)	5	"mtd": INT32: Software image number for the test. Acceptable values are: • 2 – Active Linux kernel • 3 – Active U-Boot • 4 – Active RFS • 7 – Inactive Linux kernel • 8 – Inactive U-Boot • 9 – Inactive RFS • 10 – Active application image (ChMM-700 only) • 11 - Inactive application image (ChMM-700 only)	The test calculates a checksum for the software image and compares it with the checksum in the image header.

Test name	Test ID	Parameters (Name, Type, Description)	Test description
"I2C TEST" (I^2C Bus Test)	6	"bus": INT32: I^2C bus number to test. Acceptable values are 0 and higher (bus 0 represents the main master-only I^2C bus going to the ChMM carrier (and often beyond the carrier to the chassis); other bus numbers are platform-specific). Default value is 0.	The test tries to access the specified I^2C adapter device and enumerate all I^2C devices present on that bus by reading from all possible I2C addresses on the bus and checking which addresses are acknowledged.
"IPMB TEST" (IPMB Bus Test)	7	"bus": INT32: IPMB bus number to test. Acceptable values are 0 and higher (0 represents IPMB-A, 1 represents IPMB-B, higher values represent radial IPMB links). Default value is 0.	The test queries the IPMB bus driver for extended information about the specified bus, which includes state flags and error flags. This information is then converted to a textual format and reported in the results string.

5 Using the IPMI analysis tools

The Chassis Manager software includes tools for analyzing the IPMI messaging traffic within a chassis. Categories of such traffic include IPMB messages going between the ChMM and other IPM Controllers in the chassis as well as IPMI traffic between the ChMM and external management applications via Ethernet. These tools consist of the following two parts:

- The IPMB trace collection daemon. This application runs on the ChMM and interfaces with the IPMB-specific extensions in the Monterey Linux / Pigeon Point Linux kernel on the ChMM to collect IPMB trace data from System IPMB. The collected data is either stored into a file in a local or NFS-mounted file system, or transferred directly to the GUI client on a separate platform.
- The IPMI trace analyzer. This role is typically played by a GUI application that supports viewing and analyzing an IPMB trace collected by the daemon, either from a file or directly from a ChMM via a TCP/IP connection. Wireshark and its terminal-oriented variant `tshark` are two specific applications that can play this role. In addition, Wireshark and `tshark` can also collect and analyze IPMI packets that are transferred over an Internet Protocol (IP) based network via RMCP or RMCP+, both of which are layered above UDP.

Note: these tools have not yet been updated to recognize and properly parse VITA 46.11 commands. They do however parse generic IPMI commands, which are used in VITA 46.11 systems, as well as PICMG commands. Examples of GUI and command-line output in this section show scenarios from AdvancedTCA shelves. Support for VITA 46.11 commands in Wireshark and `tshark` *is planned for a future release.*

5.1 In this section

This section contains the topics listed below. Just click on a topic to go to it.

- IPMB trace collection daemon (`ipmb_traced`)
- IPMI trace analyzer
- Collecting and analyzing a trace using the GUI and command line tools

5.2 IPMB trace collection daemon (ipmb_traced)

The IPMB trace daemon can operate in unattended trace mode or controlled trace mode.

In the unattended trace mode, the trace data is collected into a file in a ChMM-local or NFS-mounted file system. When the trace collection has been completed, the file can be transferred to the development host, then read and reviewed via the trace analyzer.

In the controlled trace mode, the trace data is transferred directly to the trace analyzer via a TCP/IP connection. The trace analyzer can display and analyze the collected data on a real time basis.

The IPMB trace collection daemon (*ipmb_traced*) runs on the ChMM and takes the role of IPMB trace collector.

The daemon interacts with the IPMB-specific extensions in the Linux kernel to capture IPMB traffic on System IPMB. Specifically, the "promiscuous mode" of I2C adapter operation is used to capture all messages on the IPMB. (Promiscuous mode configures the adapter to receive all IPMB messages on the bus, not just the messages that are addressed to or issued by the ChMM, itself. However promiscuous mode is not supported on ChMM-700. On ChMM-700, due to hardware limitations, adapters can only receive messages that are issued by or directed to a ChMM, itself, so promiscuous mode does not have its intended effect.)

A specific instance of the daemon can collect either local or remote IPMI messaging trace, but not both. Also, when collecting a remote trace, it can do so only from one IPM Controller in the chassis. To collect a remote trace from each of multiple IPM Controllers, multiple instance of the daemon can be started on the ChMM.

When collecting a local or remote trace, the trace collection daemon runs in either the unattended trace or controlled trace modes.

When initiated in unattended trace mode, the daemon initiates trace collection and starts writing collected data into a file. This continues until the daemon is terminated by the user via *^C* or the SIGINT signal is sent to the daemon process operating as a background Linux application. The resulting trace file can then be transferred to another host for analysis using the IPMB trace analyzer.

When started in controlled trace mode, the daemon begins to listen for an incoming TCP/IP connection from the IPMB analyzer running on another host. Once the connection is established, the daemon initiates trace collection and starts transferring collected data to the analyzer over TCP/IP. This continues until the GUI terminates the connection. The daemon can be terminated by the user via *^C* or the SIGINT signal sent to the daemon process operating as a background Linux application.

The daemon command line has the following syntax:

ipmb_traced [OPTIONS]

The options are described in the following table:

Table 17: Options for the IPMB trace collection daemon

OPTION NAME	DESCRIPTION
-u <file_name> | --unattended=<file_name>	Specify a file for collecting trace data in the unattended trace mode. By default the file name is */var/ipmb_traced/ipmb_traced.log*
-c | --controlled	Run daemon in the controlled trace mode
-e |	Trace IPMB state change events. Applies to

`--events`	local trace mode only.
`-d <usec> |` `--delay=<usec>`	Specify a delay in microseconds for polling the IPMB-A/IPMB-B interfaces. This option has an effect only if the `–e (--events)` option is enabled. If there are IPMB state change events, they are extracted and collected at intervals that are no longer than the defined delay, but always whenever an IPMB message is received. The delay value is specified in a range of 0 to 1000000 microseconds (1 second). If the `-d` option is omitted, the delay value defaults to 100000 microseconds (0.1 second). Applies to local trace mode only.
`-p <port>|` `--port=<port>`	In controlled trace mode, specify the TCP port number to listen to; the client application connects to this port. Default port = 19000.
`-q <size>|` `--quota=<size>`	In unattended trace mode, the maximum trace file size (in KB). After this size is reached, the existing file is closed and renamed with the extension `.old` and a new trace file is created.
`-m <mask>|` `--mask=<mask>`	In board trace mode, specify the bit mask of IPMI channels that are to be traced on the target IPMC. This bit mask is represented by the numeric parameter `<mask>` (with prefix 0x used to specify a hexadecimal number). The following channel numbers can be used: • Channel 0 (bit mask 1) = System IPMB. • Channel 7 (bit mask 0x80) = IPMB-L. • Channel 15 (bit mask 0x8000) = System (Payload) Interface. The default value is 1 (trace System IPMB only).
`-i <number>|` `--instance=<number>`	In board trace mode, specify the payload instance number to use for establishing the RMCP+ session. This number must match the IPMI Trace Payload instance number used by the target IPMC for IPMB tracing; refer to the

	board documentation for this number. The default value is *1*.
-U <username>| --username=<username>	In board trace mode, specify the RMCP+ user name for establishing the RMCP+ session with the board. The default user name is *rmcp*. Passwords are not used.
-r [<number>]| --radial[=<number>]	Enable radial IPMB link monitoring. Without this option, *ipmb_traced* doesn't monitor any radial IPMB links in the shelf. Optional *<number>* parameter allows limiting the number of device nodes to open. The default is 17, which is a cross-link plus 16 links for each board in the chassis.
-h | --help	Display help text and exit

If no options are provided, *ipmb_traced* runs in unattended trace mode, produces trace data in Wireshark format and puts them in the */var/ipmb_traced/ipmb_traced.log* file, but does not trace IPMB state change events, monitor radial IPMB links. This scenario assumes that a separate ChMM application, such as the Pigeon Point Chassis Manager, assigns IPMB addresses for the ChMM and attaches to the ChMM IPMB interface so that there can be IPMB traffic to trace.

The trace collection daemon stores the collected data in a file (or sends it directly to the analyzer) in the form of packets. Each packet describes a single IPMB transaction or event and contains the following fields.

Table 18: IPMB trace collection packet field

FIELD NAME	DESCRIPTION
Seconds	Transaction completion time in seconds
Micro seconds	Transaction completion time in microseconds (calculated from the start of the last second)
Event	IPMB interface state change information
Flags	Special flags returned by the Linux kernel driver
Bus	IPMB interface (IPMB-A or IPMB-B) on which the transaction was captured
Address	Destination I^2C address of the transaction
Size	Size of the transaction
Data	Transaction raw data

5.3 IPMI trace analyzer

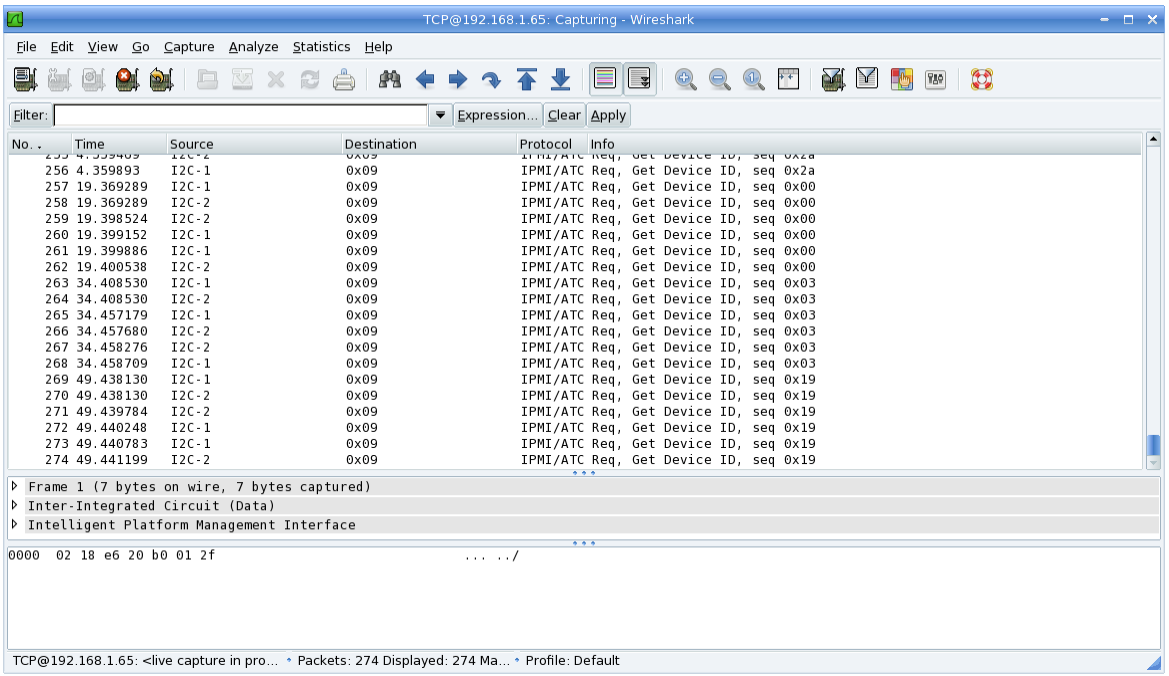
The recommended IPMI trace analyzer for Pigeon Point Chassis Manager applications is a network traffic analyzer program called Wireshark (see www.wireshark.org), with its terminal-oriented variant, *tshark*. Wireshark provides a convenient interface for a complete analysis of layered protocol stack data. The Wireshark design allows developers to easily add support for new protocols, entire protocol stacks, and trace collection data sources in a collaborative manner. There are versions of Wireshark for both Windows and Linux environments. All these considerations combine to make Wireshark a great framework for IPMI trace analyzer applications.

The following sections describe the key user interfaces of Wireshark that are relevant to IPMI trace analysis. Please consult the extensive user documentation for Wireshark at www.wireshark.org for more details.

5.3.1 Introduction to the Wireshark GUI

This section provides a brief introduction to the GUI; section 5.4 provides a step by step example of using it. Figure 6 provides a view of the GUI main window, which is composed of three frames.

Figure 6: Main window (IPMB trace analysis)



The upper frame shows a list of captured messages that provides basic information about each IPMB message. The list can be sorted by the contents of each field as a key.

In an IPMB trace analyzer usage, this frame displays the full list of collected IPMB messages. For each message, the following basic information is provided:

Table 19: IPMB message list basic information

GUI FRAME FIELD	INTERPRETATION

GUI FRAME FIELD	INTERPRETATION
No.	The index (number) of the message in the trace
Time	Message date and time stamp, displayed as an offset from the first packet. The time stamp obtained on the target is available in the detailed packet information in the bottom frame.
Source	The IPMB the message was captured on. This field is either *I2C-1* for *IPMB-A* or *I2C-2* for *IPMB-B*
Destination	Destination I^2C address of the message. (in 7-bit format). This field is "-" for IPMB events.
Protocol	The type of the captured message. This field can be one of the following: *IPMI/ATCA* for an IPMI/ATCA formatted message *EVENT* for an IPMB event *I2C READ* or *I2C WRITE* for a raw I^2C transaction
Info	Basic information about the captured message or event

The middle frame provides the protocol break down and data interpretation for a message selected in the message list provided in the upper frame.

In an IPMB trace analyzer usage, this frame provides interpretation of the layered contents of the message. Three protocol layers are supported:

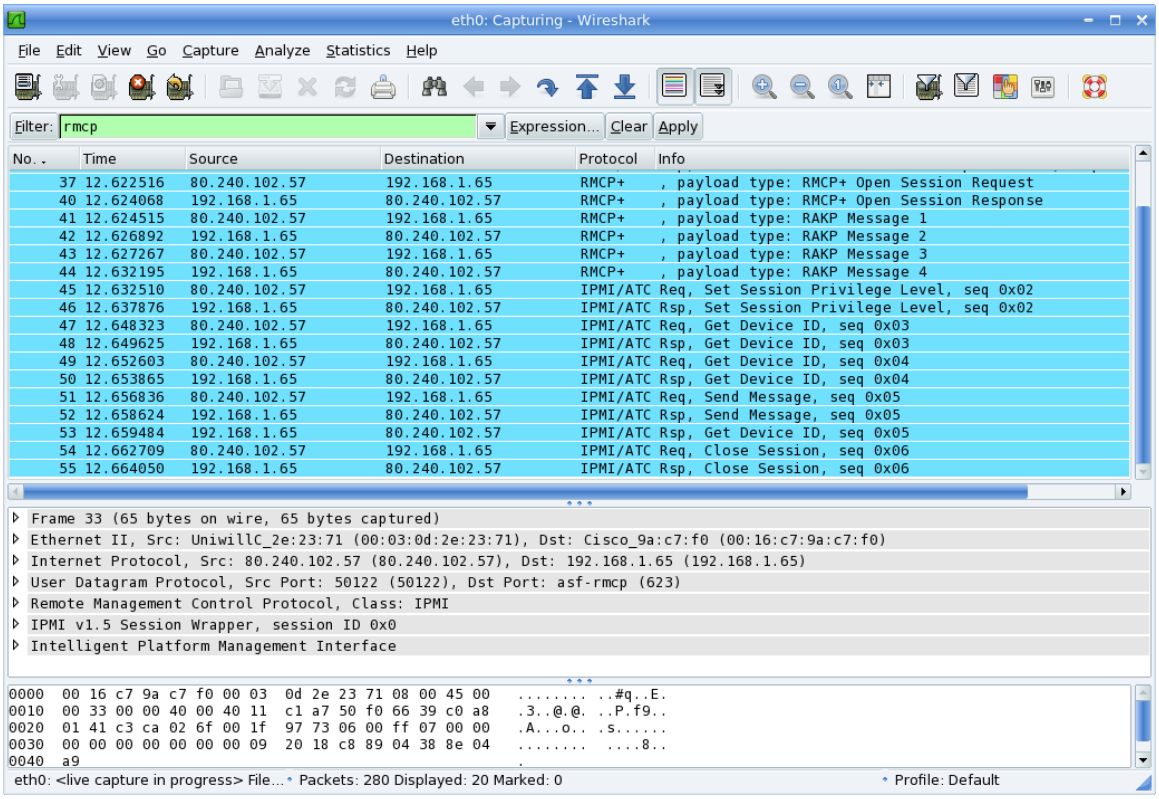
- Layer 1 provides low-level information about a message, such as the frame size, arrival time relative to the start of the trace, and the frame protocol (i2c:ipmi for IPMI messages).
- Layer 2 includes physical System IPMB and I^2C information, such as the bus the message was captured on (IPMB-A or IPMB-B) and the destination I^2C address of the frame.
- Layer 3 is available only for IPMI/ATCA messages, and provides message header information and information about the request/response data. Specifically, the message is identified as either a request or a reply, and the request/reply header fields are interpreted. The data fields are interpreted as defined by IPMI and PICMG specs for a particular request or response.

The bottom frame provides interpreted, hexadecimal and textual representations of a message selected in the message list provided in the upper frame.

The GUI provides a number of other GUI elements (dialogs, menu, windows, etc.) that are used to exercise various features of the IPMI Analyzer as well as customize its behavior. For instance, there is a trace collection dialog, trace filter window, and others. These facilities are described in detail in the Wireshark user documentation and man pages. Section 5.4, later in this document, provides a step by step procedure to use selected GUI elements for tracing and analyzing IPMB data.

When analyzing RMCP or RMCP+ based IPMI network traffic, the main GUI window looks similar, but with some small differences (as shown in Figure 7 below).

Figure 7: Main window (IPMI over network trace analysis)



Instead of I^{2}C-related information, the Source and Destination columns show the source and destination IP addresses of the captured message. Instead of the three protocol layers in the case of an IPMB trace, seven protocol layers are shown:

- Layer 1 provides low-level information about a message, such as the frame size, arrival time relative to the start of the trace, and the list of protocols used in the frame.
- Layer 2 provides Ethernet-specific information about the frame (source and destination MAC addresses and the protocol type).
- Layer 3 provides IP-specific information from the frame.
- Layer 4 provides UDP-specific information from the frame.

- Layer 5 provides RMCP header information from the frame (version, sequence number and class).
- Layer 6 includes additional information about the RMCP or RMCP+ specific fields in the message.
- Layer 7 is available only for IPMI/ATCA messages, and provides message header information and information about the request/response data. Specifically, the message is identified as either a request or a reply, and the request/reply header fields are interpreted. The data fields are interpreted as defined by IPMI and PICMG specs for a particular request or response.

5.3.2 Introduction to terminal-oriented Wireshark

The Wireshark tool suite also includes a command line-oriented utility called `tshark`. This utility can be used to analyze traffic using only a command line approach, when running a GUI application is not practical. The `tshark` tool has the same overall functionality as the GUI, but lacks its interactive capabilities.

Section 5.4 also includes step by step examples of using `tshark`, which is fully documented in the corresponding manual pages (`man tshark`). To get online help, use `tshark -?`.

5.3.3 Installing IPMI analyzer software

The IPMB trace daemon is included in the software image on the ChMM by default and does not require any specific installation.

The Wireshark client applications (and much more regarding Wireshark) can be obtained from www.wireshark.org.

The `netcat` (`nc`) utility is needed as a mediator between the IPMB trace daemon and Wireshark/tshark for real-time IPMI analysis (with IPMB trace daemon operating in controlled mode). Linux distributions typically include this utility. Its Windows variant (`nc.exe`) can be downloaded from http://www.securityfocus.com/tools/139. The source code of the utility is available at http://netcat.sourceforge.net/download.php.

5.4 Collecting and analyzing a trace using the GUI and command line tools

An IPMB trace can be collected in either of two ways: 1) unattended mode (where it is stored in a file during collection and analyzed later) or 2) controlled mode (where the trace is forwarded to directly to the analyzer as it is collected). The following sections provide guidance for each of these modes, as well as for collecting and analyzing an IPMI over network trace.

5.4.1 Collecting IPMI traces on System IPMB in unattended mode

In unattended mode, a trace is collected into a target-local file, transferred to the analyzer host, and then read and analyzed using the trace analyzer.

The following stepwise procedure shows how to collect an unattended trace and open it for analysis in the GUI. The GUI-specific steps are followed by corresponding guidance for opening the trace for analysis in *tshark*:

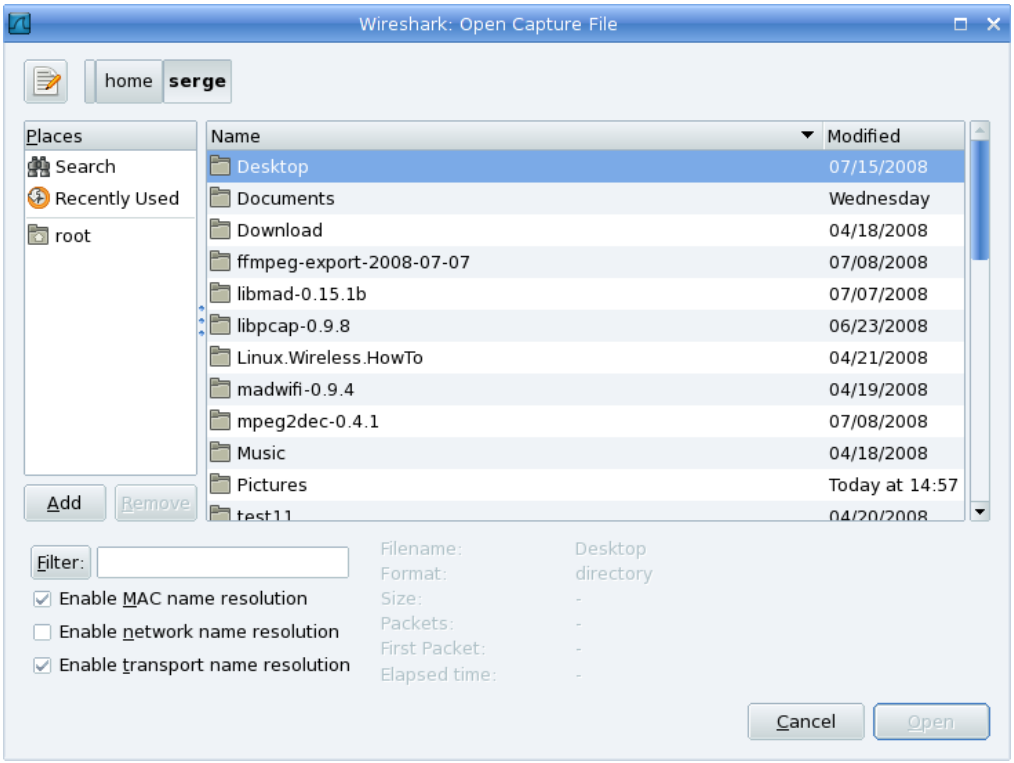
1. Start *ipmb_traced* on the ChMM with the option *-w*, optionally specifying the file to put the trace to:
   # **ipmb_traced –w -u /var/ipmb_traced.log &**
   If no *-u* option is provided, *ipmb_traced* puts the trace in the following file: */var/ipmb_traced/ipmb_traced.log*. In this case, the */var/ipmb_traced* directory must be created manually using the *mkdir* command before *ipmb_traced* is started:
   # **mkdir /var/ipmb_traced**
2. Execute actions that create the IPMI traffic on System IPMB to be analyzed. For example, if insertion of a board generates the sequence of events and commands that need to be analyzed, do such a board insertion.
3. When trace collection is complete, stop *ipmb_traced* by sending the SIGINT signal to *ipmb_traced*:
   - Obtain the ipmb_traced process ID using the *ps* command:

     ```
     # ps
     ...
        39 ttyS0   root        0   S   ipmb_traced –w -u /var/ipmb_traced.log
     ...
     ```
   - Send the SIGINT signal to the *ipmb_traced* process:

     ```
     # kill -2 39
     ```
4. Copy the trace file to the analyzer host:

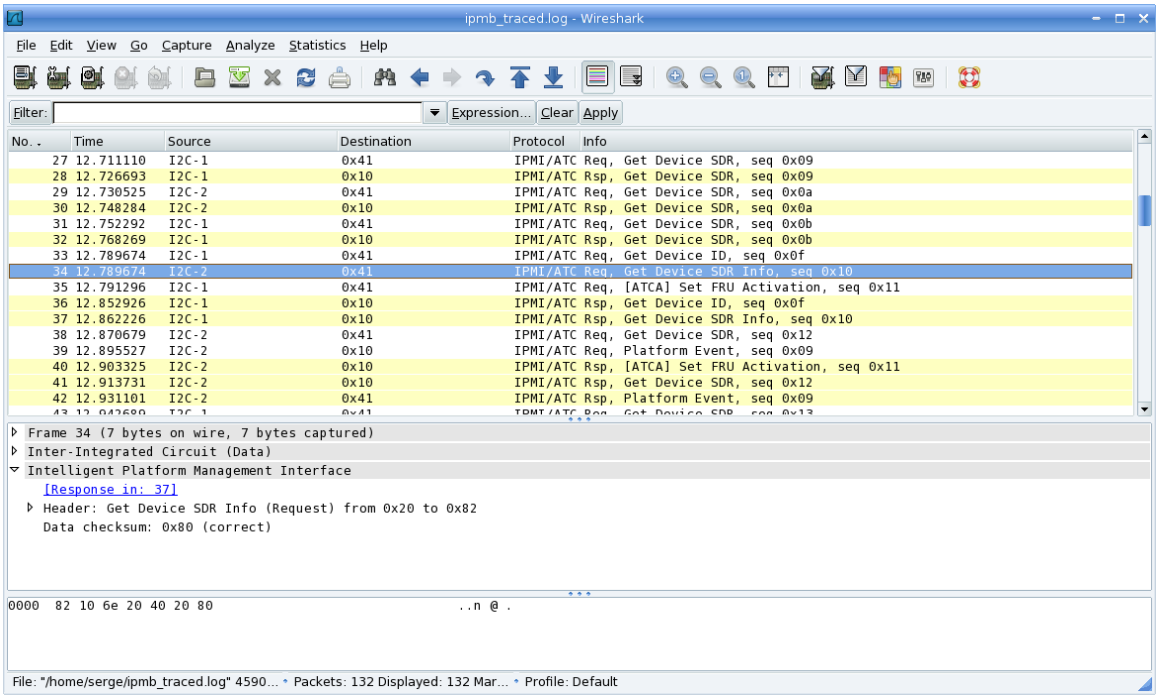
   ```
   # cd /var/
   # ftp 172.16.1.24
   Connected to 172.16.1.24.
   220 ts FTP server (Version wu-2.6.2-5) ready.
   ...
   ftp> put ipmb_traced.log
   ```
5. On the analyzer host, start the GUI:
   C:\> **wireshark**
6. In the main menu, go to *File*, then *Open* and browse to the trace file. Open the trace file:

Figure 8: Open Capture File window



This will open the trace in the GUI main window:

Figure 9: GUI Main window (unattended mode)



In a command-line context, at step 5, instead of starting the GUI utility, run the *tshark* utility (option *–r* specifies the trace data file name):

```
# tshark -r ipmb_traced.log
  1   0.000000        I2C-1 -> 0x41         IPMI/ATCA Req, Get Sensor Reading,
seq 0x28
  2   0.079507        I2C-1 -> 0x10         IPMI/ATCA Rsp, Get Sensor Reading,
seq 0x28
  3   0.579642        I2C-2 -> 0x41         IPMI/ATCA Req, Get Sensor Reading,
seq 0x29
  4   0.585695        I2C-2 -> 0x10         IPMI/ATCA Rsp, Get Sensor Reading,
seq 0x29
  5   0.679566        I2C-1 -> 0x41         IPMI/ATCA Req, Get Sensor Reading,
seq 0x2a
  6   0.686105        I2C-1 -> 0x10         IPMI/ATCA Rsp, Get Sensor Reading,
seq 0x2a
  7   0.698566        I2C-2 -> 0x41         IPMI/ATCA Req, Get Sensor Reading,
seq 0x2b
  8   0.713453        I2C-2 -> 0x10         IPMI/ATCA Rsp, Get Sensor Reading,
seq 0x2b
  9   0.724908        I2C-1 -> 0x41         IPMI/ATCA Req, Get Sensor Reading,
seq 0x2c
 10   0.739889        I2C-1 -> 0x10         IPMI/ATCA Rsp, Get Sensor Reading,
seq 0x2c
```

```
 11   0.751980        I2C-2 -> 0x41         IPMI/ATCA Req, Get Sensor Reading,
seq 0x2d
 12   0.767020        I2C-2 -> 0x10         IPMI/ATCA Rsp, Get Sensor Reading,
seq 0x2d
…
```

Use the *–V* option to obtain a detailed dump of the messages; use *–x* option to obtain a hexadecimal dump of messages, as well:

```
# tshark –r ipmb_traced.log –x -V
Frame 1 (8 bytes on wire, 8 bytes captured)
    Arrival Time: Jun 24, 2008 20:58:03.238567000
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 8 bytes
    Capture Length: 8 bytes
    [Frame is marked: False]
    [Protocols in frame: i2c:ipmi]
Inter-Integrated Circuit (Data)
    Bus: I2C-1
    Target address: 0x41
    Flags: 0x00000000
Intelligent Platform Management Interface
    [No corresponding response]
    Header: Get Sensor Reading (Request) from 0x20 to 0x82
        Target Address: 0x82
        Target LUN: 0x00, NetFN: Sensor/Event Request (0x04)
            .... ..00 = Target LUN: 0x00
            0001 00.. = NetFn: Sensor/Event Request (0x04)
        Header checksum: 0x6e (correct)
        Source Address: 0x20
        Source LUN: 0x00, SeqNo: 0x28
            .... ..00 = Source LUN: 0x00
            1010 00.. = Sequence Number: 0x28
        Command: Get Sensor Reading (0x2d)
    Data
        Sensor Number: 2
    Data checksum: 0x11 (correct)
0000  82 10 6e 20 a0 2d 02 11                            ..n .-..
```

```
Frame 2 (11 bytes on wire, 11 bytes captured)
    Arrival Time: Jun 24, 2008 20:58:03.318074000
    [Time delta from previous captured frame: 0.079507000 seconds]
    [Time delta from previous displayed frame: 0.079507000 seconds]
    [Time since reference or first frame: 0.079507000 seconds]
    Frame Number: 2
    Frame Length: 11 bytes
    Capture Length: 11 bytes
    [Frame is marked: False]
    [Protocols in frame: i2c:ipmi]
Inter-Integrated Circuit (Data)
    Bus: I2C-1
    Target address: 0x10
    Flags: 0x00000000
Intelligent Platform Management Interface
    [Response to: 1]
    [Responded in: 0.079507000 seconds]
    Header: Get Sensor Reading (Response) from 0x82 to 0x20
        Target Address: 0x20
        Target LUN: 0x00, NetFN: Sensor/Event Response (0x05)
            .... ..00 = Target LUN: 0x00
            0001 01.. = NetFn: Sensor/Event Response (0x05)
        Header checksum: 0xcc (correct)
        Source Address: 0x82
        Source LUN: 0x00, SeqNo: 0x28
            .... ..00 = Source LUN: 0x00
            1010 00.. = Sequence Number: 0x28
        Command: Get Sensor Reading (0x2d)
        Completion code: Command Completed Normally (0x00)
    Data
        Sensor Reading: 50
        Event Messages: Enabled, Sensor scanning: Enabled
            1... .... = Event Messages: Enabled
            .1.. .... = Sensor scanning: Enabled
            ..0. .... = Reading/status unavailable: False
        Threshold comparisons/assertions (byte 0)
            0... .... = Reserved / State 7 asserted: False
            .0.. .... = Reserved / State 6 asserted: False
            ..0. .... = At or above UNR threshold / State 5 asserted: False
            ...0 .... = At or above UC threshold / State 4 asserted: False
```

```
            .... 0... = At or above UNC threshold / State 3 asserted: False
            .... .0.. = At or below LNR threshold / State 2 asserted: False
            .... ..0. = At or below LC threshold / State 1 asserted: False
            .... ...0 = At or below LNC threshold / State 0 asserted: False
    Data checksum: 0xbf (correct)
0000  20 14 cc 82 a0 2d 00 32 c0 00 bf                    ....-.2...
Frame 3 (8 bytes on wire, 8 bytes captured)
    Arrival Time: Jun 24, 2008 20:58:03.818209000
    [Time delta from previous captured frame: 0.500135000 seconds]
    [Time delta from previous displayed frame: 0.500135000 seconds]
    [Time since reference or first frame: 0.579642000 seconds]
    Frame Number: 3
    Frame Length: 8 bytes
    Capture Length: 8 bytes
    [Frame is marked: False]
    [Protocols in frame: i2c:ipmi]
Inter-Integrated Circuit (Data)
    Bus: I2C-2
    Target address: 0x41
    Flags: 0x00000000
Intelligent Platform Management Interface
    [No corresponding response]
    Header: Get Sensor Reading (Request) from 0x20 to 0x82
        Target Address: 0x82
        Target LUN: 0x00, NetFN: Sensor/Event Request (0x04)
            .... ..00 = Target LUN: 0x00
            0001 00.. = NetFn: Sensor/Event Request (0x04)
        Header checksum: 0x6e (correct)
        Source Address: 0x20
        Source LUN: 0x00, SeqNo: 0x29
            .... ..00 = Source LUN: 0x00
            1010 01.. = Sequence Number: 0x29
        Command: Get Sensor Reading (0x2d)
    Data
        Sensor Number: 4
    Data checksum: 0x0b (correct)

0000  82 10 6e 20 a4 2d 04 0b                             ..n .-..
…
```

5.4.2 Collecting IPMI traces on System IPMB in controlled mode

When the trace daemon is run in the controlled mode, the trace is transferred directly to the trace analyzer via a TCP/IP connection. The following stepwise procedure shows how to collect a controlled trace and open it for analysis in the GUI. The GUI-specific steps are followed by corresponding guidance for collecting a controlled trace in *tshark*:

1. On the ChMM, start *ipmb_traced* with the *-c* and *–w* flags:

   ```
   # daemon -f ipmb_traced –c -w
   ```

   On the analyzer host, use *netcat* to feed the data to Wireshark over the TCP/IP network (specify the IP address of the ChMM and the port on which the trace daemon listens for incoming connections, 19000 by default):

   ```
   C:\> nc.exe 192.168.1.62 19000 | "c:\Program Files\Wireshark\wireshark.exe" –k –i -
   ```

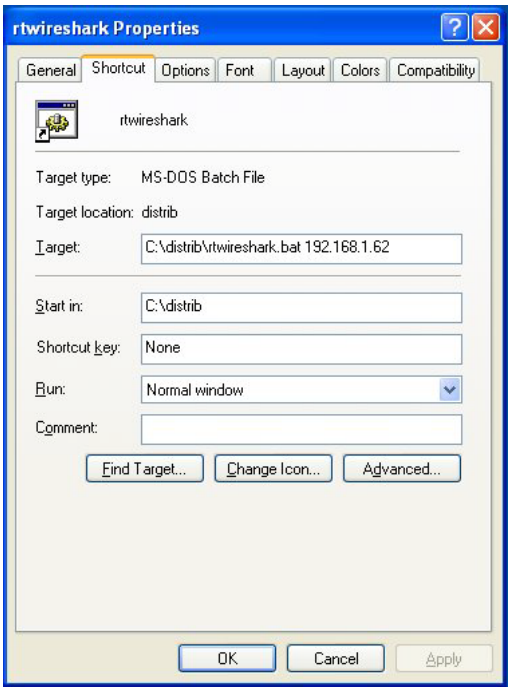
   The *-k* option instructs Wireshark to start capturing data immediately. The standard input is specified as the interface that provides the trace data. To avoid repeatedly typing the long command, a Windows batch script can be created instead:

   [rtwireshark.bat]

   ```
   C:\distrib\nc.exe %1 19000 | "%ProgramFiles%\Wireshark\wireshark.exe" -k -i -
   & exit
   ```

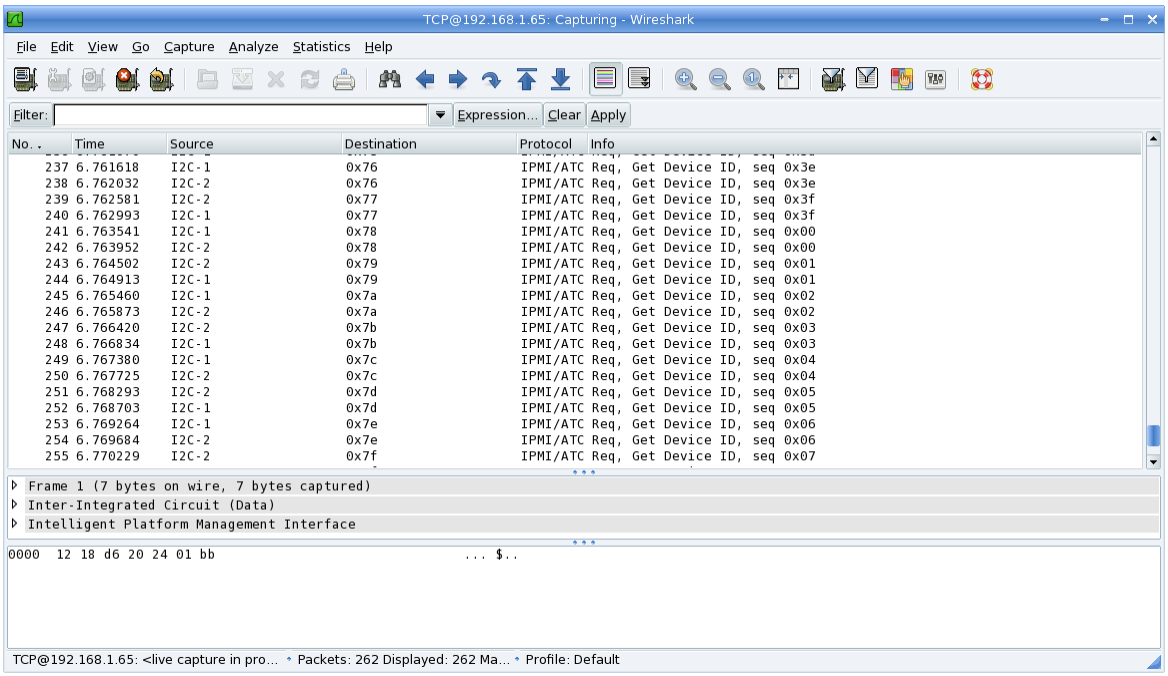
   Given that this script is defined, invoke it from the command prompt or create a shortcut on the desktop and specify the IP address as an input parameter:

   Figure 10: Capture Options window



2. Execute actions that create the IPMI traffic on System IPMB to be analyzed. For example, if insertion of a board generates the sequence of events and commands that need to be analyzed, do such a board insertion.
3. The GUI will update the main window showing the collected trace in real-time:

Figure 11: Collected trace (controlled mode)



4. To stop the capture session, go to the *Capture* submenu and select *Stop*.

   In a command-line context, redirect the output of *netcat* to the *tshark* utility instead of the Wireshark GUI application. Like Wireshark, *tshark* can take the standard input option, *-*, to indicate the source of the data. Specify the IP address of the ChMM and the port on which the trace daemon listens for incoming connections (19000 by default) as input to *netcat*. Options *–x* and *-V*, as in the case of unattended mode, can be used to request a hexadecimal and/or detailed dumps of the messages. Any additional options needed for filtering and analysis of the message stream (see section 5.4.5) should also be specified at this time.

   Next, execute the actions from step 5; the trace is then captured by *tshark* and sent to the command output. For example:

```
# nc 192.168.1.62 19000 | tshark –i -
Capturing on Standard input
1214325501.948358        I2C-1 -> 0x41         IPMI/ATCA Req, Get Sensor
Reading, seq 0x27
1214325501.999629        I2C-1 -> 0x10         IPMI/ATCA Rsp, Get Sensor
Reading, seq 0x27
1214325502.002796        I2C-2 -> 0x41         IPMI/ATCA Req, Get Sensor
Reading, seq 0x28
```

```
1214325502.017845        I2C-2 -> 0x10         IPMI/ATCA Rsp, Get Sensor
Reading, seq 0x28
1214325502.038082        I2C-1 -> 0x41         IPMI/ATCA Req, Get Sensor
Reading, seq 0x29
1214325502.038245        I2C-1 -> 0x10         IPMI/ATCA Rsp, Get Sensor
Reading, seq 0x29
1214325502.041232        I2C-2 -> 0x41         IPMI/ATCA Req, Get Sensor
Reading, seq 0x2a
…
```

To stop the capture session, terminate the *tshark* process (e.g. by pressing *^C*).

5.4.3 Collecting traces of IPMI traffic over the network

When collecting the traces of IPMI traffic transferred over the network RMCP or RMCP+ sessions, the trace daemon is not used. Wireshark or *tshark* collects the trace directly from one of the local network interfaces on the system on which that tool runs (that is, the analyzer host). Usually, the analyzer host is also the one that participates in the IPMI-oriented communication. However, if the analyzer host's local network interface supports promiscuous mode, Wireshark or *tshark* can collect a trace of IPMI communication between two separate systems that are not the analyzer host, if at least one of them is on the same LAN with the analyzer host.

The following stepwise procedure shows how to collect an IPMI-over-network trace and open it for analysis in the GUI. The GUI-specific steps are followed by corresponding guidance for collecting a similar trace in *tshark*.

1. On the analyzer host, start the GUI:

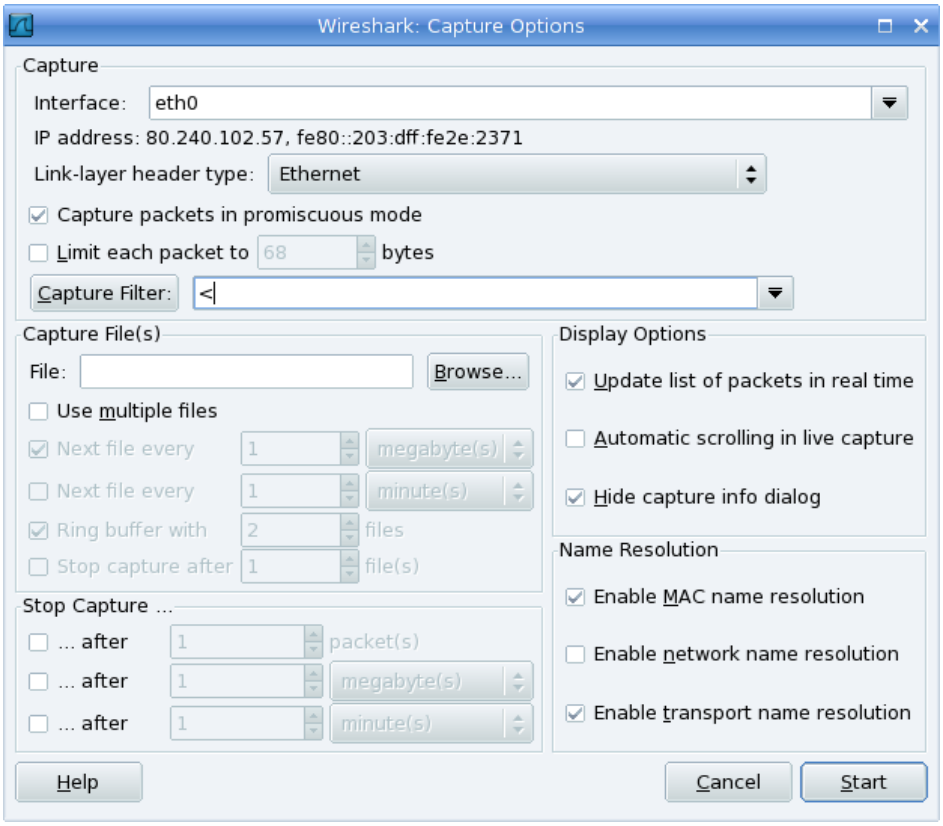
   ```
   C:\> wireshark
   ```

   In the main menu, go to *Capture* submenu and select *Options…*. This opens the *Capture Options* dialog. In the dialog, use the *Capture/Interface* window to specify the target. Choose the name of the network interface through which the IPMI communication takes place (e.g. *eth0*).

   Additionally, you can use the *Update list of packets in real time* checkbox to specify that the GUI should show the trace in real-time.

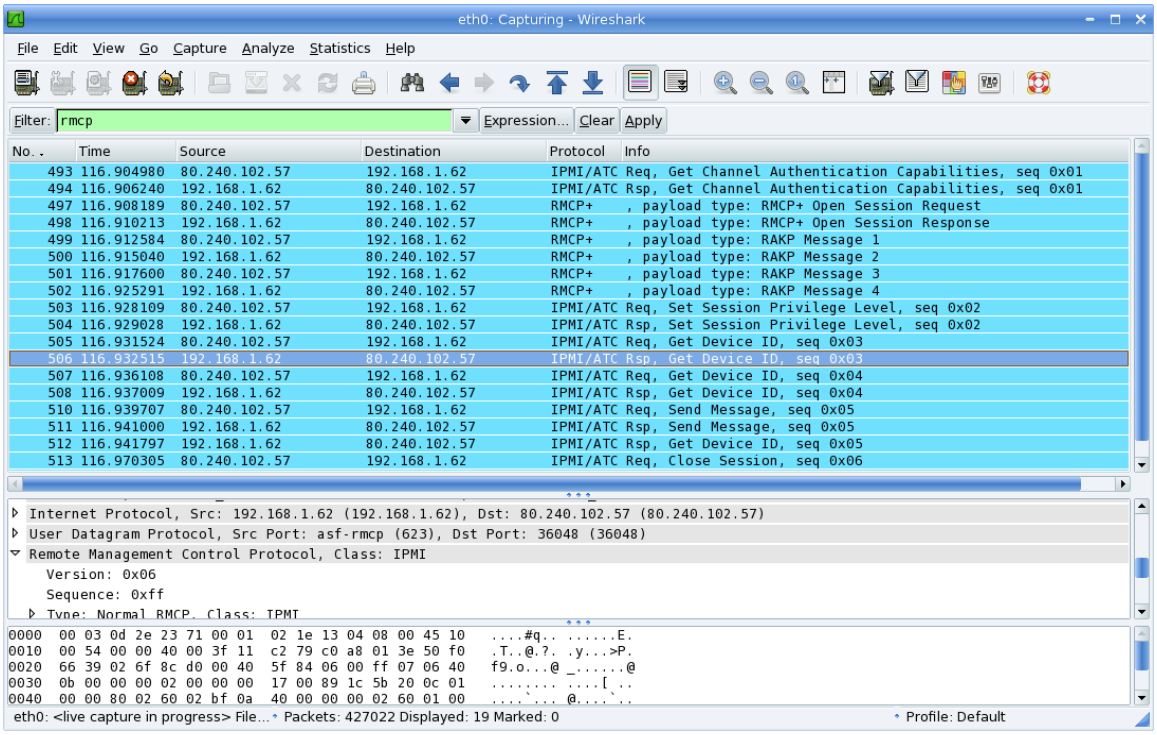
   For instance:

Figure 12: Capture Options window (analyzing networked IPMI traffic)



2. Press the *Start* button to begin collecting the trace.
3. Set up the filter “rmcp” to include only RMCP-related traffic into the trace. Otherwise, all network traffic going through the specified adapter will be included into the trace. To do this, type *rmcp* in the Filter window in the top-left corner of the main screen and press the *Apply* button.
4. Execute actions that create the IPMI traffic on the network to be analyzed. For example, run a system management application that uses RMCP/RMCP+ to talk to the Chassis Manager.
5. The GUI will update the main window showing the collected trace in real-time:

Figure 13: Collected trace (networked IPMI traffic)



6. To stop the capture session, go to the *Capture* submenu and select *Stop*.

In a command-line context, instead of steps 1-3, run the *tshark* utility (specifying the target in the form of the network interface name (e.g. *eth0*) using option *–i)*. Specify the filtering option *–R rmcp* to include only RMCP traffic in the trace. Options *–x* and *-V*, as in the case of unattended IPMB trace mode, can be used to request a hexadecimal and/or detailed dumps of the messages.

Any additional options needed for filtering and analysis of the message stream (see section 5.4.5) should also be specified at this time. Use the *&&* (“logical and”) operator to combine additional filtering options with the filtering option *rmcp*.

Next, execute the actions from step 4; the trace is then captured by *tshark* and sent to the command output. For example:

```
# tshark –i eth0–R rmcp
71.989290 80.240.102.57 -> 192.168.1.199 IPMI/ATCA Req, Get Channel
Authentication Capabilities, seq 0x0
 71.990809 192.168.1.199 -> 80.240.102.57 IPMI/ATCA Rsp, Get Channel
Authentication Capabilities, seq 0x01
 71.996590 80.240.102.57 -> 192.168.1.199 RMCP+ , payload type: RMCP+ Open
Session Reques
 71.998471 192.168.1.199 -> 80.240.102.57 RMCP+ , payload type: RMCP+ Open
Session Respons
 72.000184 80.240.102.57 -> 192.168.1.199 RMCP+ , payload type: RAKP Message 1
```

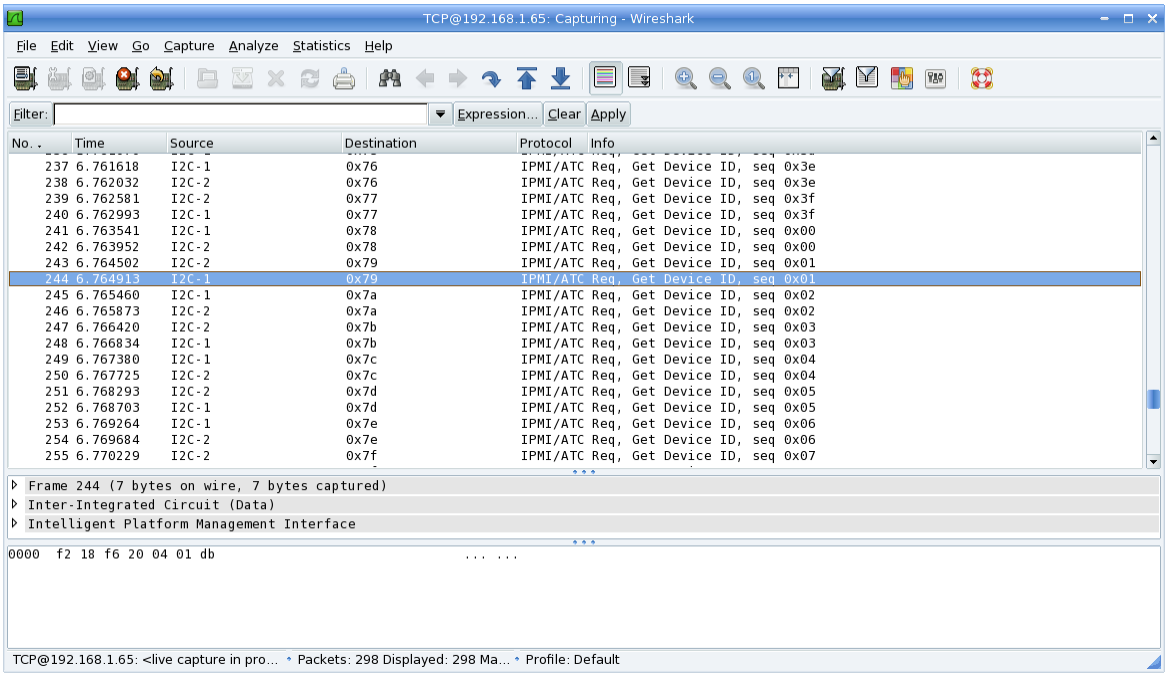
```
 72.005487 192.168.1.199 -> 80.240.102.57 RMCP+ , payload type: RAKP Message
 72.008539 80.240.102.57 -> 192.168.1.199 RMCP+ , payload type: RAKP Message 3
 72.019149 192.168.1.199 -> 80.240.102.57 RMCP+ , payload type: RAKP Message 4
 72.022416 80.240.102.57 -> 192.168.1.199 IPMI/ATCA Req, Set Session Privilege
Level, seq 0x02
 72.033355 192.168.1.199 -> 80.240.102.57 IPMI/ATCA Rsp, Set Session Privilege
Level, seq 0x02
 72.057400 80.240.102.57 -> 192.168.1.199 IPMI/ATCA Req, Get Device ID, seq 0x03
 72.059134 192.168.1.199 -> 80.240.102.57 IPMI/ATCA Rsp, Get Device ID, seq 0x03
 72.059568 80.240.102.57 -> 192.168.1.199 IPMI/ATCA Req, Get Device ID, seq 0x04
 72.062264 192.168.1.199 -> 80.240.102.57 IPMI/ATCA Rsp, Get Device ID, seq 0x04
 72.062680 80.240.102.57 -> 192.168.1.199 IPMI/ATCA Req, Send Message, seq 0x05
 72.066428 192.168.1.199 -> 80.240.102.57 IPMI/ATCA Rsp, Send Message, seq 0x05
 72.068235 192.168.1.199 -> 80.240.102.57 IPMI/ATCA Rsp, Get Device ID, seq 0x05
 72.101030 80.240.102.57 -> 192.168.1.199 IPMI/ATCA Req, Close Session, seq 0x06
 72.102654 192.168.1.199 -> 80.240.102.57 IPMI/ATCA Rsp, Close Session, seq 0x06
```

To stop the capture session, terminate the *tshark* process (e.g. by pressing *^C*).

5.4.4 Analyzing a trace using the GUI

To select a message in the Wireshark GUI main window, simply scroll to the message that interests you in the upper frame of the main window:

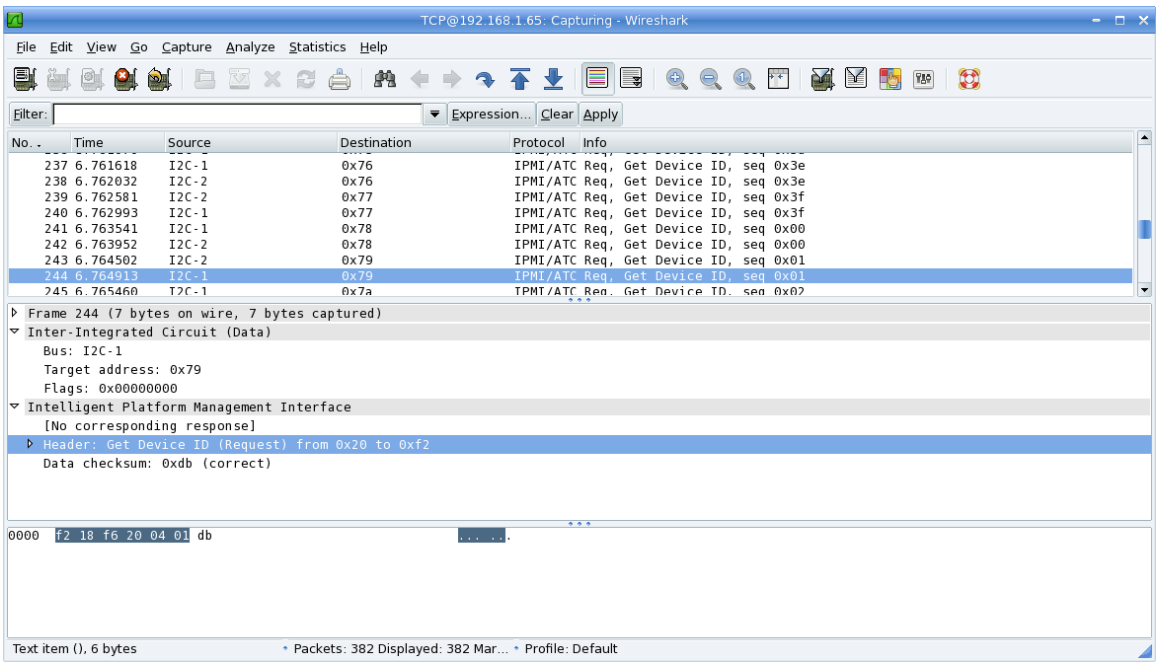
Figure 14: Message 244 is selected



After a message has been selected, its raw contents are shown in the bottom frame of the main window, both in hex and ASCII interpretations. The middle frame of the main window provides protocol interpretations of the message data.

By clicking on the right arrow/down arrow (or +/-) GUI elements, you can choose a particular protocol layer you wish to analyze:

Figure 15: IPMI protocol layers (IPMB trace)



A trace can be filtered using the filter expressions. Specifically, for IPMI trace filtering, the following expression primitives are available:

Table 20: IPMI filter primitives

FILTER PRIMITIVE NAME	DESCRIPTION
ipmi.header.broadcast	Broadcast destination address (only for Broadcast Get Device ID Request messages)
ipmi.header.target	Destination device (responder or requester) slave address
ipmi.header.netfn	Network Function (NetFn) of the message
ipmi.header.trg_lun	Destination (responder or requester) Logical Unit Number (LUN)
ipmi.header.crc	Header checksum
ipmi.header.source	Source device (requester or responder)

FILTER PRIMITIVE NAME	DESCRIPTION
	slave address
ipmi.header.sequence	Request sequence
ipmi.header.src_lun	Source (requester or responder) LUN
ipmi.header.command	Command to execute
ipmi.data.completion	Completion code (only for response messages)
ipmi.data.crc	Data checksum

The filtering expression primitives above are applicable for both IPMB traffic and RMCP/RMCP+ IPMI traffic over the network. In the latter case, several RMCP filtering expression primitives are also available, but these are not widely used.

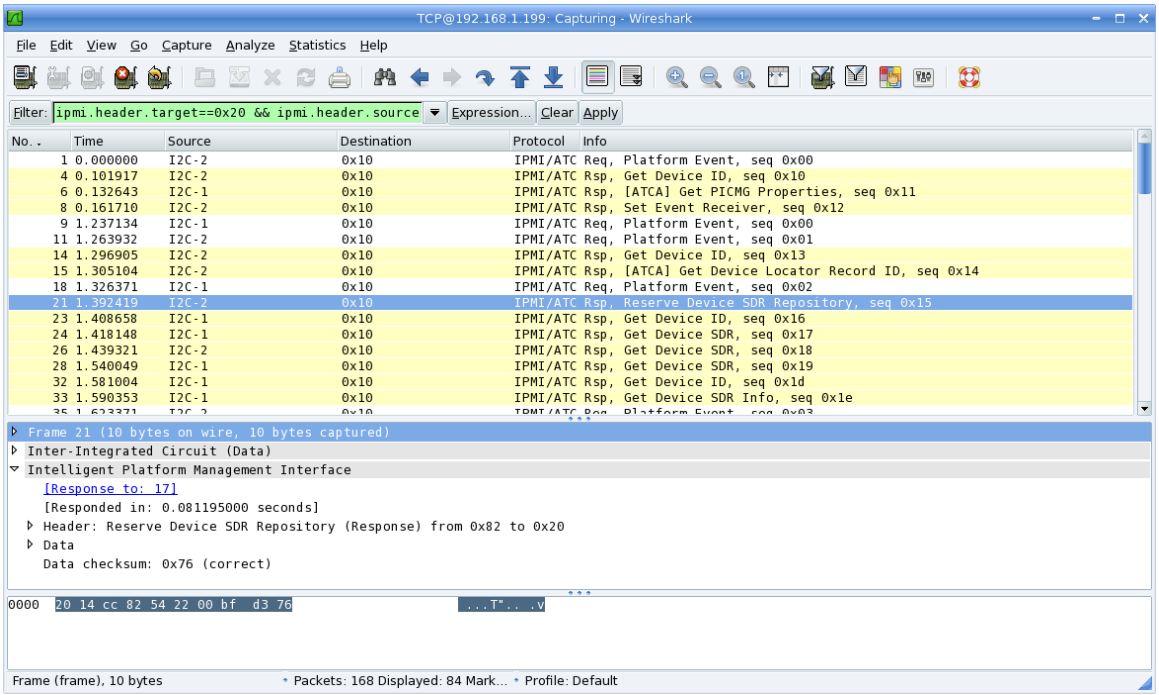
To specify a filter, use the *Filter* text entry box in the upper left corner of the main window.

For instance, to select the messages with the destination address 0x20 and the source address 0x82, type the string:

ipmi.header.target == 0x20 && ipmi.header.source == 0x82

in the *Filter* text entry box:

Figure 16: Filtered trace



Note: Since the broadcast destination address is always zero, the `ipmi.header.broadcast` *filter expression should be used by specifying the following filter string in the* `Filter` *entry of the main window:*

```
ipmi.header.broadcast == 0x00
```

5.4.5 Analyzing a trace using command-line tools

GUI tools are much more powerful for analyzing a trace than command-line tools, so you should use GUI tools for trace analysis, if possible. When using command-line tools, you should first capture the data into a file in unattended mode and then analyze the captured data file using *tshark*. In this usage style, *tshark* can be run multiple times over the same data, thus allowing some degree of interactivity in the analysis. In controlled mode, *tshark* is run only once over the captured data; as a result, all applicable options must be specified in the command line at startup.

Another limitation of *tshark*, compared to Wireshark, is that *tshark* can only provide back references for the packets (the links *Response to* and *Response in* that allow the user to navigate across packets quickly). This limitation results from the fact that *tshark* is optimized for speed and thus calls the packet dissector only once per packet. The verbose *tshark* capture in section 5.4.1 shows an example of this limitation. The description of the first packet indicates *No corresponding message*, while the description of the second packet shows *Response to: 1*.

To filter messages in *tshark* (both in controlled and unattended mode), use the *–R* option, followed by the text of the filter expression (in quotes). For example, to select and dump in details the messages with the destination address 0x20 and the source address 0x82 (as in the previous section), when reading from a trace file, use the following command line:

```
# tshark -r ipmb_traced.log -V -R "ipmi.header.target==0x20 &&
ipmi.header.source==0x82"
```

The *–T* option can be used to specify the format of the output. The following formats are supported: plain text, PostScript, PDML (Packet Details Markup Language, based on XML), PSML (Packet Summary Markup Language, based on XML) and so called "field format". For example, the following command line can be used to dump message details in the PostScript format:

```
# tshark -r ipmb_traced.log -V –T ps
```

The *–e* option can be used to select specific packet fields to be dumped; if this option is used, the "field format" must be also specified (*-T fields*). Several *–e* options can be used to specify multiple fields. In conjunction with the filtering option *–R*, this option provides a powerful tool for queries on the message stream, similar to relational database queries. For example, the following query dumps the values of net functions and command codes for IPMB messages sent from the source address 0x84 to the Chassis Manager (destination address 0x20):

```
# tshark -r ipmb_traced.log -e ipmi.header.netfn -e ipmi.header.command
"ipmi.header.target==0x84 && ipmi.header.source==0x20" -T fields
0x04    0x2d
```

```
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x06    0x01
0x04    0x2d
0x04    0x20
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
0x04    0x2d
```

Full information about *tshark* options and syntax can be found in the *tshark* manual page (*man tshark*).

6 Re-initializing the ChMM

6.1 In this section

This section applies contains the topics listed below. Just click on a topic to go to it.

- Re-initializing the U-Boot environment
- Re-initializing the file system
- Resetting the login password

6.2 Re-initializing the U-Boot environment

The U-Boot environment variables are stored in two redundant dedicated Flash partitions on the ChMM-700. If you would like to restore the default values for the U-Boot environment variables, you must first erase the environment variables stored in EEPROM/Flash and reset (or power cycle) the ChMM.

To erase the U-Boot environment variables on a ChMM-700, you need to enter the following commands from the U-Boot prompt:

```
ChMM700 sf probe 2:0
65536 KiB N25Q512A at 2:0 is now current device
ChMM700 sf erase 100000 80000
Erasing Flash... done!
ChMM700
```

Then you need to reset the ChMM using the *reset* command, as shown below with resulting console output

```
ChMM700 reset
resetting ...
U-Boot 2009.08 (Oct 24 2017 - 11:44:23)

Freescale i.MX28 family
CPU:   297 MHz
BUS:   99 MHz
EMI:   130 MHz
GPMI:  24 MHz
I2C:   ready
DRAM:  128 MB
SFGEN: S25FL512S detected, total size 64 MB
A2F:   SPICOMM protocol v1.7, M3 firmware v1.8, FPGA design v1.1.0.0
A2F:   A2F firmware, version 1.8
A2F:   Device type: A2F060M3E-FG256
A2F:   MSS clock frequency: 40 MHz
```

```
A2F:   Fabric clock frequency: 20 MHz
A2F:   Fast delay calibration: 1330 cycles per 100uS
A2F:   eNVM: 128 KB (00000000 - 00020000)
A2F:   eSRAM: 16 KB (20000000 - 20004000)
A2F:   Extram start: 20002750
A2F:   Last reset cause: SOFT
A2F:   VMON flags: 00000000
RUPG:  booting from image 1 (confirmed)
In:    serial
Out:   serial
Err:   serial
Net:   FEC0: 00:18:49:02:b2:06, FEC1: 00:18:49:02:b2:07
FEC0, FEC1
Hit any key to stop autoboot:  0
ChMM700
```

Then save these environment settings. Since U-Boot configuration storage contains 2 copies of environment variables data and the *saveenv* command changes them in round-robin order, you need to invoke the *saveenv* command twice to update both copies:

```
ChMM700 saveenv; saveenv; reset
```

6.3 Re-initializing the file system

The filesystem (*/var* and */etc*) is stored within the Flash and can be restored quite easily from the filesystem in the current RFS image. U-Boot has an environment variable called *flash_reset*. By setting this variable to *y* and then booting up the system, the */var* and */etc* files are erased and then restored from the filesystem in the current RFS image.

```
ChMM700 setenv flash_reset y
ChMM700 saveenv
ChMM700 boot
```

The *flash_reset* variable is automatically set to *n* at system startup after re-initializing the Flash. The boot command (the second command above) begins booting the Linux kernel. It is during this process that the file system is re-initialized. The following output is shown on the console.

```
/etc/rc: Mount ubi0:user
/etc/rc: Flash erase requested via U-Boot variable
/etc/rc: erasing /etc
/etc/rc: erasing /var
/etc/rc: Selecting carrier from the environment:
/etc/rc:
/etc/rc: Extracting shelfman rootfs patch from /1/sentry.chmm700.app
/etc/rc: Extraction result :  0
```

```
/etc/rc: Placed /var/tmp to ram disk
/etc/rc: Setting hostname chmm700
/etc/rc: Started syslogd and klogd
/etc/rc: Strobing the reliable upgrade WDT
/etc/rc: /etc appears to be empty ... restoring from RFS /etc...
```

6.4 Resetting the login password

The default login for the ChMM, as set in the factory-provided RFS images, is a user ID of “root” and the empty password “”. A chassis supplier can change the default password. We highly encourage users to change the password when configuring the Chassis Manager. In the event that the new password is forgotten, the password can be reset to the default value, as assigned in the RFS image, via the *`password_reset`* U-Boot variable. By setting this variable to *y* and then booting up the system, the root password is removed.

```
ChMM700 setenv password_reset y
ChMM700 saveenv
ChMM700 boot
```

The following output is shown on the console, during boot up.

```
/etc/rc: hostname demo
```

7 Re-programming the ChMM-700

This section describes how to update the firmware on the ChMM-700, using the reliable upgrade facilities.

7.1 In this section

This section contains the topics listed below. Just click on a topic to go to it.

- Firmware reliable upgrade procedure overview
- Flash partitioning and image layout
- Changing the 'user' flash partition format to JFFS2
- Reliable upgrade procedure status file
- Reliable upgrade utility
- Reliable upgrade of SmartFusion cSoC image
- Reliable upgrade utility use scenarios
- Reliable upgrade examples

7.2 Firmware reliable upgrade procedure overview

WARNING: Before downgrading the ChMM-700 Linux kernel, RFS and U-Boot images from a 1.1.0 or later Chassis Manager release to a pre-1.1.0 Chassis Manager release, users **MUST** verify that the Flash 'user' partition is formatted to UBIFS, for further details refer to section 7.4.

For the ChMM-700, a reliable upgrade procedure for the firmware images on a running and functioning ChMM is provided. An arbitrary combination of the following four software images can be upgraded reliably: the U-Boot firmware, the Linux kernel, the Linux root file system (RFS) and the application package. If an attempted reliable upgrade fails, an automatic rollback to the previous version of the firmware takes place. An upgrade failure can be caused, for example, by installation of a faulty kernel image that cannot properly boot up the ChMM, or a faulty Chassis Manager application that fails to start.

The ChMM-700 firmware, in addition to the U-boot, kernel and RFS, includes a fourth component that contains just the application binaries for the Chassis Manager and accompanying applications; in many cases, it will be sufficient to upgrade only this component in order to upgrade to the next version of the Chassis Manager, which should result in faster upgrades. On the other hand, the RFS image on the ChMM-700 contains only system files and executables, and does not contain application binaries and application-specific files, which makes it smaller and less prone to change in successive versions of the ChMM-700 firmware.

Two sets of upgrade component images exist on the ChMM-700. Only the U-Boot images are assigned to dedicated Flash partitions. Kernel and RFS images are stored as files on a single partition, formatted under the UBIFS file system. U-Boot on ChMM-700 is able to read files from

UBIFS partitions and therefore is able to locate and load the proper kernel and RFS images. Also, there exist two instances of the application component package, one of which represents the candidate firmware and the other of which represents the confirmed firmware. They are stored on another (user-accessible) UBIFS Flash partition, in a compressed form. When the RFS is loaded and the initial startup script (`/etc/rc`) is executed, it picks up the appropriate instance of the application package, and unpacks it into certain directories in the root RAMFS file system, thus creating application binaries in the RAMFS file system on the fly.

Also, there are two instances of writable directories `/etc` and `/var` on the ChMM-700, so that each of the two image sets has its own instance of these directories. These directories contain writable application data. When the corresponding image set is loaded, the corresponding instance of `/etc` and `/var` directory is found on the Flash and mounted as `/etc` and `/var` in the current file tree.

When a stable firmware configuration is established in one of these image sets, it is designated as the confirmed image set. When a new firmware configuration is installed, it goes in the other image set, which is initially designated candidate. Once a new firmware configuration in the candidate set is validated, that image set is designated the confirmed image set and continues to be uses until a future upgrade cycle starts the process over again.

ChMM reliable upgrades are enabled by hardware mechanisms which ensure that if defective firmware is present in the candidate image set and the candidate image set fails to boot or initialize, an automatic rollback takes place to the safe software copy in the confirmed image set. Even if the candidate image set boots and initializes successfully, but embedded software integrity checks run at the application level during initialization indicate that the new software image is not fully functional, hardware mechanisms can be invoked to reset the ChMM and revert to the safe confirmed image set.

At a higher level, the reliable upgrade utility uses these hardware mechanisms and also implements software mechanisms for carrying out reliable upgrades properly. For the benefit of the user, the progress of the current reliable upgrade session and its final outcome (success or failure) is logged in a special file (`/var/upgrade/status`) located on the User UBIFS partition.

The reliable upgrade procedure implemented on ChMM-700 was designed primarily to handle reliable upgrades of the Pigeon Point Chassis Manager firmware. In addition, however, the procedure can be applied to arbitrary embedded applications running on the ChMM. To achieve that, the procedure supports “hook” scripts that can be invoked at certain stages of the reliable upgrade procedure and can be used for customization of the procedure. This support for hook scripts is briefly discussed in section 7.6; however, the rest of this Section 7 mainly concentrates on reliable upgrades in the context of Chassis Manager firmware, where hook scripts are not used.

Also, a special procedure is defined for the reliable upgrade of the SmartFusion cSoC image. The Microsemi SmartFusion cSoC is a hardware component of the ChMM-700 that includes an FPGA fabric and an ARM Cortex-M3 subsystem, combined on a single chip. Correspondingly, the image for the SmartFusion cSoC consists of an FPGA fabric image and the program code for the Cortex-M3 processor, combined in a single composite image. Update of the SmartFusion cSoC image

requires special precautions and is implemented as a special case of the reliable upgrade procedure. This special case is documented in detail in section 7.7. It is expected, however, that the SmartFusion cSoC image will be relatively stable, and that most new releases of the Chassis Manager in future will not require an upgrade of the SmartFusion cSoC.

7.3 Flash partitioning and image layout

The ChMM-700 provides a hardware mechanism that allows swapping of the two Flash partitions at certain addresses under control of software running on the i.MX287. This capability is implemented in support of the reliable upgrade procedure and applies to U-Boot images, stored on dedicated Flash partitions. Other firmware images (the Linux kernel, RFS and application package) are stored in files on UBIFS volumes, and there are two sets of these images (numbered 0 and 1). The choice of the set to use is made in U-Boot based on the chosen instance number evaluated by the U-Boot. Therefore, a hardware mechanism is needed only for choosing the U-Boot instance and exposing the chosen instance number to the software.

To accelerate loading of images from a UBIFS partition, U-Boot implements a special cache where it stores the map of the last booted Linux kernel and RFS images. The map describes how the images are laid out in terms of physical Flash sectors, and using this map allows U-Boot, during the second and subsequent boots of the same image set, to load images using direct Flash read operations and to avoid expensive mounting of UBIFS volumes. This information is stored on a dedicated Flash partition. It is invalidated explicitly by the reliable upgrade utility before writing a new image set during a reliable upgrade.

The U-Boot instance number and the number of the chosen set of images is calculated by the following formula:

Number = UBOOT_IMAGE_SEL ^ Candidate

where:

“UBOOT_IMAGE_SEL” is the ChMM-700 soft jumper, the value of which is the number of the current confirmed U-Boot image and other firmware images (0/1)

“Candidate” is a flag that has a value of 1 if a reliable upgrade is currently in progress and the candidate set of images should be loaded and 0 otherwise.

“^” indicates the Exclusive-OR logical operation

The following table provides a summary of the Flash partitions maintained by Pigeon Point Linux for the ChMM-700.

Table 21: Flash partitioning for ChMM-700s (64MB Flash) in UBIFS mode

Offset in Flash (in MBytes)	Size (in MBytes)	Device node	Mounted as (on startup)	Content
0	0.5	/dev/mtd0	Not mounted	U-Boot image 0
0.5	0.5	/dev/mtd2	Not mounted	U-Boot image 1

Offset in Flash (in MBytes)	Size (in MBytes)	Device node	Mounted as (on startup)	Content
1	0.25	/dev/mtd4	Not mounted	U-Boot environment variables, first copy
1.25	0.25	/dev/mtd6	Not mounted	U-Boot environment variables, second copy
1.5	0.25	/dev/mtd8	Not mounted	UBIFS cache (used for boot acceleration)
1.75	30.25	/dev/mtd10	Not mounted (temporarily mounted by U-Boot to load images and by the reliable upgrade utility to write images)	Boot partition: UBI with one UBIFS volume (named 'boot'); contains Linux kernel and RFS images
32	32	/dev/mtd12	/0, /1	User partition: UBI with one UBIFS volume (named 'user'); contains application package images, Chassis Manager files, other application files and user data

The following table lists names of the files and directories on UBIFS volumes that contain other firmware images. The file names for the SmartFusion cSoC upgrade files are also listed; the SmartFusion cSoC upgrade process is described in section 7.7.

Table 22: File names for firmware images stored as UBIFS files

File name	Flash partition	Description
uImage.0	Boot	Linux kernel image 0
uImage.1	Boot	Linux kernel image 1
rfs.0	Boot	Root file system (RFS) image 0

File name	Flash partition	Description
rfs.1	Boot	Root file system (RFS) image 1
a2f-upgrade.dat	Boot	SmartFusion (A2F) cSoC upgrade image (present only during an upgrade of the SmartFusion cSoC)
a2f-auto-rollback.dat	Boot	Rollback SmartFusion cSoC upgrade image (for use if a SmartFusion cSoC upgrade fails for any reason)
a2f-manual-rollback.dat	Boot	Previous SmartFusion cSoC upgrade image for a manual rollback
/0/etc	User	/etc directory: instance 0; mounted as /etc if image set 0 is active
/1/etc	User	/etc directory: instance 1; mounted as /etc if image set 1 is active
/0/var	User	/var directory: instance 0; mounted as /var if image set 0 is active
/1/var	User	/var directory: instance 1; mounted as /var if image set 1 is active
/0/sentry.chmm700.app	User	Application package image 0. Starting with release 1.1.0, this file, if image set 0 is active, is a symbolic link to the app.0 file in the Boot partition.
/1/sentry.chmm700.app	User	Application package image 1. Starting with release 1.1.0, this file, if image set 1 is active, is a symbolic link to the app.1 file in the Boot partition.

7.4 Changing the 'user' flash partition format to JFFS2

Starting with release 1.1.0, it is possible to change the format of the 'user' partition from UBIFS to JFFS2 and back. This enhancement was added due to a report of very rare cases of the UBIFS partition becoming unrecoverable after a power cycle during a write operation on Flash. There is no known solution to this reported problem in the UBIFS driver, so the ability to switch to a more reliable filesystem for the 'user' partition has been added. The risk of a 'boot' partition corruption is much lower, since it is mounted in read-only mode except for a short upgrade time that takes about a minute.

Corruption of UBIFS-formatted 'user' partition has the following implications:

- The Chassis Manager Application images for both firmware image sets are located in the 'user' partition, so even if the partition is re-formatted after corruption, the Chassis Manager Application images are lost and have to be restored manually.
- The upgrade status files are located in the 'user' partition, so in case of corruption they are lost.
- The 'user' partition filesystem is not separated between firmware image sets, so corruption leads to the loss of `/var` and `/etc` files for both image sets. Also, it is possible to fill the entire 'user' partition while working with one image set (for example when the `ipmb_traced` utility is configured to save IPMB trace in a file in flash without the size restriction). In that case, an upgrade that attempts to preserve all `/var` and `/etc` contents will not work, due to the lack of file system space.

All these issues are resolved with the JFFS2 conversion enhancement. The Chassis Manager Application images, which are included in the application package, are now located in the safe 'boot' partition; the old locations remain as symbolic links. The 'user' partition is split into 3 parts in JFFS2 mode, so that there are separate partitions for each image set. In addition, there is a partition for the upgrade status files with the minimum possible size. The following table provides a summary of the Flash partitions for the ChMM-700 working in the JFFS2 mode.

IMPORTANT NOTICE

JFFS2 support for the ChMM-700 is available starting with release 1.1.0 and affects U-Boot, Linux kernel and RFS images. If the ChMM-700 is configured to use JFFS2 and it is necessary to downgrade it to a pre-1.1.0 release, there are two possible ways:

- Downgrade only the APP image, preserving JFFS2-capable U-Boot, Linux kernel and RFS images from release 1.1.0 or above.
- Change the 'user' partition format from JFFS2 to UBIFS before the downgrade, so that it is possible to downgrade any of the firmware components.

Table 23: Flash partitioning for ChMM-700s (64MB Flash) in JFFS2 mode

Offset in Flash (in MBytes)	Size (in MBytes)	Device node	Mounted as (on startup)	Content
0	0.5	/dev/mtd0	Not mounted	U-Boot image 0
0.5	0.5	/dev/mtd2	Not mounted	U-Boot image 1
1	0.25	/dev/mtd4	Not mounted	U-Boot environment variables, first copy

Offset in Flash (in MBytes)	Size (in MBytes)	Device node	Mounted as (on startup)	Content
1.25	0.25	/dev/mtd6	Not mounted	U-Boot environment variables, second copy
1.5	0.25	/dev/mtd8	Not mounted	UBIFS cache (used for boot acceleration)
1.75	30.25	/dev/mtd10	Not mounted (temporarily mounted by U-Boot to load images and by the reliable upgrade utility to write images)	Boot partition: UBI with one UBIFS volume (named 'boot'); contains Linux kernel and RFS images
32	15.25	/dev/mtdblock6	/0	JFFS2 User partition for image set 0; contains Chassis Manager files and user data.
47.25	15.25	/dev/mtdblock7	/1	JFFS2 User partition for image set 1; contains Chassis Manager files and user data.
62.5	1.5	/dev/mtdblock8	/var/upgrade	Partition for the upgrade status files. The minimum possible size for a JFFS2 partition is 5 erase blocks.

Before changing the 'user' partition format to JFFS2 it is necessary to verify that the ChMM-700 is running at least the Linux kernel, RFS and U-Boot images from release 1.1.0, or a later release, which can be done via the `imls` command. If the currently running Chassis Manager release includes the Linux kernel image, the RFS image or the U-Boot image from a release older than 1.1.0 (i.e. the Created date is prior to March 2015), it is necessary to upgrade the image(s) to release 1.1.0 or later, before changing the 'user' partition format to JFFS2.

To change the 'user' partition format to JFFS2, it is necessary to run the following commands.

```
# setenv user_partition jffs2 ; setenv flash_reset user ; reboot
```

WARNING: All files stored in the 'user' partition will be erased when the format is changed to JFFS2 (or from JFFS2 to UBIFS). The default files from the currently loaded RFS image will be copied to the 'user' partition in, for example, the */var* and */etc* directories.

The result can be verified in the *mount* command output at the ChMM shell prompt e.g.

```
# mount
  …
  ubi0:boot on /boot type ubifs (ro,sync,relatime,bulk_read)
  /dev/mtdblock7 on /var type jffs2 (rw,relatime)
  /dev/mtdblock8 on /var/upgrade type jffs2 (rw,relatime)
  /dev/mtdblock7 on /etc type jffs2 (rw,relatime)
```

In case it is necessary to downgrade the Linux kernel image, the RFS image or the U-Boot image to a release older than 1.1.0 that does not support JFFS2, the 'user' partition format must be changed to UBIFS before the downgrade. This can be done via the following commands.

```
# setenv user_partition ubifs ; setenv flash_reset user ; reboot
```

The result can be verified in the *mount* command output at the ChMM shell prompt e.g.

```
# mount
  …
  ubi1:boot on /boot type ubifs (ro,sync,relatime,bulk_read)
  ubi0:user on /var type ubifs (rw,sync,relatime,bulk_read)
  ubi0:user on /var/upgrade type ubifs (rw,sync,relatime,bulk_read)
  ubi0:user on /etc type ubifs (rw,sync,relatime,bulk_read)
```

7.5 Reliable upgrade procedure status file

The special file */var/upgrade/status* that is located on the User Flash partition contains the status of the most recent reliable upgrade session, that is, the session that is currently in progress or most recently completed.

/var/upgrade/status is a text file and consists of multiple lines; each line describes the status of a particular stage in the upgrade procedure. The format of a record line is as follows:

```
<stage>: <status> : <description>
```

where

- *<stage>* is an integer in the range 1 to 14 (where stage 14 is the final stage of a reliable upgrade session, meaning that the procedure is complete)
- *<status>* is a human-readable string that describes the status ("passed" or "failed") of the corresponding stage
- *<description>* is a description of the corresponding stage. The list of stages and the corresponding description of each stage is summarized in Table 24: below.

The status file can be used to track the progress of the reliable upgrade procedure; it is also used by the reliable upgrade utility itself to synchronize the state of the upgrade procedure across ChMM reboots.

Table 24: Reliable upgrade stages

STAGE NUMBER	DESCRIPTION
1	Images have been downloaded to /tmp and verified
2	Boot volume mount is about to begin
3	Boot volume mount is complete
4	Candidate has been marked as non-valid
5	U-Boot image write is complete
6	Kernel image write is complete
7	RFS image write is complete
8	Boot volume umount is complete
9	Application package has been removed
10	New application package write is complete
11	Sync() is complete
12	/var and /etc have been processed
13	First strobe after reboot has been done
14	Upgrade is confirmed

7.6 Reliable upgrade utility

A special user-space utility *rupgrade* is provided on the ChMM. This utility has several modes of operation and allows the user to perform multiple tasks, including reliable upgrades of various ChMM software components, status checks for the most recent upgrade, and manual rollbacks to the previous software versions.

Another utility *backend* is provided that is not intended to be invoked directly by an interactive user. This utility performs more specific actions needed for reliable upgrade, like starting, strobing and stopping the reliable upgrade watchdog timer, getting and setting the number of the

candidate and confirmed image set, activation of the candidate image set, etc. It can be invoked from system scripts, like the system startup script `/etc/rc`.

Also, the `rupgrade` utility uses the services provided by the `backend` utility during its operation.

The `rupgrade` utility is fully functional only when called from the superuser (root) account. If run from a non-root account, the utility can be used for read-only operations (like getting the upgrade status).

As the `rupgrade` utility operates in upgrade mode (see below), it logs the status of each action as it progresses through the stages of the procedure to the status file `/var/upgrade/status`. In the case of a failure, the reliable upgrade utility adds a record to this file indicating that the reliable upgrade has completed unsuccessfully and then terminates with an appropriate error code.

The `rupgrade` utility outputs any informational messages to `stdout` and any error messages to `stderr`. All messages are also recorded to the `/var/upgrade/log.txt` file.

The `rupgrade` utility can operate in several modes depending on the command-line parameters:

Upgrade mode:

```
rupgrade [-u <URL> | --uboot <URL>] [-k <URL> | --kernel <URL>] [-r
<URL> | --rootfs <URL>] [-a <URL> | --application <URL>] [--base
<string>] [--copy-config | --copy-home | --copy-etc | --copy-var |
--copy-all | --erase-all] [--clone] [--install-only] [--skip-
checksums] [-v <verbose level> | --verbose <verbose level>]
```

The main command-line parameters have the following meanings (and at least one of the following parameters must be specified):

`-u`, `--uboot <URL>` – specify the location of the new U-Boot image

`-k`, `--kernel <URL>` – specify the location of the new Linux kernel image

`-r`, `--rootfs <URL>` – specify the location of the new Linux root file system image

`-a`, `--application <URL>` – specify the location of the new application package image.

The following additional parameters are optional:

`--verbose <verbose level>` – specify the verbosity level (see below)

`--base <string>` – specify the string used as a prefix for all URLs in the command

`--copy-config` – copy the SSH keys, the Chassis Manager configuration files and the subdirectory `/var/nvdata` from the current image set to the new image set

`--copy-home` – copy the entire `/home` directory from the current image set to the new image set

`--copy-etc` – copy the entire `/etc` directory and the subdirectory `/var/nvdata` from the current image set to the new image set

`--copy-var` – copy the SSH keys and the entire `/var` directory from the current image set to the new image set

`--copy-all` – copy the entire `/var` directory and the entire `/etc` directory from the current image set to the new image set

`--erase-all` – erase both the `/var` and `/etc` directories for the new image set; copy only the SSH keys from the current image set

`--skip-checksums` – skip the verification of checksums in the images used for upgrade

`--install-only` – don't activate the new image set after the installation; continue running from the current image set. The command `backend --activate` can be used later to activate the new image set.

The following verbosity levels are supported:

- `none` – do not output verbose messages to `stdout`; do not write them to the log file
- `file` – do not output verbose messages to `stdout`; do write them to the log file
- `all` – output verbose messages to `stdout` and write them to the log file

The following command variants invoke the HPM.1 upgrade mode:

```
rupgrade -p <URL> [--copy-config | --copy-home | --copy-etc |
--copy-var | --copy-all | --erase-all] [--skip-checksums]
[--install-only]
```

```
rupgrade --package <URL> [--copy-config | --copy-home | --copy-etc
| --copy-var | --copy-all | --erase-all] [--skip-checksums]
[--install-only]
```

Here, `-p <URL>` or `--package <URL>` specifies the location of the new image in HPM.1 format (which contains some combination of the new U-Boot, kernel, RFS and application package images). Other optional parameters have the same meaning as for a non-HPM.1 upgrade.

The following command variant invokes the SmartFusion (A2F) cSoC upgrade mode (see section 7.7):

```
rupgrade –-upgrade-a2f <URL> [--skip-checksums]
```

The following command invokes rollback mode:

```
rupgrade --rollback
```

In this mode, the rollback takes place to the other image set (which is assumed to be the image set used before the last upgrade). Rollback doesn't affect the state of the SmartFusion cSoC image because the SmartFusion cSoC has its own auto-rollback mechanism that is utilized during an A2F reliable upgrade procedure.

Additional command variants invoke other modes of the reliable upgrade utility.

Get Upgrade Status mode (output current upgrade status on `stdout`):

```
rupgrade -s
rupgrade --status
```

Help mode (output a brief help summary on `stdout`):

rupgrade –h

rupgrade --help

Get Version mode (output the version of the utility to *stdout*):

rupgrade –-version

Add Hook Mode:

rupgrade --add-hook <URL>

In this mode, the utility adds a new hook script (specifically, the script file designated by the *<URL>*) to the list of scripts executed during the upgrade procedure. Scripts are stored in the directory */etc/hook*; to remove a hook script, just delete the corresponding file from this directory.

rupgrade –-clone

In this mode the utility copies the current image set into the new image set. This new image set is not activated.

The following table lists the protocols that are recognized in the URLs by the reliable upgrade utility:

Table 25: Protocols recognized in the URLs by the reliable upgrade utility

PROTOCOL NAME	DESCRIPTION	URL SYNTAX	EXAMPLE URL
File	Get image from a local ChMM file.	file://<abs_path> <abs_path> - absolute path to the image file on the ChMM (starts with "/")	file:///tmp/rootfs.image
HTTP	Download the image from an HTTP server	http://<host>/<path> <host> - host name or IP address <path> - relative path to the image file (path origin depends on HTTP server configuration)	http://192.168.1.253/ChMM700/rootfs.image

PROTOCOL NAME	DESCRIPTION	URL SYNTAX	EXAMPLE URL
TFTP	Download the image using the TFTP protocol	tftp://<host>/<path> <host> - host name or IP address <path> - relative path to the image file (path origin depends on TFTP server configuration)	tftp://192.168.1.253/ChMM700/rootfs.image
FTP	Download the image using the FTP protocol	ftp://[<user>[:<pass>]@]<host>/<path> <user> - user name <pass> - password <host> - host name or IP address <path> - relative path to the image file (path origin depends on FTP server configuration)	ftp://user:password@192.168.1.253/ChMM700/rootfs.image
SCP	Download the image using the SSH/SCP protocol	scp://[<user>@]<host>/<path> <user> - user name <host> - host name or ip address <path> - relative path to the image file (path origin is usually the user's home directory). An absolute path can be specified by beginning it with "//". The password must be entered interactively by the user.	scp://user@192.168.1.253/ChMM700/rootfs.image scp://user@192.168.1.253//home/user/ChMM700/rootfs.image

In upgrade mode (both regular and HPM.1), the reliable upgrade utility performs the following actions:

- obtains the images to copy: locally or via the network;
- copies the images to the candidate locations;

- copies non-volatile data to the candidate locations or erases the corresponding candidate locations (`/etc`, `/var`), depending on the presence of command-line parameters `--copy-config`, `--copy-home`, `--copy-etc`, `--copy-var`, `--copy-all` and `--erase-all`.
- if the option `--install-only` is not specified, resets the ChMM and instructs it to boot the candidate U-Boot (which in turn loads images from the candidate image set).

Because of the last stage, an invocation of `rupgrade` in upgrade mode typically does not return and instead resets the ChMM (unless the option `--install-only` is used). If `rupgrade` in upgrade mode does return, it indicates that the reliable upgrade procedure has failed and was terminated before proceeding to reset the ChMM in order to boot the candidate images.

Not all images need to be specified to `rupgrade` in upgrade mode (or, in the HPM.1 upgrade mode, the HPM.1 image may not include all possible components). If any components are not specified during the reliable upgrade, the missing components are copied from current confirmed component set. This allows reliable upgrades to be done when only certain firmware components (e.g. the application package) change between releases while other components are left intact; in that case, upgrade images can be made smaller and the upgrade itself will be faster. An example of such a partial upgrade is shown as Example 2 at the end of this section.

The `backend` utility has the following synopsis. Only the command-line parameters that are intended to be used by external programs are shown here; the remaining parameters, not shown here, are for the exclusive use of the `rupgrade` utility.

Output the current image set index (0 or 1) to `stdout`:

```
backend -c
backend --get-current
```

Activate the new image set previously installed via `rupgrade` invocation with the option `--install-only` (that is, start the upgrade watchdog timer, set the other image set as candidate and reset the ChMM instructing it to boot the candidate U-Boot):

```
backend --activate
```

Confirm and finalize the current upgrade:

```
backend --confirm
```

Cancel (abort) the current upgrade:

```
backend --cancel
```

Strobe the reliable upgrade watchdog timer:

```
backend --strobe
```

Help mode (output a brief help to `stdout`):

```
backend -h
backend --help
```

Get Version mode (output the version of the utility to `stdout`):

backend --version

Table 26: Standard source file names for upgrade components

COMPONENT NAME	SOURCE FILE NAME
U-Boot	sentry.chmm700.u-boot
Linux kernel	sentry.chmm700.kernel
RFS	sentry.chmm700.rfs
Application Package	sentry.chmm700.app

The image upgrade works as follows:

For each of the specified source images, the image is copied to the ChMM via the protocol specified in the URL (FTP, TFTP, etc.). The image is copied to the RAM file system of the ChMM (specifically, the copy is to the */tmp* directory which is cleaned up after each reboot) and after that it is validated. Image validation involves verification of the image checksum and is performed unless the command-line parameter *--skip-checksums* is specified. This procedure is repeated for all images specified in the command line.

After all images have been successfully obtained and validated, they are finally, one by one, dispatched to their destination locations; that is, U-Boot is copied to the destination Flash partition and other images are copied to appropriate locations on the UBIFS Boot and User volumes. If the reliable upgrade utility does not succeed in copying or validating at least one image, it immediately terminates the overall upgrade. Subsequent images, if any, are ignored; they are not even copied to the ChMM.

Before replacing kernel and RFS images in the Boot volume, the reliable upgrade utility cleans up the UBIFS cache partition to tell U-Boot that it is not valid any more. The UBIFS cache is re-created by U-Boot automatically. When copying the kernel and RFS images to the Boot UBIFS Flash partition, the reliable upgrade utility mounts the partition before copying the image and then unmounts it shortly after the completion of the copy operation. Normally, the Boot Flash partition stays unmounted. This ensures that applications cannot accidentally erase the critical boot-up images and allows U-Boot to cache the location of boot images for faster loading.

After all the specified images have been installed at their respective destinations, the utility mounts the candidate */etc* and */var* directories and either erases all files in them or copies the current contents of these directories to the candidate locations. This behavior is controlled by several command-line parameters (all of them mutually exclusive):

- *--copy-config*: if specified, the SSH keys (*/etc/ssh*), the Chassis Manager configuration files (*/etc/shelfman.conf**) and the subdirectory */var/nvdata* are copied to the candidate directories */etc* and */var* from the current location; any other files in the candidate */etc* and */var* directories are erased

- `--copy-home`: if specified, the entire `/home` directory is copied to the candidate directory `/home` from the current location; any other files in the candidate `/etc` and `/var` directories are erased
- `--copy-etc`: if specified, the entire `/etc` directory and the subdirectory `/var/nvdata` are copied to the candidate directories `/etc` and `/var` from the current location; any other files in the candidate `/var` directory are erased
- `--copy-var`: if specified, the SSH keys (`/etc/ssh`) and the entire `/var` directory are copied to the candidate directories `/etc` and `/var` from the current location; any other files in the candidate `/etc` directory are erased
- `--copy-all`: if specified, the entire `/etc` directory and the entire `/var` directory are copied to the candidate directories `/etc` and `/var` from the current location
- `--erase-all`: if specified, the candidate directories `/var` and `/etc` are erased, only the SSH keys (`/etc/ssh`) are copied from the current location.
- If none of the options above is specified (default mode), then the SSH keys (`/etc/ssh`) and the subdirectory `/var/nvdata` are copied to the candidate directories `/etc` and `/var` from the current location; any other files in the candidate `/etc` and `/var` directories are erased.

As the final stage of the upgrade (which takes place if the option `--install-only` is not specified), the utility starts the reliable upgrade watchdog timer (WDT) and resets the ChMM using the SmartFusion cSoC capabilities for ChMM reset. (Refer to *ChMM-700 Hardware Architecture Specification* for background and details on the reliable upgrade WDT.) The reliable upgrade WDT is started with a 60 second timeout period. This period should be sufficient for the ChMM software to reach (after the reset and without strobing the reliable upgrade WDT) the point in time where the ChMM software can detect whether a reliable upgrade is in progress (by invoking `rupgrade –-status`) and if so, strobe the reliable upgrade WDT via `backend --strobe`.

The reliable upgrade utility can invoke an optional “hook” script that enables custom actions required by an application. After each stage of the upgrade procedure, the hook script is invoked as a sub-shell and given a single parameter: the stage number. Table 24: above describes all stages of the reliable upgrade that are visible to the hook script.

The script returns `0` on success and a non-zero value in the case of a failure (in which case the upgrade procedure is terminated).

No hook script is set up by default. The user can run the reliable upgrade utility in the Add Hook mode to install a hook script. The script is copied to the directory `/etc/hook`; so if the directory `/etc` is copied to the candidate location, this installation is permanent; however, if the command-line parameter `--reset-etc` is specified, all hook scripts are erased after the upgrade.

7.7 Reliable upgrade of SmartFusion cSoC image

To upgrade the SmartFusion cSoC image, run the reliable upgrade utility in the SmartFusion (A2F) cSoC upgrade mode:

rupgrade –-upgrade-a2f <URL> [--skip-checksums]

Here *<URL>* designates the file that contains the SmartFusion cSoC upgrade image, the option *--skip-checksums*, if specified, requests the utility not to verify the checksum of the image file. The SmartFusion cSoC image file includes a SmartFusion A2F060 image concatenated with a SmartFusion A2F200 image. Each individual image file includes all the programmable content needed for the A2F060 device or the A2F200 device (including FPGA fabric and Cortex-M3 firmware). The reliable upgrade utility when run in the SmartFusion cSoC upgrade mode will determine if the SmartFusion device on the ChMM-700 is an A2F060 or an A2F200 and program the appropriate image from the concatenated SmartFusion cSoC image file.

Customers that intend to run the Chassis Manager in a system that supports operating the System IPMB in FAST-MODE (400 kHz) MUST upgrade to the SmartFusion image supplied with the release 1.3.0 or later. For customers that intend to run the Chassis Manager in a system that only supports operating the System IPMB in STANDARD-MODE (100 kHz), upgrading to the SmartFusion image supplied with the release 1.3.0 or later is not required.

A reliable upgrade of the SmartFusion cSoC image is done separately from a reliable upgrade of any other ChMM-700 firmware components and involves resetting the ChMM. Also, if the current image set has number 1, this procedure has a side effect that U-Boot image 0 is overwritten by the current U-Boot image; if the current image set has number 0, there are no such side effects.

The reliable upgrade of the SmartFusion cSoC image includes the following steps:

- The upgrade utility copies the upgrade image file onto the Flash Boot partition as *a2f-upgrade.dat*.
- If the current image set has number 1, the upgrade utility saves the contents of U-Boot image 0 to a file, and copies the U-Boot image from partition 1 to partition 0. This is done to ensure that if the upgrade process is interrupted, the ChMM-700 can still recover by booting from partition 0 (see below).
- The upgrade utility reboots the ChMM.
- When U-Boot sees the *a2f-upgrade.dat* file in the boot partition at startup, it initiates an A2F upgrade procedure via the SmartFusion cSoC JTAG pins. This procedure is performed with Flash remapping logic turned off (so that U-Boot has direct access to the Flash and U-Boot image 0 is always the active one). Also, the built-in i.MX287 watchdog timer is enabled during the upgrade so that if the upgrade process gets stuck, the ChMM-700 still has a potential to recover. As part of the A2F upgrade procedure the SmartFusion device type (A2F060 or A2F200) implemented on the ChMM-700 is determined so the appropriate image from the concatenated SmartFusion cSoC image file can programmed onto the device.
- If the upgrade is successful, U-Boot boots Linux from the current image set. When Linux boots, it performs some basic tests, and if everything works as expected, it disables the built-in i.MX287 watchdog timer and turns on the Flash remapping logic. It then restores the contents of the first U-Boot partition (if the backup file is present) and renames the SmartFusion cSoC

upgrade file on the boot partition to `a2f-auto-rollback.dat` to complete the upgrade process.

- If an `a2f-auto-rollback.dat` file was already present, it is renamed as `a2f-manual-rollback.dat` before overwriting it with the current upgrade file. This allows a manual rollback procedure to be initiated by invoking the rupgrade utility running in the `--upgrade-a2f` mode with the file `a2f-manual-rollback.dat` as the parameter. The UBIFS Boot volume may need to be explicitly mounted first.
- If the upgrade process is interrupted or an error is detected, the ChMM-700 will get reset with the Flash remapping logic turned off. As a result, it will boot from partition 0, and U-Boot will detect that Flash remapping logic is turned off. In this case, it will try to revert to the previous image by repeating the above steps with the auto-rollback image file name (`a2f-auto-rollback.dat`).

Note: the SmartFusion cSoC upgrade images provided by nVent are encrypted with a private AES key. This key is pre-programmed in the SmartFusion cSoC device on the ChMM-700R during manufacturing. As a consequence, the SmartFusion cSoC upgrade images that are not built by nVent cannot be installed onto the ChMM700 SmartFusion cSoC device. Attempting to do so will result in a failed upgrade.

7.8 Reliable upgrade utility use scenarios

It is intended that the reliable upgrade utility is used in the following sequence in order to carry out an upgrade of the ChMM:

- The user makes a call to `rupgrade` in upgrade mode (or in HPM.1 upgrade mode) to initiate the upgrade procedure. The call can be made either locally from the ChMM serial console or remotely over the network via `telnet`, `rsh`, `ssh`, or any equivalent.
- The user waits for `rupgrade` to reboot the ChMM. A user who has access to the ChMM serial console can see the messages produced there by U-Boot firmware and the Linux kernel, and therefore can easily monitor the status of the reboot. However, in the case of a remote connection to the ChMM, the status of the reboot can be more difficult to determine.
- For example, a telnet- or ssh-based virtual terminal connection usually times out on the reboot of the ChMM. The user can either assume that the upgrade was successful or wait for some time, reconnect to the ChMM and remotely invoke `rupgrade --status` in order to find out the current upgrade session status. The amount of time to wait is determined by the initialization time of the ChMM (the time needed to boot the Linux kernel and start the proper network daemons) and by the time needed to validate the functionality of the newly installed firmware image set.
- On the ChMM, the startup script `/etc/rc` unconditionally makes a call to `rupgrade --status`. The returned value indicates whether there is an upgrade session in progress (a value `0` is returned), no upgrade session is in progress (a positive value is returned) or the current upgrade session has failed (a negative error code is returned). If there is no upgrade

session in progress (the returned value is non-zero), the startup script continues with the normal mode boot-up sequence. If however, there is an upgrade session in progress (the returned value is *0*), the startup script proceeds with sanity validation of the newly installed software. In the middle of validation, it calls *backend --strobe* to strobe the upgrade WDT in case the validation takes longer than the upgrade WDT timeout period. Finally, the startup script starts the Chassis Manager to perform final validation. The watchdog timer interval is set to 60 seconds; so the processing time in the */etc/rc* script between the call to *rupgrade –-status* and the call to strobe the WDT and between the call to strobe the WDT and the startup of the Chassis Manager must not exceed 60 seconds each.

- During initialization, the Chassis Manager strobes the upgrade WDT once again, before trying to establish a network connection with the peer Chassis Manager. Establishing a network connection may take up to 6 seconds. After that, and after successfully finishing the initialization (which indicates validity of the new configuration), the Chassis Manager makes a call to *backend --confirm*, which completes the upgrade procedure.
- At this point, the user can invoke *rupgrade –-status* in order to find out the status of the current upgrade session. This may be useful if a reliable upgrade procedure is started remotely over the network via a virtual terminal (*telnet*, *rsh* or *ssh*) connection, because in that case the messages issued to the local serial console are not directly available.
- After the completion of the reliable upgrade, the user can revert to the original images if he/she detects that the new images are not acceptable for any reason. To do this, the user invokes *rupgrade --rollback*.

This sequence can be performed manually by an operator. However, if necessary, one can easily automate the above sequence via a script that runs on a host machine that is connected to the ChMM over the network, and remotely issues appropriate commands to the *rupgrade* and *backend* utilities on the ChMM.

7.9 Finding out component versions via the imls utility

A user can run the *imls* utility without parameters on the ChMM-700 to find out information (including version information) about the software components (U-Boot, kernel, RFS and the application package) that comprise both the current and the other image sets. The current image set number and the current version of the SmartFusion cSoC image is also shown. For example:

```
# imls
Current flash bank: 0

Current U-Boot:
Image Name:   U-Boot 9.0.8 for chmm700 board
Created:      Tue Oct 24 16:56:35 2017
Image Type:   ARM U-Boot Firmware (uncompressed)
Data Size:    524224 Bytes = 511.94 kB = 0.50 MB
Load Address: 00000000
```

```
Entry Point:  00000000

Current Kernel:
Image Name:   Linux-2.6.34.8-chmm700
Created:      Tue Oct 24 16:55:32 2017
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    1541476 Bytes = 1505.35 kB = 1.47 MB
Load Address: 40008000
Entry Point:  40008000

Current RFS:
Image Name:   ChMM700 RFS 1.2.0
Created:      Tue Oct 24 16:50:43 2017
Image Type:   ARM Linux RAMDisk Image (gzip compressed)
Data Size:    3092645 Bytes = 3020.16 kB = 2.95 MB
Load Address: 00000000
Entry Point:  00000000

Current Application:
Image Name:   ChMM700 APP 1.2.0
Created:      Tue Oct 24 16:50:45 2017
Image Type:   ARM Linux Filesystem Image (gzip compressed)
Data Size:    2495206 Bytes = 2436.72 kB = 2.38 MB
Load Address: 00000000
Entry Point:  00000000


Other U-Boot:
Image Name:   U-Boot 9.0.8 for chmm700 board
Created:      Mon Jun  8 18:31:36 2015
Image Type:   ARM U-Boot Firmware (uncompressed)
Data Size:    524224 Bytes = 511.94 kB = 0.50 MB
Load Address: 00000000
Entry Point:  00000000

Other Kernel:
Image Name:   Linux-2.6.34.8-chmm700
Created:      Mon Jun  8 18:29:25 2015
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    1540396 Bytes = 1504.29 kB = 1.47 MB
```

```
Load Address: 40008000
Entry Point:  40008000

Other RFS:
Image Name:   ChMM700 RFS 1.1.0
Created:      Mon Jun  8 18:21:15 2015
Image Type:   ARM Linux RAMDisk Image (gzip compressed)
Data Size:    2668775 Bytes = 2606.23 kB = 2.55 MB
Load Address: 00000000
Entry Point:  00000000

Other Application:
Image Name:   ChMM700 APP 1.1.0
Created:      Mon Jun  8 18:21:18 2015
Image Type:   ARM Linux Filesystem Image (gzip compressed)
Data Size:    2375024 Bytes = 2319.36 kB = 2.26 MB
Load Address: 00000000
Entry Point:  00000000

A2F:   SPICOMM protocol v1.7, M3 firmware v1.8, FPGA design v1.1.0.0
```

7.10 Reliable upgrade examples

This section provides reliable upgrade examples for ChMM-700.

7.10.1 Example 1

This example shows a reliable upgrade of all four components (U-Boot, kernel, RFS image and application package), copying *`/etc`* and *`/var/nvdata`* non-volatile directories to the candidate locations. All images are taken from the local *`/tmp`* (which implies that they have already been copied there in some unspecified way). The U-Boot image is taken from *`/tmp/sentry.chmm700.u-boot`*, the kernel image is taken from *`/tmp/sentry.chmm700.kernel`*, the RFS image is taken from *`/tmp/sentry.chmm700.rfs`*, the application image is taken from *`/tmp/sentry.chmm700.app`*. The upgrade procedure is started from the serial console. Comments are interspersed in the console log to provide additional background on the steps of the upgrade procedure.

First, the *`rupgrade`* utility is started from the command prompt in upgrade mode. The parameters show that all four images are to be updated, with the Chassis Manager non-volatile data and configuration file preserved as well.

```
# rupgrade --skip-checksums --copy-etc -k file:///tmp/sentry.chmm700.kernel -r
file:///tmp/sentry.chmm700.rfs -u file:///tmp/sentry.chmm700.u-boot -a
file:///tmp/sentry.chmm700.app
```

The upgrade utility attempts to invoke a validation script to check the images in */tmp* currently supplied. If any of the specified file designators is not found in */tmp*, the utility stops and a message like the following is produced.

```
<INFO> Downloading file "file:///tmp/u-boot.bin"
<ERR>  Failed to download file "file:///tmp/u-boot.bin": Error
```

The utility proceeds to copy the images to the specified destinations in candidate Flash.

```
<INFO> Downloading file "file:///tmp/sentry.chmm700.u-boot"
<INFO> Downloading file "file:///tmp/sentry.chmm700.kernel"
<INFO> Downloading file "file:///tmp/sentry.chmm700.rfs"
<INFO> Downloading file "file:///tmp/sentry.chmm700.app"
<INFO> Initiating partial upgrade:
<INFO>  * uboot "/tmp/sentry.chmm700.u-boot"
<INFO>  * kernel "/tmp/sentry.chmm700.kernel"
<INFO>  * rfs "/tmp/sentry.chmm700.rfs"
<INFO>  * app "/tmp/sentry.chmm700.app"
<INFO>  + copy etc
<INFO> Calling backend to handle partial upgrade "/sbin/backend --uboot
/tmp/sentry.chmm700.u-boot --kernel /tmp/sentry.chmm700.kernel --rootfs
/tmp/sentry.chmm700.rfs --application /tmp/sentry.chmm700.app --copy-etc"
217: Executing: /sbin/backend --uboot /tmp/sentry.chmm700.u-boot --kernel
/tmp/sentry.chmm700.kernel --rootfs /tmp/sentry.chmm700.rfs --application
/tmp/sentry.chmm700.app --copy-etc
221: Executing: /etc/hook/base_script.sh --stage 1 --silent
221: System returned 0, exec result 0.
<INFO> Write storage state: not available -> available [ALLOW]
221: Executing: mkdir -p /var/upgrade/boot
221: System returned 0, exec result 0.
<INFO> Resetting UBIFS cache
221: Executing: /sbin/eraseall /dev/mtd8
Erasing 256 Kibyte @ 0 --  0 % complete.
221: System returned 0, exec result 0.
221: Executing: /etc/hook/base_script.sh --stage 2 --silent
221: System returned 0, exec result 0.
221: Executing: /etc/hook/base_script.sh --stage 3 --silent
221: System returned 0, exec result 0.
<INFO> Burning image:..
<INFO> 524288 bytes were written from file "/tmp/sentry.chmm700.u-boot" to dev
1[1]
```

Here and below, *rupgrade* reports that a new image has been written to the storage device. The number in brackets is the candidate image set number. In this example, the confirmed image number is 0 and the candidate number is 1.

```
221: Executing: eraseall /dev/mtd6 >/dev/null 2>/dev/null
221: System returned 0, exec result 0.
221: Executing: dd if=/dev/mtd4 of=/dev/mtd6 >/dev/null 2>/dev/null
221: System returned 0, exec result 0.
221: Executing: /etc/hook/base_script.sh --stage 4 --silent
221: System returned 0, exec result 0.
<INFO> Cleaning image "/var/upgrade/boot/uImage.1"
<INFO> Burning
image:.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................
<INFO> 1502912 bytes were written from file "/tmp/sentry.chmm700.kernel" to dev
2[1]
221: Executing: /etc/hook/base_script.sh --stage 5 --silent
221: System returned 0, exec result 0.
<INFO> Cleaning image "/var/upgrade/boot/rfs.1"
<INFO> Burning
image:.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.........................
<INFO> 2697316 bytes were written from file "/tmp/sentry.chmm700.rfs" to dev
4[1]
221: Executing: /etc/hook/base_script.sh --stage 6 --silent
221: System returned 0, exec result 0.
221: Executing: /etc/hook/base_script.sh --stage 7 --silent
221: System returned 0, exec result 0.
<INFO> Burning
image:.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
...
<INFO> 2280508 bytes were written from file "/tmp/sentry.chmm700.app" to dev
8[1]
221: Executing: /etc/hook/base_script.sh --stage 8 --silent
221: System returned 0, exec result 0.
```

```
<INFO> Write storage state: available -> not available [ALLOW]
221: Executing: /etc/hook/base_script.sh --stage 9 --silent
221: System returned 0, exec result 0.
221: Executing: /etc/hook/base_script.sh --stage 10 --silent
221: System returned 0, exec result 0.
<INFO> Erasing user data on candidate bank
221: Executing: (cd /1 ; rm -rf var etc ; mkdir -p var/nvdata etc/ssh )
221: System returned 0, exec result 0.
221: Executing: ln -sf /boot/app.1 /1/sentry.chmm700.app
221: System returned 0, exec result 0.
<INFO> Copy /etc.
221: Executing: cp -a /0/etc /1
221: System returned 0, exec result 0.
221: Executing: cp -a /0/var/nvdata /1/var
221: System returned 0, exec result 0.
221: Executing: /etc/hook/base_script.sh --stage 11 --silent
221: System returned 0, exec result 0.
221: Executing: /etc/hook/base_script.sh --stage 12 --silent
221: System returned 0, exec result 0.
221: Executing: dd if=/dev/mtd8 of=/dev/null >/dev/null 2>/dev/null
221: System returned 0, exec result 0.
<INFO> Write confirmed: 0 -> 0 [ALLOW]
<INFO> Write candidate: 1 -> 1 [ALLOW]
<INFO> Write upgrade watchdog: 0 -> 1 [ALLOW]
```

Here, the reliable upgrade procedure resets the ChMM. This causes U-Boot to start from the candidate Flash.

```
U-Boot 2009.08 (Oct 24 2017 - 11:44:23)

Freescale i.MX28 family
CPU:   297 MHz
BUS:   99 MHz
EMI:   130 MHz
GPMI:  24 MHz
I2C:   ready
DRAM:  128 MB
SFGEN: S25FL512S detected, total size 64 MB
A2F:   SPICOMM protocol v1.7, M3 firmware v1.8, FPGA design v1.1.0.0
A2F:   A2F firmware, version 1.8
A2F:   Device type: A2F060M3E-FG256
A2F:   MSS clock frequency: 40 MHz
```

```
A2F:   Fabric clock frequency: 20 MHz
A2F:   Fast delay calibration: 1330 cycles per 100uS
A2F:   eNVM: 128 KB (00000000 - 00020000)
A2F:   eSRAM: 16 KB (20000000 - 20004000)
A2F:   Extram start: 20002750
A2F:   Last reset cause: RUWD
A2F:   VMON flags: 00000000
RUPG:  booting from image 1 (candidate)
```

Here, U-Boot reports that reliable upgrade is in progress and the candidate image set is being used. The candidate image set index is 0 and the confirmed image set is 1.

```
In:    serial
Out:   serial
Err:   serial
Net:   FEC0: 00:18:49:02:b2:06, FEC1: 00:18:49:02:b2:07
FEC0, FEC1
Hit any key to stop autoboot:  0
RUPG:  reliable upgrade is in progress, skipping
```

Here, U-Boot reports that a reliable upgrade is in progress and therefore SmartFusion cSoC upgrade stage is skipped; it doesn't matter if the SmartFusion cSoCupgrade file is present or not.

```
Creating 1 MTD partitions on "spi0":
0x0000001c0000-0x000002000000 : "mtd=5"
```

Below, U-Boot attaches the UBI partition and mounts the UBIFS volume to load the new kernel and RFS images. This stage is skipped and the messages listed below do not appear during regular boots because a UBIFS cache is used.

```
UBI: attaching mtd1 to ubi0
UBI: MTD device 1 is write-protected, attach in read-only mode
UBI: physical eraseblock size:   262144 bytes (256 KiB)
UBI: logical eraseblock size:    262016 bytes
UBI: smallest flash I/O unit:    1
UBI: VID header offset:          64 (aligned 64)
UBI: data offset:                128
UBI: attached mtd1 to ubi0
UBI: MTD device name:            "mtd=5"
UBI: MTD device size:            30 MiB
UBI: number of good PEBs:        121
UBI: number of bad PEBs:         0
UBI: max. allowed volumes:       128
UBI: wear-leveling threshold:    4096
UBI: number of internal volumes: 1
UBI: number of user volumes:     1
```

```
UBI: available PEBs:             0
UBI: total number of reserved PEBs: 121
UBI: number of PEBs reserved for bad PEB handling: 0
UBI: max/mean erase counter: 27/19
UBIFS: read-only UBI device
UBIFS: mounted UBI device 0, volume 0, name "boot"
UBIFS: mounted read-only
UBIFS: file system size:   28297728 bytes (27634 KiB, 26 MiB, 108 LEBs)
UBIFS: journal size:       2096129 bytes (2047 KiB, 1 MiB, 6 LEBs)
UBIFS: media format:       w4/r0 (latest is w4/r0)
UBIFS: default compressor: LZO
UBIFS: reserved for root:  1398497 bytes (1365 KiB)
Loading file 'uImage.1' to addr 0x42000000 with size 1502912 (0x0016eec0)...
Done
Loading file 'rfs.1' to addr 0x46000000 with size 2697316 (0x00292864)...
Done
## Booting kernel from Legacy Image at 42000000 ...
   Image Name:   Linux-2.6.34.8-chmm700
   Created:      2017-10-24  16:44:21 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    1541472 Bytes =  1.5 MB
   Load Address: 40008000
   Entry Point:  40008000
   Verifying Checksum ... OK
## Loading init Ramdisk from Legacy Image at 46000000 ...
   Image Name:   ChMM700 RFS 1.2.0
   Created:      2017-10-24  16:39:38 UTC
   Image Type:   ARM Linux RAMDisk Image (gzip compressed)
   Data Size:    3092652 Bytes =  2.9 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...
Starting kernel ...

init started: BusyBox v1.25.1 (2017-10-24 11:38:54 CDT)
/etc/rc: Current image set is 1
```

```
/etc/rc: Mounting filesystems...
/etc/rc: Mounted /proc
/etc/rc: Mounted /dev/pts
/etc/rc: Mounted /tmp
/etc/rc: Mounted /sys
/etc/rc: Checking the reliable upgrade watchdog timer...activated
<INFO> Write upgrade watchdog: 1 -> 2 [ALLOW]
```

Here, *rupgrade* is invoked for the first time after the reboot to strobe the reliable upgrade watchdog timer.

```
/etc/rc: Attaching ubi0:user...done
/etc/rc: Mounting  ubi0:user...done
/etc/rc: Attaching ubi1:boot...done
/etc/rc: Mounting  ubi1:boot...done
/etc/rc: Extracting application package /boot/app.1...done
/etc/rc: Mounted /1/var to /var
/etc/rc: Strobing the reliable upgrade WDT
```

Here, *rupgrade* is invoked for the second time to strobe the reliable upgrade WDT.

```
<INFO> Write upgrade watchdog: 1 -> 2 [ALLOW]
89: Executing: /etc/hook/base_script.sh --stage 13 --silent
89: System returned 0, exec result 0.
/etc/rc: Mounted /1/etc to /etc
/etc/rc: Setting timezone to MSK-4
/etc/rc: Placed /var/tmp to ram disk
/etc/rc: Setting hostname chmm~110
/etc/rc: Started syslogd and klogd
/etc/rc: Calling /etc/rc.chmm700-hpdl
/etc/readhwaddr: Board Hardware Address: 0x10
/etc/netconfig: /etc/hosts has valid chmm~110 192.168.1.110 entry
/etc/netconfig: Updating /etc/profile.sentry with IP settings
/etc/netconfig: ifconfig eth0 192.168.1.110 netmask 255.255.255.0
/etc/netconfig: ifconfig usb0 192.168.0.110 netmask 255.255.255.0
/etc/netconfig: route add default gw 192.168.1.253
/etc/netconfig: Starting /bin/inetd...
/etc/netconfig: Starting time synchronization script
/etc/rc.chmm700-hpdl: Updating /etc/profile.sentry with specific settings
/etc/rc.chmm700-hpdl: RC2 daemons not started by request
/etc/rc: start_rc2_daemons=n - finishing reliable upgrade process now.
```

The reliable upgrade WDT is strobed once again, just to make sure it won't fire during the confirm procedure.

```
<INFO> Write confirmed: 0 -> 1 [ALLOW]
```

rupgrade changes the confirmed image set index to the candidate image set index.

```
<INFO> Write upgrade_state: "in progress" (2) -> "confirmed" (4) [ALLOW]
<INFO> Write upgrade watchdog: 1 -> 0 [ALLOW]
```

And finally, *rupgrade* turns the reliable upgrade WDT off. The reliable upgrade is complete.

Finally, the user checks the status of the reliable upgrade, by calling *rupgrade –s*.

```
# rupgrade -s
<INFO> Last upgrade log:
<INFO> Write storage state: not available -> available [ALLOW]
<INFO> Resetting UBIFS cache
<INFO> Burning image:<INFO> 524288 bytes were written from file
"/tmp/sentry.chmm700.u-boot" to dev 1[1]
<INFO> Cleaning image "/var/upgrade/boot/uImage.1"
<INFO> Burning image:<INFO> 1502912 bytes were written from file
"/tmp/sentry.chmm700.kernel" to dev 2[1]
<INFO> Cleaning image "/var/upgrade/boot/rfs.1"
<INFO> Burning image:<INFO> 2697316 bytes were written from file
"/tmp/sentry.chmm700.rfs" to dev 4[1]
<INFO> Burning image:<INFO> 2280508 bytes were written from file
"/tmp/sentry.chmm700.app" to dev 8[1]
<INFO> Write storage state: available -> not available [ALLOW]
<INFO> Erasing user data on candidate bank
<INFO> Copy /etc.
<INFO> Write confirmed: 0 -> 0 [ALLOW]
<INFO> Write candidate: 1 -> 1 [ALLOW]
<INFO> Write upgrade watchdog: 0 -> 1 [ALLOW]
<INFO> Write upgrade watchdog: 1 -> 2 [ALLOW]
<INFO> Write confirmed: 0 -> 1 [ALLOW]
<INFO> Write upgrade_state: "in progress" (2) -> "confirmed" (4) [ALLOW]
<INFO> Write upgrade watchdog: 1 -> 0 [ALLOW]
<INFO> Last upgrade stages log:
0 (initial): PASSED
1 (before storage mount): PASSED
2 (after storage mount): PASSED
3 (after marking image set invalid): PASSED
4 (after u-boot burn): PASSED
5 (after kernel burn): PASSED
6 (after rootfs burn): PASSED
7 (after storage umount): PASSED
8 (after cleaning previous apps set): PASSED
```

```
9 (after writting new apps set): PASSED
10 (after sync): PASSED
11 (after processing var and etc): PASSED
12 (after boot select): PASSED
13 (after reboot): PASSED
14 (after confirm): PASSED
237: Executing: /sbin/backend -s
237: System returned 3584, exec result 14.
<INFO> Last upgrade stage from HAL: 14
#
```

7.10.2 Example 2

This example shows a reliable upgrade of the application package image only, copying `/etc` and `/var/nvdata` non-volatile directories from confirmed to candidate locations. The application package image is taken from a TFTP server at the IP address 192.168.1.253. The path to the application package image on the TFTP server is `ChMM700/sentry.chmm700.app`. The upgrade procedure is started from a telnet session.

> *Note: Since only the application package image is explicitly updated, the U-Boot, kernel and RFS images are automatically copied from the confirmed locations to the candidate locations.*

The ChMM must be able to access the TFTP server over the network (that is, its network adapter must be up and configured and a route must exist from the ChMM to the TFTP server). In the example below, the ChMM is configured with the network address 192.168.1.123 (which is in the same network with the TFTP server):

```
# telnet 192.168.1.123
Trying 192.168.1.123...
Connected to 192.168.1.123.
Escape character is '^]'.
ChMM700 login: root
Password:
#
```

The parameters to `rupgrade` indicate that only the application package is being upgraded and that the copy protocol is TFTP, accessing a specified IP address and file.

```
# rupgrade --skip-checksums --copy-etc -a
tftp://192.168.1.253/ChMM700/sentry.chmm700.app
143: Executing: /sbin/backend --get-wdt-state
143: System returned 0, exec result 0.
<INFO> Downloading file "tftp://192.168.1.253/ChMM700/sentry.chmm700.app"
<INFO> Initiating partial upgrade:
<INFO>  * app "/tmp/_ChMM700_sentry.chmm700.app"
```

```
<INFO>  + copy etc
<INFO> Calling backend to handle partial upgrade "/sbin/backend --application
/tmp/_ChMM700_sentry.chmm700.app --copy-etc"
143: Executing: /sbin/backend --application /tmp/_ChMM700_sentry.chmm700.app --
copy-etc
147: Executing: /etc/hook/base_script.sh --stage 1 --silent
147: System returned 0, exec result 0.
<INFO> Write storage state: not available -> available [ALLOW]
147: Executing: mkdir -p /var/upgrade/boot
147: System returned 0, exec result 0.
<INFO> Resetting UBIFS cache
147: Executing: /sbin/eraseall /dev/mtd8
Erasing 256 Kibyte @ 0 --  0 % complete.
147: System returned 0, exec result 0.
147: Executing: /etc/hook/base_script.sh --stage 2 --silent
147: System returned 0, exec result 0.
147: Executing: /etc/hook/base_script.sh --stage 3 --silent
147: System returned 0, exec result 0.
<INFO> Clone:..
<INFO> File "/dev/mtd0" of device 1[0] cloned to file "/dev/mtd2" of device
1[1].
147: Executing: eraseall /dev/mtd6 >/dev/null 2>/dev/null
147: System returned 0, exec result 0.
147: Executing: dd if=/dev/mtd4 of=/dev/mtd6 >/dev/null 2>/dev/null
147: System returned 0, exec result 0.
147: Executing: /etc/hook/base_script.sh --stage 4 --silent
147: System returned 0, exec result 0.
<INFO> Cleaning image "/var/upgrade/boot/uImage.1"
<INFO>
Clone:.....................................................................
.........................................................................
.........................................................................
.........................................................................
.........................................................
<INFO> File "/var/upgrade/boot/uImage.0" of device 2[0] cloned to file
"/var/upgrade/boot/uImage.1" of device 2[1].
147: Executing: /etc/hook/base_script.sh --stage 5 --silent
147: System returned 0, exec result 0.
<INFO> Cleaning image "/var/upgrade/boot/rfs.1"
<INFO>
Clone:.....................................................................
.........................................................................
.........................................................................
```

```
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................
<INFO> File "/var/upgrade/boot/rfs.0" of device 4[0] cloned to file
"/var/upgrade/boot/rfs.1" of device 4[1].
147: Executing: /etc/hook/base_script.sh --stage 6 --silent
147: System returned 0, exec result 0.
147: Executing: /etc/hook/base_script.sh --stage 7 --silent
147: System returned 0, exec result 0.
<INFO> Burning
image:...............................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
...
<INFO> 2280508 bytes were written from file "/tmp/_ChMM700_sentry.chmm700.app"
to dev 8[1]
147: Executing: /etc/hook/base_script.sh --stage 8 --silent
147: System returned 0, exec result 0.
<INFO> Write storage state: available -> not available [ALLOW]
147: Executing: /etc/hook/base_script.sh --stage 9 --silent
147: System returned 0, exec result 0.
147: Executing: /etc/hook/base_script.sh --stage 10 --silent
147: System returned 0, exec result 0.
<INFO> Erasing user data on candidate bank
147: Executing: (cd /1 ; rm -rf var etc ; mkdir -p var/nvdata etc/ssh )
147: System returned 0, exec result 0.
147: Executing: ln -sf /boot/app.1 /1/sentry.chmm700.app
147: System returned 0, exec result 0.
<INFO> Copy /etc.
147: Executing: cp -a /0/etc /1
147: System returned 0, exec result 0.
147: Executing: cp -a /0/var/nvdata /1/var
147: System returned 0, exec result 0.
147: Executing: /etc/hook/base_script.sh --stage 11 --silent
147: System returned 0, exec result 0.
147: Executing: /etc/hook/base_script.sh --stage 12 --silent
147: System returned 0, exec result 0.
```

```
147: Executing: dd if=/dev/mtd8 of=/dev/null >/dev/null 2>/dev/null
147: System returned 0, exec result 0.
<INFO> Write confirmed: 0 -> 0 [ALLOW]
<INFO> Write candidate: 1 -> 1 [ALLOW]
<INFO> Write upgrade watchdog: 0 -> 1 [ALLOW]
```

At this point, the telnet session is closed after a certain inactivity period; after several seconds, it is possible to reconnect to the target again and check the status of the reliable upgrade by invoking *rupgrade –s*.

```
# telnet 192.168.1.123
Trying 192.168.1.123...
Connected to 192.168.1.123.
Escape character is '^]'.

ChMM700 login: root
Password:
# rupgrade -s
<INFO> Last upgrade log:
<INFO> Write storage state: not available -> available [ALLOW]
<INFO> Resetting UBIFS cache
<INFO> Clone:..
<INFO> File "/dev/mtd0" of device 1[0] cloned to file "/dev/mtd2" of device
1[1].
<INFO> Cleaning image "/var/upgrade/boot/uImage.1"
<INFO>
Clone:.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................
<INFO> File "/var/upgrade/boot/uImage.0" of device 2[0] cloned to file
"/var/upgrade/boot/uImage.1" of device 2[1].
<INFO> Cleaning image "/var/upgrade/boot/rfs.1"
<INFO>
Clone:.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.................................
<INFO> File "/var/upgrade/boot/rfs.0" of device 4[0] cloned to file
"/var/upgrade/boot/rfs.1" of device 4[1].
```

```
<INFO> Burning image:<INFO> 2280508 bytes were written from file
"/tmp/_ChMM700_sentry.chmm700.app" to dev 8[1]
<INFO> Write storage state: available -> not available [ALLOW]
<INFO> Erasing user data on candidate bank
<INFO> Copy /etc.
<INFO> Write confirmed: 0 -> 0 [ALLOW]
<INFO> Write candidate: 1 -> 1 [ALLOW]
<INFO> Write upgrade watchdog: 0 -> 1 [ALLOW]
<INFO> Write upgrade watchdog: 1 -> 2 [ALLOW]
<INFO> Write confirmed: 0 -> 1 [ALLOW]
<INFO> Write upgrade_state: "in progress" (2) -> "confirmed" (4) [ALLOW]
<INFO> Write upgrade watchdog: 1 -> 0 [ALLOW]
<INFO> Last upgrade stages log:
0 (initial): PASSED
1 (before storage mount): PASSED
2 (after storage mount): PASSED
3 (after marking image set invalid): PASSED
4 (after u-boot burn): PASSED
5 (after kernel burn): PASSED
6 (after rootfs burn): PASSED
7 (after storage umount): PASSED
8 (after cleaning previous apps set): PASSED
9 (after writting new apps set): PASSED
10 (after sync): PASSED
11 (after processing var and etc): PASSED
12 (after boot select): PASSED
13 (after reboot): PASSED
14 (after confirm): PASSED
161: Executing: /sbin/backend -s
161: System returned 3584, exec result 14.
<INFO> Last upgrade stage from HAL: 14
#
```

7.10.3 Example 3

This example shows an unsuccessful reliable upgrade. Power is turned off after the boot from the candidate locations, but before the reliable upgrade is finalized. After turning the power back on, the rollback to the confirmed images occurs. This reliable upgrade is initiated from the serial console. The default copy mode is used (copy SSH keys and `/var/nvdata`). All four images are assumed to be already in `/tmp`.

```
# rupgrade --skip-checksums -k file:///tmp/sentry.chmm700.kernel -r
file:///tmp/sentry.chmm700.rfs -u file:///tmp/sentry.chmm700.u-boot -a
file:///tmp/sentry.chmm700.app
165: Executing: /sbin/backend --get-wdt-state
165: System returned 0, exec result 0.
<INFO> Downloading file "file:///tmp/sentry.chmm700.u-boot"
<INFO> Downloading file "file:///tmp/sentry.chmm700.kernel"
<INFO> Downloading file "file:///tmp/sentry.chmm700.rfs"
<INFO> Downloading file "file:///tmp/sentry.chmm700.app"
<INFO> Initiating partial upgrade:
<INFO>  * uboot "/tmp/sentry.chmm700.u-boot"
<INFO>  * kernel "/tmp/sentry.chmm700.kernel"
<INFO>  * rfs "/tmp/sentry.chmm700.rfs"
<INFO>  * app "/tmp/sentry.chmm700.app"
<INFO> Calling backend to handle partial upgrade "/sbin/backend --uboot
/tmp/sentry.chmm700.u-boot --kernel /tmp/sentry.chmm700.kernel --rootfs
/tmp/sentry.chmm700.rfs --application /tmp/sentry.chmm700.app"
165: Executing: /sbin/backend --uboot /tmp/sentry.chmm700.u-boot --kernel
/tmp/sentry.chmm700.kernel --rootfs /tmp/sentry.chmm700.rfs --application
/tmp/sentry.chmm700.app
169: Executing: /etc/hook/base_script.sh --stage 1 --silent
169: System returned 0, exec result 0.
<INFO> Write storage state: not available -> available [ALLOW]
169: Executing: mkdir -p /var/upgrade/boot
169: System returned 0, exec result 0.
<INFO> Resetting UBIFS cache
169: Executing: /sbin/eraseall /dev/mtd8
Erasing 256 Kibyte @ 0 --  0 % complete.
169: System returned 0, exec result 0.
169: Executing: /etc/hook/base_script.sh --stage 2 --silent
169: System returned 0, exec result 0.
169: Executing: /etc/hook/base_script.sh --stage 3 --silent
169: System returned 0, exec result 0.
<INFO> Burning image:..
<INFO> 524288 bytes were written from file "/tmp/sentry.chmm700.u-boot" to dev
1[1]
169: Executing: eraseall /dev/mtd6 >/dev/null 2>/dev/null
169: System returned 0, exec result 0.
169: Executing: dd if=/dev/mtd4 of=/dev/mtd6 >/dev/null 2>/dev/null
169: System returned 0, exec result 0.
169: Executing: /etc/hook/base_script.sh --stage 4 --silent
169: System returned 0, exec result 0.
```

```
<INFO> Cleaning image "/var/upgrade/boot/uImage.1"
<INFO> Burning
image:.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................
<INFO> 1502912 bytes were written from file "/tmp/sentry.chmm700.kernel" to dev
2[1]
169: Executing: /etc/hook/base_script.sh --stage 5 --silent
169: System returned 0, exec result 0.
<INFO> Cleaning image "/var/upgrade/boot/rfs.1"
<INFO> Burning
image:.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.........................
<INFO> 2697316 bytes were written from file "/tmp/sentry.chmm700.rfs" to dev
4[1]
169: Executing: /etc/hook/base_script.sh --stage 6 --silent
169: System returned 0, exec result 0.
169: Executing: /etc/hook/base_script.sh --stage 7 --silent
169: System returned 0, exec result 0.
<INFO> Burning
image:.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
...
<INFO> 2280508 bytes were written from file "/tmp/sentry.chmm700.app" to dev
8[1]
169: Executing: /etc/hook/base_script.sh --stage 8 --silent
169: System returned 0, exec result 0.
<INFO> Write storage state: available -> not available [ALLOW]
169: Executing: /etc/hook/base_script.sh --stage 9 --silent
169: System returned 0, exec result 0.
169: Executing: /etc/hook/base_script.sh --stage 10 --silent
169: System returned 0, exec result 0.
<INFO> Erasing user data on candidate bank
```

```
169: Executing: (cd /1 ; rm -rf var etc ; mkdir -p var/nvdata etc/ssh )
169: System returned 0, exec result 0.
169: Executing: ln -sf /boot/app.1 /1/sentry.chmm700.app
169: System returned 0, exec result 0.
<INFO> Copy ssh keys
169: Executing: cp -a /0/etc/ssh /1/etc
169: System returned 0, exec result 0.
169: Executing: cp -a /0/var/nvdata /1/var
169: System returned 0, exec result 0.
169: Executing: /etc/hook/base_script.sh --stage 11 --silent
169: System returned 0, exec result 0.
169: Executing: /etc/hook/base_script.sh --stage 12 --silent
169: System returned 0, exec result 0.
169: Executing: dd if=/dev/mtd8 of=/dev/null >/dev/null 2>/dev/null
169: System returned 0, exec result 0.
<INFO> Write confirmed: 0 -> 0 [ALLOW]
<INFO> Write candidate: 1 -> 1 [ALLOW]
<INFO> Write upgrade watchdog: 0 -> 1 [ALLOW]
```

The reliable upgrade procedure resets the ChMM here and starts U-Boot from the candidate Flash.

```
U-Boot 2009.08 (Oct 24 2017 - 11:44:23)

Freescale i.MX28 family
CPU:   297 MHz
BUS:   99 MHz
EMI:   130 MHz
GPMI:  24 MHz
I2C:   ready
DRAM:  128 MB
SFGEN: S25FL512S detected, total size 64 MB
A2F:   SPICOMM protocol v1.7, M3 firmware v1.8, FPGA design v1.1.0.0
A2F:   A2F firmware, version 1.8
A2F:   Device type: A2F060M3E-FG256
A2F:   MSS clock frequency: 40 MHz
A2F:   Fabric clock frequency: 20 MHz
A2F:   Fast delay calibration: 1330 cycles per 100uS
A2F:   eNVM: 128 KB (00000000 - 00020000)
A2F:   eSRAM: 16 KB (20000000 - 20004000)
A2F:   Extram start: 20002750
A2F:   Last reset cause: RUWD
A2F:   VMON flags: 00000000
```

```
RUPG:  booting from image 1 (confirmed)
```

Here, U-Boot reports that a reliable upgrade is in progress and the candidate image set is being used. The candidate image set index is 1 and the confirmed image set is 0.

```
In:    serial
Out:   serial
Err:   serial
Net:   FEC0: 00:18:49:02:b2:06, FEC1: 00:18:49:02:b2:07
FEC0, FEC1
Hit any key to stop autoboot:  0
RUPG:  reliable upgrade is in progress, skipping
Creating 1 MTD partitions on "spi0":
0x0000001c0000-0x000002000000 : "mtd=5"
UBI: attaching mtd1 to ubi0
UBI: MTD device 1 is write-protected, attach in read-only mode
UBI: physical eraseblock size:   262144 bytes (256 KiB)
UBI: logical eraseblock size:    262016 bytes
UBI: smallest flash I/O unit:    1
UBI: VID header offset:          64 (aligned 64)
UBI: data offset:                128
UBI: attached mtd1 to ubi0
UBI: MTD device name:            "mtd=5"
UBI: MTD device size:            30 MiB
UBI: number of good PEBs:        121
UBI: number of bad PEBs:         0
UBI: max. allowed volumes:       128
UBI: wear-leveling threshold:    4096
UBI: number of internal volumes: 1
UBI: number of user volumes:     1
UBI: available PEBs:             0
UBI: total number of reserved PEBs: 121
UBI: number of PEBs reserved for bad PEB handling: 0
UBI: max/mean erase counter: 28/20
UBIFS: read-only UBI device
UBIFS: mounted UBI device 0, volume 0, name "boot"
UBIFS: mounted read-only
UBIFS: file system size:   28297728 bytes (27634 KiB, 26 MiB, 108 LEBs)
UBIFS: journal size:       2096129 bytes (2047 KiB, 1 MiB, 6 LEBs)
UBIFS: media format:       w4/r0 (latest is w4/r0)
UBIFS: default compressor: LZO
UBIFS: reserved for root:  1398497 bytes (1365 KiB)
```

```
Loading file 'uImage.1' to addr 0x42000000 with size 1502912 (0x0016eec0)...
```

Power is turned off here. After some time, power is turned back on. The candidate Flash state has been lost because of the power loss, so the ChMM reverts back to the confirmed Flash.

```
U-Boot 2009.08 (Oct 24 2017 - 11:44:23)

Freescale i.MX28 family
CPU:   297 MHz
BUS:   99 MHz
EMI:   130 MHz
GPMI:  24 MHz
I2C:   ready
DRAM:  128 MB
SFGEN: S25FL512S detected, total size 64 MB
A2F:   SPICOMM protocol v1.7, M3 firmware v1.8, FPGA design v1.1.0.0
A2F:   A2F firmware, version 1.8
A2F:   Device type: A2F060M3E-FG256
A2F:   MSS clock frequency: 40 MHz
A2F:   Fabric clock frequency: 20 MHz
A2F:   Fast delay calibration: 1330 cycles per 100uS
A2F:   eNVM: 128 KB (00000000 - 00020000)
A2F:   eSRAM: 16 KB (20000000 - 20004000)
A2F:   Extram start: 20002750
A2F:   Last reset cause: PWRUP
A2F:   VMON flags: 00000000
RUPG:  booting from image 0 (confirmed)
```

After a power cycle, U-Boot reports that there is no reliable upgrade in progress and the confirmed image set is being used. The confirmed image set index is 0.

```
In:    serial
Out:   serial
Err:   serial
Net:   FEC0: 00:18:49:02:b2:06, FEC1: 00:18:49:02:b2:07
FEC0, FEC1
Hit any key to stop autoboot:  0
Creating 1 MTD partitions on "spi0":
0x0000001c0000-0x000002000000 : "mtd=5"
UBI: attaching mtd1 to ubi0
UBI: MTD device 1 is write-protected, attach in read-only mode
UBI: physical eraseblock size:   262144 bytes (256 KiB)
UBI: logical eraseblock size:    262016 bytes
UBI: smallest flash I/O unit:    1
```

```
UBI: VID header offset:          64 (aligned 64)
UBI: data offset:                128
UBI: attached mtd1 to ubi0
UBI: MTD device name:            "mtd=5"
UBI: MTD device size:            30 MiB
UBI: number of good PEBs:        121
UBI: number of bad PEBs:         0
UBI: max. allowed volumes:       128
UBI: wear-leveling threshold:    4096
UBI: number of internal volumes: 1
UBI: number of user volumes:     1
UBI: available PEBs:             0
UBI: total number of reserved PEBs: 121
UBI: number of PEBs reserved for bad PEB handling: 0
UBI: max/mean erase counter: 28/20
UBIFS: read-only UBI device
UBIFS: mounted UBI device 0, volume 0, name "boot"
UBIFS: mounted read-only
UBIFS: file system size:   28297728 bytes (27634 KiB, 26 MiB, 108 LEBs)
UBIFS: journal size:       2096129 bytes (2047 KiB, 1 MiB, 6 LEBs)
UBIFS: media format:       w4/r0 (latest is w4/r0)
UBIFS: default compressor: LZO
UBIFS: reserved for root:  1398497 bytes (1365 KiB)
RUPG:  upgrade file is not present, skipping
Loading file 'uImage.0' to addr 0x42000000 with size 1503000 (0x0016ef18)...
Done
Loading file 'rfs.0' to addr 0x46000000 with size 2739074 (0x0029cb82)...
Done
## Booting kernel from Legacy Image at 42000000 ...
   Image Name:   Linux-2.6.34.8-chmm700
   Created:      2015-04-20  17:35:21 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    1502936 Bytes =  1.4 MB
   Load Address: 40008000
   Entry Point:  40008000
   Verifying Checksum ... OK
## Loading init Ramdisk from Legacy Image at 46000000 ...
   Image Name:   ChMM700 RFS 1.2.0
   Created:      2017-10-24  13:50:43 UTC
   Image Type:   ARM Linux RAMDisk Image (gzip compressed)
```

```
   Data Size:    3092645 Bytes =  2.9 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...

init started: BusyBox v1.25.1 (2017-10-24 08:49:58 CDT)
/etc/rc: Current image set is 0
/etc/rc: Mounting filesystems...
/etc/rc: Mounted /proc
/etc/rc: Mounted /dev/pts
/etc/rc: Mounted /tmp
/etc/rc: Mounted /sys
/etc/rc: Checking the reliable upgrade watchdog timer...inactive
/etc/rc: Attaching ubi0:user...done
/etc/rc: Mounting  ubi0:user...done
/etc/rc: Attaching ubi1:boot...done
/etc/rc: Mounting  ubi1:boot...done
/etc/rc: Extracting application package /boot/app.0...done
/etc/rc: Mounted /0/var to /var
/etc/rc: Finalizing successful A2F upgrade
A2F:   SPICOMM protocol v1.7, M3 firmware v1.8, FPGA design v1.1.0.0
/etc/rc: Mounted /0/etc to /etc
/etc/rc: Setting timezone to MSK-3
/etc/rc: Placed /var/tmp to ram disk
/etc/rc: Setting hostname chmm~111
/etc/rc: Started syslogd and klogd
/etc/rc: Calling /etc/rc.carrier7-hpdl
/etc/readhwaddr: Board Hardware Address: 0x12
/etc/netconfig: /etc/hosts has valid chmm~111 192.168.1.111 entry
/etc/netconfig: Updating /etc/profile.sentry with IP settings
/etc/netconfig: ifconfig eth0 192.168.1.111 netmask 255.255.255.0
/etc/netconfig: ifconfig usb0 192.168.0.111 netmask 255.255.255.0
/etc/netconfig: route add default gw 192.168.1.253
/etc/netconfig: Starting /bin/inetd...
/etc/netconfig: Starting time synchronization with 192.168.1.253 rdate
/etc/rc.carrier7-hpdl: Updating /etc/profile.sentry with specific settings
```

```
/etc/rc.carrier7-hpdl: RC2 daemons not started by request
chmm~111 login:
```

Finally, the user checks the status of the reliable upgrade, by calling *rupgrade –s*.

```
# rupgrade -s
<INFO> Last upgrade log:
<INFO> Write storage state: not available -> available [ALLOW]
<INFO> Resetting UBIFS cache
<INFO> Burning image:<INFO> 524288 bytes were written from file
"/tmp/sentry.chmm700.u-boot" to dev 1[1]
<INFO> Cleaning image "/var/upgrade/boot/uImage.1"
<INFO> Burning image:<INFO> 1502912 bytes were written from file
"/tmp/sentry.chmm700.kernel" to dev 2[1]
<INFO> Cleaning image "/var/upgrade/boot/rfs.1"
<INFO> Burning image:<INFO> 2697316 bytes were written from file
"/tmp/sentry.chmm700.rfs" to dev 4[1]
<INFO> Burning image:<INFO> 2280508 bytes were written from file
"/tmp/sentry.chmm700.app" to dev 8[1]
<INFO> Write storage state: available -> not available [ALLOW]
<INFO> Erasing user data on candidate bank
<INFO> Copy ssh keys
<INFO> Write confirmed: 0 -> 0 [ALLOW]
<INFO> Write candidate: 1 -> 1 [ALLOW]
```

The log contains no ‘confirm’ message; the confirmed and candidate image set numbers are the same they were before reliable upgrade initiation.

```
<INFO> Write upgrade watchdog: 0 -> 1 [ALLOW]
<INFO> Last upgrade stages log:
0 (initial): PASSED
1 (before storage mount): PASSED
2 (after storage mount): PASSED
3 (after marking image set invalid): PASSED
4 (after u-boot burn): PASSED
5 (after kernel burn): PASSED
6 (after rootfs burn): PASSED
7 (after storage umount): PASSED
8 (after cleaning previous apps set): PASSED
9 (after writting new apps set): PASSED
10 (after sync): PASSED
11 (after processing var and etc): PASSED
133: Executing: /sbin/backend -s
133: System returned 3072, exec result 12.
<INFO> Last upgrade stage from HAL: 12
```

#

7.10.4 Example 4

This example shows a reliable upgrade where image installation and activation are separated. Initially, the new images are installed into the other image set via *rupgrade* with the option *--install-only*. Later, the other image set is activated via the command *backend --activate*.

The default copy mode is used (copy SSH keys and */var/nvdata*). All four images are assumed to be already in */tmp*.

```
# rupgrade --install-only --skip-checksums -k file:///tmp/sentry.chmm700.kernel
-r file:///tmp/sentry.chmm700.rfs -a file:///tmp/sentry.chmm700.app -u
file:///tmp/sentry.chmm700.u-boot
1773: Executing: /sbin/backend --get-wdt-state
1773: System returned 0, exec result 0.
<INFO> Downloading file "file:///tmp/sentry.chmm700.u-boot"
<INFO> Downloading file "file:///tmp/sentry.chmm700.kernel"
<INFO> Downloading file "file:///tmp/sentry.chmm700.rfs"
<INFO> Downloading file "file:///tmp/sentry.chmm700.app"
<INFO> Initiating partial upgrade:
<INFO>  * uboot "/tmp/sentry.chmm700.u-boot"
<INFO>  * kernel "/tmp/sentry.chmm700.kernel"
<INFO>  * rfs "/tmp/sentry.chmm700.rfs"
<INFO>  * app "/tmp/sentry.chmm700.app"
<INFO>  + install only
<INFO> Calling backend to handle partial upgrade "/sbin/backend --uboot
/tmp/sentry.chmm700.u-boot --kernel /tmp/sentry.chmm700.kernel --rootfs
/tmp/sentry.chmm700.rfs --application /tmp/sentry.chmm700.app --install-only"
1773: Executing: /sbin/backend --uboot /tmp/sentry.chmm700.u-boot --kernel
/tmp/sentry.chmm700.kernel --rootfs /tmp/sentry.chmm700.rfs --application
/tmp/sentry.chmm700.app --install-only
1777: Executing: /etc/hook/base_script.sh --stage 1 --silent
1777: System returned 0, exec result 0.
<INFO> Write storage state: not available -> available [ALLOW]
1777: Executing: mkdir -p /var/upgrade/boot
1777: System returned 0, exec result 0.
<INFO> Resetting UBIFS cache
1777: Executing: /sbin/eraseall /dev/mtd8
Erasing 256 Kibyte @ 0 --  0 % complete.
1777: System returned 0, exec result 0.
1777: Executing: /etc/hook/base_script.sh --stage 2 --silent
1777: System returned 0, exec result 0.
1777: Executing: /etc/hook/base_script.sh --stage 3 --silent
```

```
1777: System returned 0, exec result 0.
<INFO> Burning image:..
<INFO> 524288 bytes were written from file "/tmp/sentry.chmm700.u-boot" to dev
1[0]
<INFO> U-Boot checksum test after writing image: succeeded
1777: Executing: eraseall /dev/mtd4 >/dev/null 2>/dev/null
1777: System returned 0, exec result 0.
1777: Executing: dd if=/dev/mtd6 of=/dev/mtd4 >/dev/null 2>/dev/null
1777: System returned 0, exec result 0.
1777: Executing: /etc/hook/base_script.sh --stage 4 --silent
1777: System returned 0, exec result 0.
<INFO> Cleaning image "/var/upgrade/boot/uImage.0"
<INFO> Burning
image:.........................................................................
.........................................................................
.........................................................................
.........................................................................
.............................................................
<INFO> 1541536 bytes were written from file "/tmp/sentry.chmm700.kernel" to dev
2[0]
<INFO> Kernel checksum test after writing image: succeeded
1777: Executing: /etc/hook/base_script.sh --stage 5 --silent
1777: System returned 0, exec result 0.
<INFO> Cleaning image "/var/upgrade/boot/rfs.0"
<INFO> Burning
image:.........................................................................
.........................................................................
.........................................................................
.........................................................................
.........................................................................
.........................................................................
.........................................................................
.........................................................................
.........................................................................
.........................................
<INFO> 3092716 bytes were written from file "/tmp/sentry.chmm700.rfs" to dev
4[0]
<INFO> RFS checksum test after writing image: succeeded
1777: Executing: /etc/hook/base_script.sh --stage 6 --silent
1777: System returned 0, exec result 0.
1777: Executing: /etc/hook/base_script.sh --stage 7 --silent
1777: System returned 0, exec result 0.
<INFO> Burning
image:.........................................................................
.........................................................................
.........................................................................
```

```
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................
<INFO> 2430869 bytes were written from file "/tmp/sentry.chmm700.app" to dev
8[0]
<INFO> APP checksum test after writing image: succeeded
1777: Executing: /etc/hook/base_script.sh --stage 8 --silent
1777: System returned 0, exec result 0.
1777: Executing: if [ -f /var/upgrade/boot/app.1 ]; then mv -f
/var/upgrade/boot/app.1 /1/.app; mv -f /1/.app /1/sentry.chmm700.app; fi
1777: System returned 0, exec result 0.
1777: Executing: if [ -f /var/upgrade/boot/app.0 ]; then mv -f
/var/upgrade/boot/app.0 /0/.app; mv -f /0/.app /0/sentry.chmm700.app; fi
1777: System returned 0, exec result 0.
<INFO> Write storage state: available -> not available [ALLOW]
<INFO> Successfully re-mounted ubi device "ubi1:boot" as "/boot" in read-only
mode
1777: Executing: /etc/hook/base_script.sh --stage 9 --silent
1777: System returned 0, exec result 0.
1777: Executing: /etc/hook/base_script.sh --stage 10 --silent
1777: System returned 0, exec result 0.
<INFO> Erasing user data on candidate bank
1777: Executing: (cd /0 ; rm -rf var etc ; mkdir -p var/nvdata etc/ssh )
1777: System returned 0, exec result 0.
<INFO> Copy ssh keys
1777: Executing: cp -a /1/etc/ssh /0/etc
1777: System returned 0, exec result 0.
1777: Executing: cp -a /1/var/nvdata /0/var
1777: System returned 0, exec result 0.
1777: Executing: /etc/hook/base_script.sh --stage 11 --silent
1777: System returned 0, exec result 0.
1773: System returned 0, exec result 0.
```

#**backend --activate**

```
1843: Executing: /etc/hook/base_script.sh --stage 12 --silent
1843: System returned 0, exec result 0.
1843: Executing: dd if=/dev/mtd8 of=/dev/null >/dev/null 2>/dev/null
1843: System returned 0, exec result 0.
<INFO> Write confirmed: 1 -> 1 [ALLOW]
<INFO> Write candidate: 0 -> 0 [ALLOW]
```

```
<INFO> Write upgrade watchdog: 0 -> 1 [ALL

U-Boot 2009.08 (Oct 24 2017 - 11:44:23)

Freescale i.MX28 family
CPU:   297 MHz
BUS:   99 MHz
EMI:   130 MHz
GPMI:  24 MHz
I2C:   ready
DRAM:  128 MB
SFGEN: S25FL512S detected, total size 64 MB
A2F:   SPICOMM protocol v1.7, M3 firmware v1.8, FPGA design v1.1.0.0
A2F:   A2F firmware, version 1.8
A2F:   Device type: A2F060M3E-FG256
A2F:   MSS clock frequency: 40 MHz
A2F:   Fabric clock frequency: 20 MHz
A2F:   Fast delay calibration: 1330 cycles per 100uS
A2F:   eNVM: 128 KB (00000000 - 00020000)
A2F:   eSRAM: 16 KB (20000000 - 20004000)
A2F:   Extram start: 20002750
A2F:   Last reset cause: RUWD
A2F:   VMON flags: 00000000
RUPG:  booting from image 0 (candidate)
In:    serial
Out:   serial
Err:   serial
Net:   FEC0: 00:18:49:02:b2:06, FEC1: 00:18:49:02:b2:07
FEC0, FEC1
Hit any key to stop autoboot:  0
RUPG:  reliable upgrade is in progress, skipping
Creating 1 MTD partitions on "spi0":
0x0000001c0000-0x000002000000 : "mtd=5"
UBI: attaching mtd1 to ubi0
UBI: MTD device 1 is write-protected, attach in read-only mode
UBI: physical eraseblock size:   262144 bytes (256 KiB)
UBI: logical eraseblock size:    262016 bytes
UBI: smallest flash I/O unit:    1
UBI: VID header offset:          64 (aligned 64)
UBI: data offset:                128
```

```
UBI: attached mtd1 to ubi0
UBI: MTD device name:            "mtd=5"
UBI: MTD device size:            30 MiB
UBI: number of good PEBs:        121
UBI: number of bad PEBs:         0
UBI: max. allowed volumes:       128
UBI: wear-leveling threshold:    4096
UBI: number of internal volumes: 1
UBI: number of user volumes:     1
UBI: available PEBs:             0
UBI: total number of reserved PEBs: 121
UBI: number of PEBs reserved for bad PEB handling: 0
UBI: max/mean erase counter: 127/111
UBIFS: read-only UBI device
UBIFS: mounted UBI device 0, volume 0, name "boot"
UBIFS: mounted read-only
UBIFS: file system size:   28297728 bytes (27634 KiB, 26 MiB, 108 LEBs)
UBIFS: journal size:       2096129 bytes (2047 KiB, 1 MiB, 6 LEBs)
UBIFS: media format:       w4/r0 (latest is w4/r0)
UBIFS: default compressor: LZO
UBIFS: reserved for root:  1398497 bytes (1365 KiB)
Loading file 'uImage.0' to addr 0x42000000 with size 1541536 (0x001785a0)...
Done
Loading file 'rfs.0' to addr 0x46000000 with size 3092716 (0x002f30ec)...
Done
## Booting kernel from Legacy Image at 42000000 ...
   Image Name:   Linux-2.6.34.8-chmm700
   Created:      2017-10-24  16:44:21 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    1541472 Bytes =  1.5 MB
   Load Address: 40008000
   Entry Point:  40008000
   Verifying Checksum ... OK
## Loading init Ramdisk from Legacy Image at 46000000 ...
   Image Name:   ChMM700 RFS 1.2.0
   Created:      2017-10-24  16:39:38 UTC
   Image Type:   ARM Linux RAMDisk Image (gzip compressed)
   Data Size:    3092652 Bytes =  2.9 MB
   Load Address: 00000000
   Entry Point:  00000000
```

```
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...

init started: BusyBox v1.25.1 (2017-10-24 11:38:54 CDT)
/etc/rc: Current image set is 0
/etc/rc: Mounting filesystems...
/etc/rc: Mounted /proc
/etc/rc: Mounted /dev/pts
/etc/rc: Mounted /tmp
/etc/rc: Mounted /sys
/etc/rc: Checking the reliable upgrade watchdog timer...activated
<INFO> Write upgrade watchdog: 1 -> 2 [ALLOW]
/etc/rc: Attaching ubi0:user...done
/etc/rc: Mounting  ubi0:user...done
/etc/rc: Attaching ubi1:boot...done
/etc/rc: Mounting  ubi1:boot...done
<INFO> Write upgrade watchdog: 1 -> 2 [ALLOW]
/etc/rc: Extracting application package /boot/app.0...done
/etc/rc: Mounted /0/var to /var
/etc/rc: Strobing the reliable upgrade WDT
<INFO> Write upgrade watchdog: 1 -> 2 [ALLOW]
92: Executing: /etc/hook/base_script.sh --stage 13 --silent
92: System returned 0, exec result 0.
/etc/rc: /etc appears to be empty ... restoring from /etc in RFS...
/etc/rc: Preserved ssh_host_dsa_key in /etc/ssh
/etc/rc: Preserved ssh_host_rsa_key in /etc/ssh
/etc/rc: Mounted /0/etc to /etc
/etc/rc: Setting timezone to MSK-3
/etc/rc: Placed /var/tmp to ram disk
/etc/rc: Setting hostname chmm~110
/etc/rc: Started syslogd and klogd
/etc/rc: Calling /etc/rc.carrier7-hpdl
/etc/readhwaddr: Board Hardware Address: 0x10
/etc/netconfig: /etc/hosts updated with chmm~110 192.168.1.110 entry
/etc/netconfig: Updating /etc/profile.sentry with IP settings
/etc/netconfig: ifconfig eth0 192.168.1.110 netmask 255.255.255.0
/etc/netconfig: ifconfig usb0 192.168.0.110 netmask 255.255.255.0
```

```
/etc/netconfig: ifconfig eth1 172.16.2.110 netmask 255.255.255.0
/etc/netconfig: route add default gw 192.168.1.253
/etc/netconfig: Starting /bin/inetd...
/etc/netconfig: Starting time synchronization with 192.168.1.253 rdate
/etc/rc.carrier7-hpdl: Updating /etc/profile.sentry with specific settings
/etc/rc.carrier7-hpdl: RC2 daemons not started by request
/etc/rc: start_rc2_daemons=n - finishing reliable upgrade process now.
<INFO> Write confirmed: 1 -> 0 [ALLOW]
<INFO> Write upgrade_state: "in progress" (2) -> "confirmed" (4) [ALLOW]
<INFO> Write upgrade watchdog: 1 -> 0 [ALLOW]
chmm~110 login:
```

8 Customer support

If you are having problems with the Pigeon Point Chassis Manager or ChMM product, please contact your supplier for the Pigeon Point products with questions and problem reports.

If you have any questions about direct purchase of Pigeon Point products (one of the Pigeon Point Board Management Reference designs, for instance), please contact nVent; for guidance on such contacts, visit www.pigeonpoint.com/contact.html.

9 Revision History

This section lists the changes that have been made in each revision of this document, starting with the 1.0.0 release.

9.1 Release 1.0.0

- Initial revision of the document.

9.2 Release 1.0.1

- Section 2.4.1: adds an overall summary of the compatibility of the Pigeon Point Chassis Manager with the Pigeon Point Shelf Manager.
- Section 3.3: adds descriptions of configuration variables *PET_SNMP_VERSION* and *DHCP_REQUEST_OPTION_LIST*.

9.3 Release 1.1.0

- Sections 3.2.1, 3.4.4, 4.2, 6.3 and 6.4: change instances of the term “factory defaults” to alternate terms appropriate to each context.
- Section 3.3: configuration parameters added: *ALLOWED_CIPHER_SUITES*, *COOLING_EXHAUST_SENSOR_LIST*, *COOLING_INTAKE_SENSOR_LIST*, *COOLING_NO_ALL_FRUS_POWER_DOWN_IN_CRITICAL_ALERT*, *DEFAULT_HPM2_CLIENT_IP_ADDRESS*, *DEFAULT_HPM2_GATEWAY_IP_ADDRESS*, *DEFAULT_HPM2_NETMASK*, *DEFAULT_RMCP_IP_ADDRESS*, *DEFAULT_RMCP_IP_ADDRESS2*, *DEFAULT_RMCP_NETMASK2*, *HPDL_NETWORK_ELEMENT_ID_EEPROM_OFFSET*, *HPM2_ENABLE_CLIENT_SESSIONS*, *HPM2_ENABLE_INACTIVE_STATE_MANAGEMENT*, *HPM2_ENABLE_SHMM_INACTIVE_STATE*, *HPM2_CLIENT_MIRROR_CHANNEL*, *HPM2_NET_ADAPTER*, *IGNORE_FAILED_DIRECTED_POWER_DOWN*, *IPMB_LINK_ISOLATION_TIMEOUT*, *IPMB_LOCAL_FAILURE_RETRIES*, *ISOLATE_MUX_IGNORE_TIME*, *PROPAGATE_RMCP_ADDRESS*, *RMCP_PLUS_ONLY*, *RMCP_WITHOUT_SHELF_FRU*, *SAVE_CRASH_INFORMATION*, *WATCHDOG_TIMEOUT*. Parameter *COOLING_IGNORE_LOCAL_CONTROL* was removed, since it was not used in the Chassis Manager.
- New section **Error! Reference source not found.**: covers configuration considerations for an HPM.2 channel.
- New section 3.4.7 and updated section 3.14: covers configuration of IPv6-based services of the Chassis Manager.
- New section 3.8 covers how to configure local sensors for a non-HPDL chassis.

- Section 4.5.3: updates Reboot Reason Sensor values list, along with some recommendations about watchdog timer settings and a description of the file `/var/nvdata/reboot_reason.log`.
- Section 6.2: updates the procedure for saving environment variables.
- New section 7.4: adds a description of the `user_partition` and `flash_reset` variables in U-Boot and details for 'user' partition format changes.
- Section 8: contact information updated.

9.4 Revision 1.2.0

- Overall: implements branding changes to reflect the acquisition of Pigeon Point Systems by Pentair.
- Section 3.2.1: adds a description of the `log_remote2` variable.
- Section 3.3: adds the description of new configuration variables `DISABLE_SESSION_AUDIT_EVENTS`, `PREFER_CHASSIS_HPDL_SDRS_FROM_FILE`, `PET_PORT`. Fixes the default value of `MAX_PENDING_IPMB_REQUESTS` configuration variable.
- Section 3.4: adds notes on the risks of doing manual changes in the ChMM network configuration separately from the Chassis Manager.
- Section 3.4.6: adds information about the new DHCPv4 client.
- Section 3.14: adds information about configuring IPv6 support for various network services.
- Section 4.6.1: adds coverage for the SmartFusion A2F200 cSoC device. Chassis Manager Release 1.2.0 supports ChMM-700R modules that use a SmartFusion A2F200 or A2F060 device and Carrier boards that use SmartFusion A2F200 or A2F060 devices to implement a radial System IPMB topology.
- Section: adds information on how to cause a switchover to be initiated if the number of failed I^2C multiplexer reset attempts exceeds a certain threshold.
- Section 5.2: adds a description of `–r` option to enable radial IPMB link monitoring in the `ipmb_traced` daemon.
- Section 7.6: adds the `--clone` option for the `rupgrade` utility, describes the `--install-only` option of the `rupgrade` utility and the `--activate` option of the `backend` utility, describes the `–copy-home` option of the `rupgrade` utility.
- New section 7.9: describes the `imls` utility.
- New section 7.10.4: adds an example of a reliable upgrade scenario using the `--install-only` option.
- Section 8: updates contact information.

9.5 Revision 1.2.1

- Overall: implements branding changes to reflect the split of nVent from Pentair.

9.6 Revision 1.2.2

- Overall: deletes all the references to HPM.2 functionality.
- Section 3.3: deletes descriptions of the obsolete configuration variables: *DEFAULT_HPM2_CLIENT_IP_ADDRESS*, *HPM2_ENABLE_CLIENT_SESSIONS*, *HPM2_ENABLE_INACTIVE_STATE_MANAGEMENT*, *HPM2_ENABLE_SHMM_INACTIVE_STATE*, *HPM2_CLIENT_MIRROR_CHANNEL*, *HPM2_NET_ADAPTER*.
- Section 3.12.2: an example of the external event handler script is added.

9.7 Revision 1.3.0

- New section 3.16: introduces support of VPX Control/GPIO signals.
- Section 7.7: adds information on the SmartFusion (A2F) cSoC upgrade of a system that supports operating the System IPMB in FAST-MODE (400 kHz).
- New section 3.4.3: Gigabit Ethernet support description. Also, corresponding variables have been added to Table 3.