Intel® NetStructure™ ZT 7102 Chassis Management Module

Technical Product Specification

August 2004

Order Number: 273770-017

Contents

Figures

Tables

Revision History

Date	Revision	Description
August 2004	017	Changes related to V4.11h firmware Added new BIST events for RTC and FRU checking. Added new Chassis Slot event for IPMI capable Added information on /etc/version checking Added infomation on chassis FRU validity checking.
April 2004	016	Changes related to V4.11c firmware Added rdate feature information
March 2004	015	Changes related to V4.11b frimware verison
March 2004	014	Changes related to V4.11a firmware version Changes to Software Update section Slotcontrol description Changes to Heatlh Event Strings
October 2003	010	Changes related to V4.0x firmware version New Password Reset Implementation PSInhibit Power Sequencing Policies Updated Warranty and Customer Support information.
April 2003	005	V3.00a Release

1.0 Introduction

1.1 Overview

The Intel® NetStructure™ ZT 7102 is a 3U, single-slot Chassis Management Module (CMM) intended for use with PICMG* 2.1, 2.16, and 2.9-compliant systems (the CompactPCI* Hot Swap, Packet Switching Backplane, and System Management specifications, respectively). This document details the features and specifications of the ZT 7102.

The ZT 7102 plugs into a dedicated slot in compatible systems. It provides centralized management and alarming for up to 21 node and/or fabric slots as well as for system power supplies and fans. The ZT 7102 may be paired with a backup for use in high-availability applications.

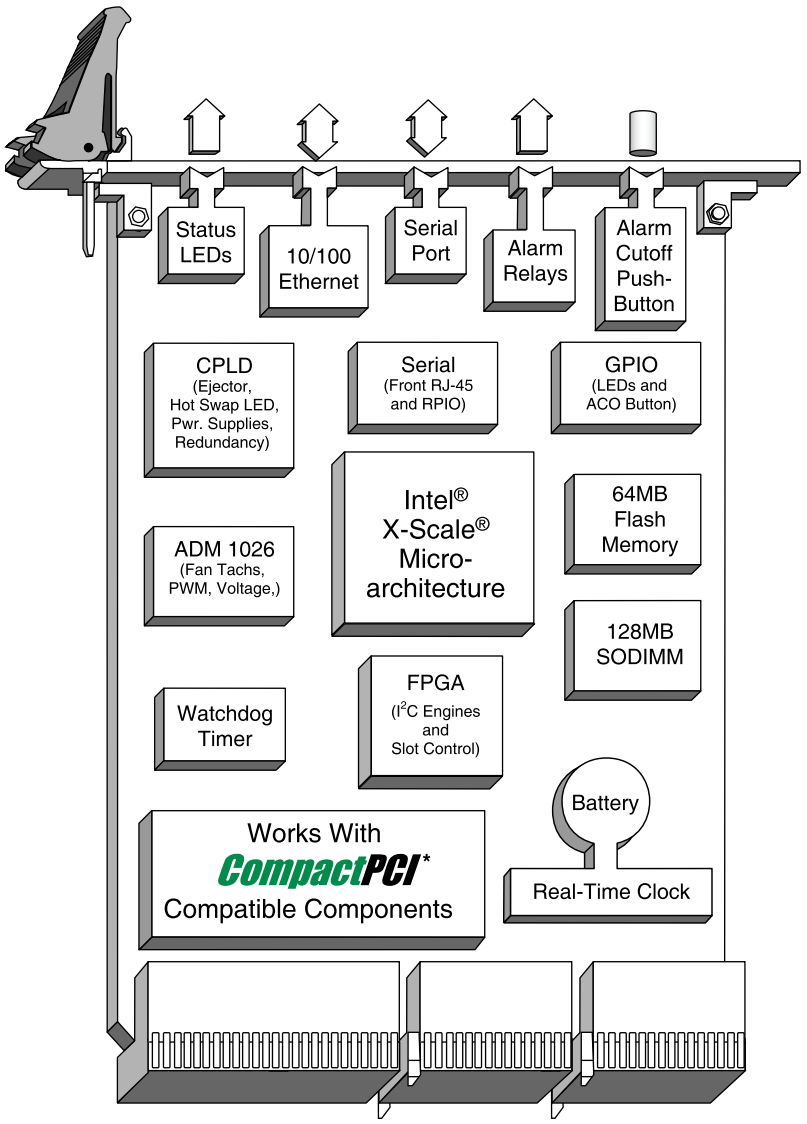
The ZT 7102 is a special purpose single board computer with its own CPU, memory, PCI bus, operating system, and peripherals. The ZT 7102 monitors and configures Intelligent Platform Management Interface (IPMI)-based components in the chassis. When thresholds such as temperature and voltage are crossed or a failure occurs, the CMM captures these events, stores them in an event log, sends SNMP traps, and drives the Telco alarm relays and alarm LEDs. The CMM can query FRU information (such as serial number, model number, manufacture date, etc.), detect presence of components (such as fan tray, CPU board, etc.), and perform health monitoring of each component. In addition, the CMM controls the power-up sequencing of each node board and the power to each slot via the BD_SEL# signal.

The ZT 7102 can also be used at a limited degree to manage non-IPMI based chassis components.

1.2 ZT 7102 Features

- High-density 3U X 1-slot form factor
- Compatible with PICMG* 2.1, 2.16, and 2.9-compliant chassis and components
- Monitors via the Intelligent Platform Management Bus (IPMB) protocol
- Provides isolated IPMB signals for each slot for maximum security and reliability
- Manage through the command line interface, SNMP v1/v3, or Remote Procedure Call (RPC)
- Hot-add/hot-swap support for IPMI-based, field-replaceable components
- µDB15 Telco Alarm Interface at the front panel
- Critical, Major, and Minor alarm LEDs at the front panel
- CMM status and hot swap LEDs at the front panel
- Monitors backplane voltages and status for up to eight power supplies
- Monitors system temperature sensors
- Monitors system fan tray presence
- Monitors tachometers for up to 16 system fans
- Monitors sensors on PICMG 2.9 compliant single board computers
- Power state control for single board compute blades

Figure 1. Functional Block Diagram



1.3 Terms Used in This Document

Table 1. Glossary

Acronym	Description
CMM	Chassis Management Module
CLI	Command Line Interface
FRU	Field Replaceable Unit
IPMI	Intelligent Platform Management Interface
IPMB	Intelligent Platform Management Bus
MIB	Management Information Base
MIB II	RFC1213 - A standard Management Information Base for Network Management
SEL	System Event Log
SBC	Single Board Computer
SNMP	Simple Network Management Protocol
SNMP v.1	SNMP version 1
SNMP v.3	SNMP version 3
SDR	Sensor Data Record

2.0 Hardware Specifications

2.1 Overview

The chassis management module (CMM) is intended to operate as a chassis management building block in a CompactPCI chassis. The CMM uses a unique backplane interface pinout and requires a dedicated slot within the chassis.

Caution: The ZT 7102 plugs into a dedicated chassis management slot. System components will be damaged if a standard 3U board is plugged into the chassis management slot.

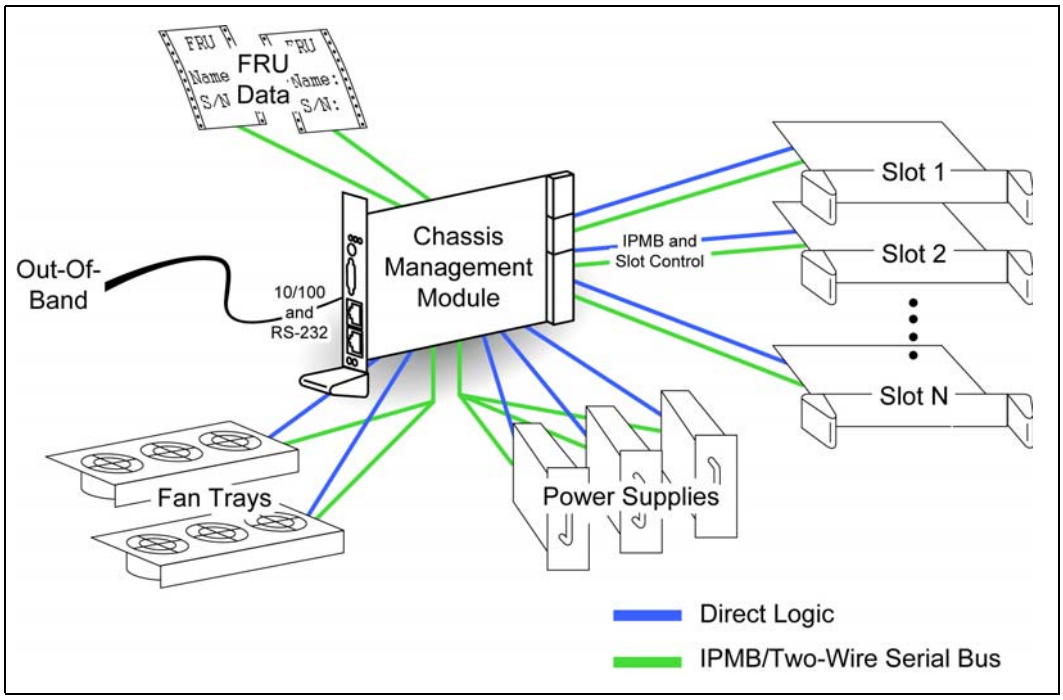
Intelligent Platform Management Buses (IPMBs) are the primary management connection between components. A unique star topology radiating from the CMMs is used to provide reliability, flexibility and security. In some cases, direct logic is used to support non-intelligent devices and to eliminate active components on the backplane and chassis.

The CMM provides 30 master/slave two-wire serial bus (2WSB) interfaces. The 2WSBs are electrically equivalent to I2C*. As such, all outputs are open drain, so they are driven low and passively pulled high. The 2WSB interfaces support multi-master operation on the bus, however they do not support being the target of a read transaction from another master device. The 2WSB interfaces are used by the CMM as IPMBs or non-intelligent management busses. Refer to the individual sections for specific usage information. The 2WSB interfaces are used for:

- Up to 21 general-purpose slots (21 independent 2WSBs, 2WSB_S0 - 2SB_S20)
- Power supplies (two buses, 2WSB_PS0 and 2WSB_PS1)
- Fan trays (one bus, 2WSB_FT)
- Chassis sensors (one bus, 2WSB_CS)
- Chassis FRU modules (two independent buses, 2WSB_CF0 and 2WSB_CF1)
- Redundant CMM link (one bus for bi-directional communication, 2WSB_RCMM)

2.2 System Architecture

Figure 2. System Management Architecture



2.3 Backplane Connections

2.3.1 Slot Connections

The slots are connected to the CMM via an IPMB defined in the PICMG* 2.9 specification. The CMM also has connections for BD_SEL# and HEALTHY# signals to each slot. Each slot has an independent IPMB, BD_SEL#, and HEALTHY#. The CMM supports up to 21 general-purpose node slots.

The IPMB is used for IPMI management communication between the CMM and the board in the slot. Each board that is to be managed by the CMM is required to support the IPMB as a management channel.

BD_SEL# is used in CompactPCI slots to control board power and to detect board presence. The CMM provides a bi-directional open drain driver for each BD_SEL#. The board installed in the slot provides a pull-up resistor (1.2 KΩ ± 5% per PICMG* 2.0). The CMM provides a weak pulldown resistor (and a diode clamp). When not asserted, BD_SEL# can be read to determine if a board is present in the slot. If BD_SEL# is high, a board is present. If BD_SEL# is not asserted a board is not present. Asserting BD_SEL# allows the board to power up.

HEALTHY# is an input to the CMM. The CMM provides a pull-up resistor. The board in the slot asserts HEALTHY# based on board power being good and optionally other board-specific requirements. BD_SEL# must be asserted for HEALTHY# to be asserted. When BD_SEL# is asserted, and a board is removed, HEALTHY# will be deasserted.

2.3.2 Chassis Sensor Connections

Chassis sensors are connected to the CMM via one two-wire serial bus.

2.3.3 Chassis FRU Device Connections

Chassis sensors are connected to the CMM via one two-wire serial bus. The CMM provides two 2WSB interfaces for chassis FRU modules. Each chassis FRU module is on a dedicated 2WSB in order to provide redundant access to vital chassis information (e.g., the physical location of the chassis). The backplane FRU storage device used MUST be an Atmel* AT24C16 or other 24C16-compatible device.

2.3.4 Redundancy

The CMM supports redundant operation with automatic failover under hardware or software control. The following hardware interfaces exist for the support of redundancy and automatic failover:

- Cross-connected CMM present inputs (PRES_I#) and outputs (PRES_O#)
- Cross-connected CMM healthy inputs (HLY_I#) and outputs (HLY_O#)
- Cross-connected negotiation inputs (NEG_I) and outputs (NEG_O)

The active CMM monitors its PRES_I# and HLY_I# inputs to determine if it has a healthy, standby CMM. The active CMM deasserts its HLY_O# output to trigger a failover to the standby CMM.

The cross-connected negotiation signals are used to assure that only one CMM is active at a time. At anytime, the standby CMM can trigger a failover by driving its NEG_O output low.

2.4 Power Modules

Power supply sleds are connected to the CMM via two IPMBs. Each power supply sled connects to one IPMB. Multiple power supplies can share a single IPMB. The CMM also provides independent DEG#, FAIL#, and INH# signals as defined in PICMG* 2.11 for up to eight power supplies. The CMM will communicate with intelligent supplies via the IPMBs. Non-intelligent supplies are supported via the DEG# (degrade), FAIL# (fail), and INH# (inhibit) signals.

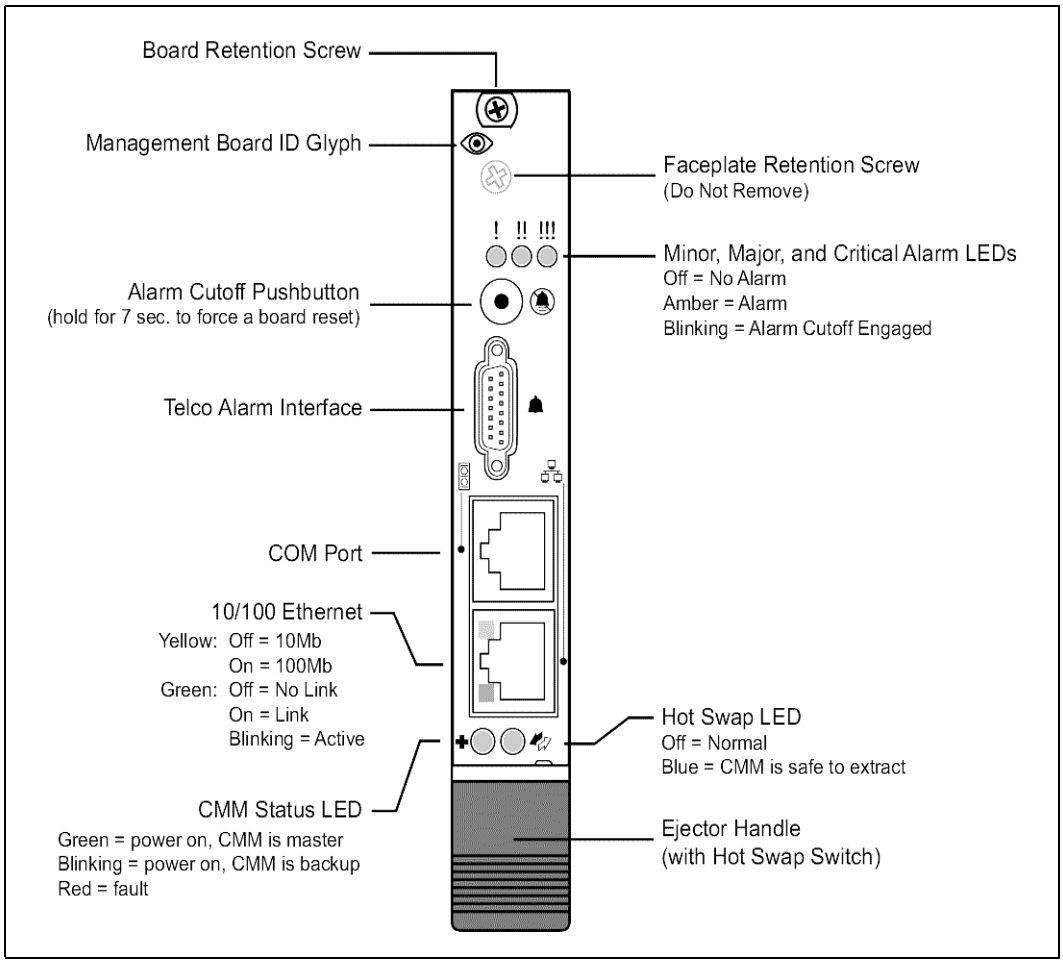
The CMM uses INH# rather than EN# to control the power supplies. The EN# pin is grounded on the backplane to signal to a power supply when it is fully seated in its connector.

2.5 Fan Modules

Fan trays are connected to the CMM via one two-wire serial bus (2WSB). Intelligent fan trays communicate with the CMM via the IPMB. To support non-intelligent fan trays, the CMM also provides independent fan tachometer inputs for up to 16 fans, fan tray present inputs for up to four fan trays, and four fan speed outputs (four buffered copies of a single PWM). Non-intelligent fan trays are monitored and controlled via the fan tachometer inputs and the fan speed output.

2.6 Face Plate

Figure 3. Face Plate



2.6.1 Faceplate Features

The following features are found on the faceplate of the ZT 7102:

Table 2. Faceplate Features

Feature	Purpose
Minor, Major, and Critical Alarm LEDs	These LEDs indicate the presence of various event-triggered alarms. The LEDs function as follows: Off = no alarm triggered. Amber = alarm triggered. Blinking = alarm cutoff (ACO) is activated.
Alarm Cutoff (ACO) push button	This push button toggles the ACO state. When ACO is activated, the active alarm LEDs blink and all of the alarm relays are deactivated. This button does not clear alarms. The ZT 7102 can be reset by pressing and holding the ACO button for about five seconds.
Telco Alarm Interface (µDB15 connector)	This interface relays alarm signals to off-board equipment. Contact Intel for a compatible cable (µDB15 to standard DB15). See Appendix G, "Customer Support." for details.
COM Port (RJ-45 connector)	This serial port may be used to access the CMM's Command Line Interface (CLI).
Ethernet Port (RJ-45 with LEDs)	This port provides an out-of-band 10/100 Ethernet connection. The port's integrated LEDs function as follows: Yellow: Off = 10 Mbit On = 100 Mbit Green: Off = No Link On = Link Blinking = Activity
CMM Status LED	This LED indicates CMM status as follows: Green = power is on and the board is acting as the master (active) CMM. Blinking green = power is on and the board is acting as the backup (standby) CMM. Red = the CMM needs attention (a critical problem exists).
Hot Swap LED	This LED indicates when it is safe to remove the CMM from a live (powered-on) chassis. The LED functions as follows: Off = the CMM is not ready to be removed from a live chassis. Blue = the CMM is ready to be removed from a live chassis.
Ejector with hot swap switch	The ejector functions as a handle and a lever for installing or removing the CMM. The ejector incorporates a switch that tells the CMM when the board is about to be removed from a system.

2.7 Mechanical

2.7.1 Board Dimensions and Weight

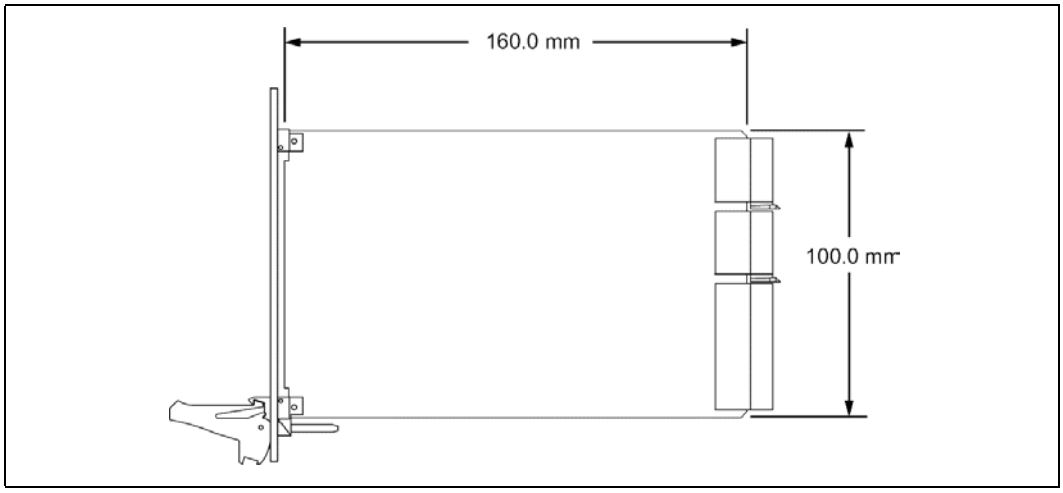
In a compatible enclosure, the ZT 7102 occupies a single 3U CMM slot. Mechanical dimensions are shown in Figure 4 and outlined in Table 3.

Table 3. Board Dimensions and Weight

PCB Dimensions:	100 mm x 160 mm x 1.6 mm
Board Dimensions:	3U x 4HP (one slot)
Weight:	7.7 ounces w/ 128 Mbyte SODIMM

2.7.2 PCB Dimensions

Figure 4. PCB Dimensions



2.8 Connectors

As shown in Figure 5, the ZT 7102 includes several connectors to interface to application-specific devices. A brief description of each connector is given in Table 4 below. A detailed description and pinout for the backplane connectors is given in the following sections.

Table 4. Connector Assignments

Connector	Function
J1, Table 5	Backplane Connector (110-pin, 2 mm x 2 mm, female)
J2, Table 6	Backplane Connector (55-pin, 2 mm x 2 mm, female)
J3, Table 7	Backplane Connector (55-pin, 2 mm x 2 mm, female)
J4	Connector reserved for test.
JA1, Table 10	Ethernet Connector (RJ-45, 8-pin)
J6, Table 11	Serial Port (RJ-45, 8-pin)

Table 4. Connector Assignments

Connector	Function
J7, Table 12	Telco Alarm Connector (uDB-15, 15-pin)
J8,	SODIMM Socket (144-pin)
J9	Connector reserved for test.

Figure 5. Connector Locations

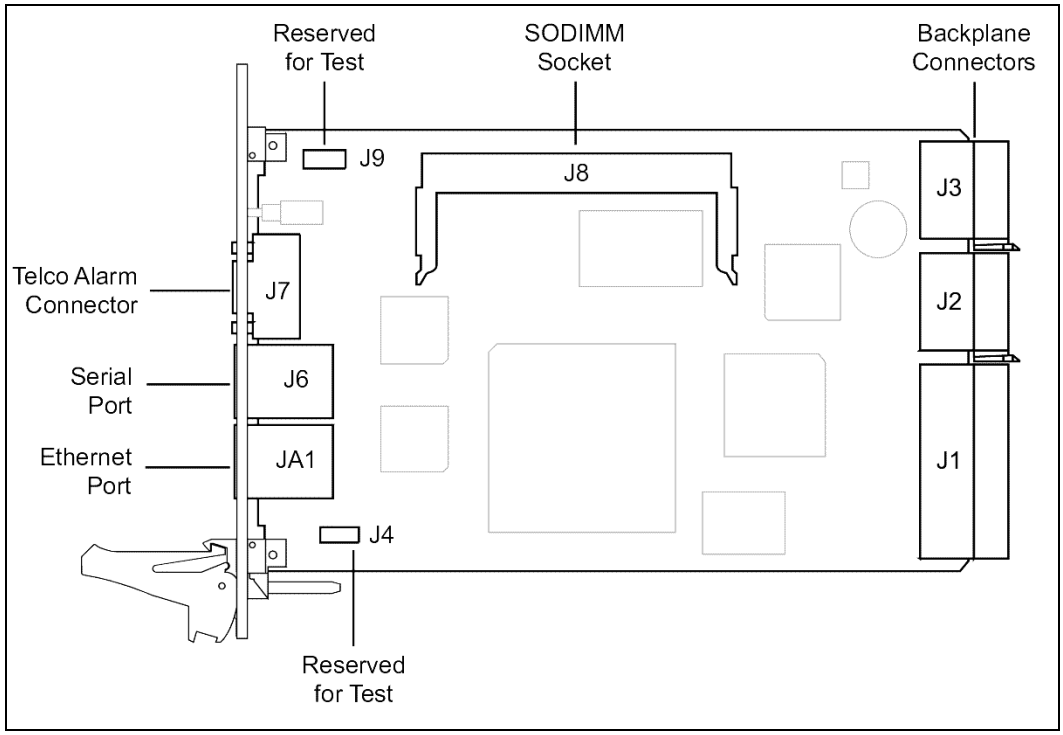
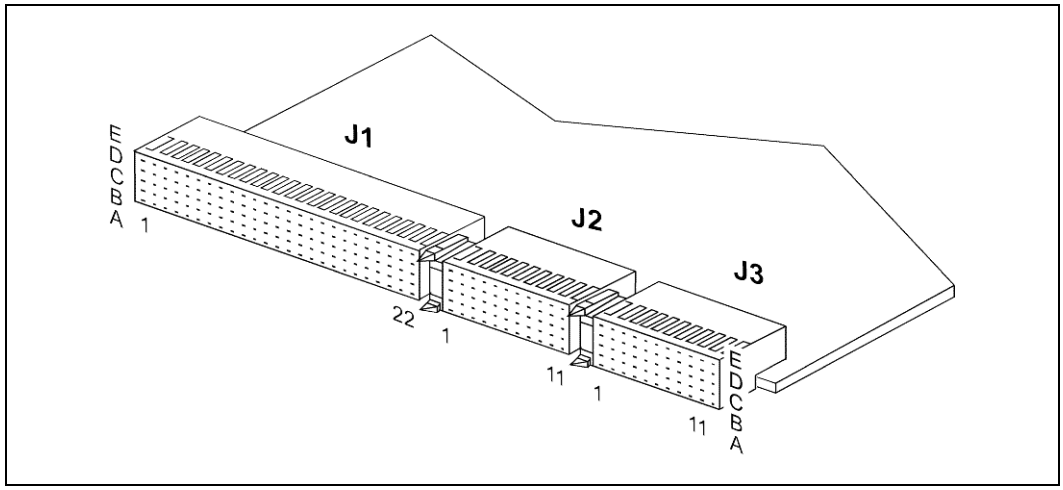


Figure 6. Backplane Connectors - Pin Locations



2.8.1 J1 Backplane Connector

J1 is a 110-contact, 2 mm x 2 mm female CompactPCI form-factor connector (AMP 352152-1). See Table 5 for pin definitions and Figure 6, “Backplane Connectors - Pin Locations” on page 23 for relative pin placement.

Table 5. J1 Connector Pinout

Pin#	A	B	C	D	E	F
22	PDEG0#	PDEG2#	GND	PDEG4#	PDEG6#	GROUND SHIELD
21	PDEG1#	PDEG3#	IPMB_PWR	PDEG5#	PDEG7#	
20	PFAIL0#	PFAIL2#	GND	PFAIL4#	PFAIL6#	
19	PFAIL1#	PFAIL3#	IPMB_PWR	PFAIL5#	PFAIL7#	
18	PINH0#	PINH2#	GND	PINH4#	PINH6#	
17	PINH1#	PINH3#	IPMB_PWR	PINH5#	PINH7#	
16	PS_SCL0	PS_SDA0	GND	PRES_I#	GA0	
15	PS_SCL1	PS_SDA1	RES	FANP0#	FANP2#	
14	FT_SCL	FT_SDA	GND	FANP1#	FANP3#	
13	CS_SCL	CS_SDA	RES	FANPWM0	FANPWM2	
12	CF_SCL0	CF_SDA0	GND	FANPWM1	FANPWM3	
11	CF_SCL1	CF_SDA1	RES	FANTK0	FANTK8	
10	RES	RES	GND	FANTK1	FANTK9	
9	RES	RES	RES	FANTK2	FANTK10	
8	STx	SRx	SRI	FANTK3	FANTK11	
7	SCTS	SRTS	SCD	FANTK4	FANTK12	
6	SDSR	SDTR	RES	FANTK5	FANTK13	
5	RPMAC#	RPMIC#	RES	FANTK6	FANTK14	
4	RPCR#	RPMAR#	RPMIR#	FANTK7	FANTK15	
3	BP_5V	BP_N12V	BP_12V	BP_3.3V	VIO	
2	SwTx+	SwTx-	GND	RpTx+	RpTx-	
1	SwRx+	SwRx-	GND	RpRx+	RpRx-	

NOTE: # Designates a low true signal. All signals interface to medium length pins on the backplane.

2.8.2 J2 Backplane Connector

J2 is a 55-contact, 2 mm x 2 mm female CompactPCI* form-factor connector (AMP* 352115-1). See Table 6 for pin definitions and Figure 6 for relative pin placement.

Table 6. J2 Connector Pinout

Pin#	A	B	C	D	E	F
11	N_SCL1	N_SDA1	N_SCL10	N_SDA10	N_SCL18	GROUND SHIELD
10	N_SCL2	N_SDA2	N_SCL11	N_SDA11	N_SDA18	
9	N_SCL3	N_SDA3	N_SCL12	N_SDA12	N_SCL19	
8	N_SCL4	N_SDA4	N_SCL13	N_SDA13	N_SDA19	
7	N_SCL5	N_SDA5	N_SCL14	N_SDA14	N_SCL20	
6	N_SCL6	N_SDA6	N_SCL15	N_SDA15	N_SDA20	
5	N_SCL7	N_SDA7	N_SCL16	N_SDA16	N_SCL21	
4	N_SCL8	N_SDA8	N_SCL17	N_SDA17	N_SDA21	
3	N_SCL9	N_SDA9	GND	NEG_O	HLY_O#	
2	R_SCL	IPMB_PWR	RES	IPMB_PWR	HLY_I#	
1	GND	R_SDA	CMM_SEL#	NEG_I	GND	

NOTE: # Designates a low true signal.
All signals interface to medium length pins on the backplane except as noted.
= Interfaces to long connector pins on the backplane.
= Interfaces to short connector pins on the backplane.

2.8.3 J3 Backplane Connector

J3 is a 55-contact 2 mm x 2 mm female CompactPCI form-factor connector (AMP 352115-1). See Table 7 for pin definitions and Figure 6 for relative pin placement.

Table 7. J3 Connector Pinout

Pin#	A	B	C	D	E	F
11	N_HLY1#	N_BDS1#	N_HLY10#	N_BDS10#	N_HLY18#	GROUND SHIELD
10	N_HLY2#	N_BDS2#	N_HLY11#	N_BDS11#	N_BDS18#	
9	N_HLY3#	N_BDS3#	N_HLY12#	N_BDS12#	N_HLY19#	
8	N_HLY4#	N_BDS4#	N_HLY13#	N_BDS13#	N_BDS19#	
7	N_HLY5#	N_BDS5#	N_HLY14#	N_BDS14#	N_HLY20#	
6	N_HLY6#	N_BDS6#	N_HLY15#	N_BDS15#	N_BDS20#	
5	N_HLY7#	N_BDS7#	N_HLY16#	N_BDS16#	N_HLY21#	
4	N_HLY8#	N_BDS8#	N_HLY17#	N_BDS17#	N_BDS21#	
3	N_HLY9#	N_BDS9#	IPMB_PWR	PIMP0#	RES	
2	RES	GND	RES	GND	RES	
1	IPMB_PWR	RES	PRES_O#	PIMP1#	IPMB_PWR	

NOTE: # Designates a low true signal.
All signals interface to medium length pins on the backplane except as noted.
= Interfaces to long connector pins on the backplane.
= Interfaces to short connector pins on the backplane.

2.8.4 Backplane Pin Descriptions

Table 8. Pin Type Definitions

Pin Type	Definition
OD	Open Drain
I	Input
I/O	Input/Output
O	Output

Table 9. Pin Descriptions (Sheet 1 of 2)

Name	Count	Type	Description
N_SCL[1..21]	21	OD	Node IPMI clock
N_SDA[1..21]	21	OD	Node IMPI data
PS_SCL[0..1]	2	OD	Power Supply Chassis IMPI clock
PS_SDA[0..1]	2	OD	Power Supply Chassis IMPI data
FT_SCL	1	OD	Fan Tray Chassis IMPI clock
FT_SDA	1	OD	Fan Tray Chassis IMPI data
CS_SCL	1	OD	Chassis Sensor IMPI clock
CS_SDA	1	OD	Chassis Sensor IMPI data
CF_SCL[0..1]	2	OD	Chassis FRU IMPI clock
CF_SDA[0..1]	2	OD	Chassis FRU IMPI data
R_SCL	1	OD	Redundant CMM serial clock
R_SDA	1	OD	Redundant CMM serial data
N_HLY#[1..21]	21	I	Node Healthy (0 = Node is healthy)
N_BDS#[1..21]	21	OD I/O	Node Board Select (5 V = Node is present, drive to 0 to turn node on)
SwTx(+/-)	2	I/O	10/100 Ethernet To Switch
SwRx(+/-)	2	I/O	10/100 Ethernet From Switch
FANTK[0..15]	16	I	Fan Tach Inputs
FANPWM[0..3]	4	O	Fan Speed Control (3.3 V = ON)
FANP#[0..3]	4	I	Fan tray present (3.3 V = fan tray is missing)
PDEG#[0..7]	8	I	Power supply degrade
PFAIL#[0..7]	8	I	Power supply fail
PINH#[0..7]	8	O	Power supply inhibit
STx	1	O	Serial transmit
SRx	1	I	Serial receive
SCTS	1	I	Serial clear to send
SRTS	1	O	Serial request to send
SDSR	1	I	Serial data set ready

Table 9. Pin Descriptions (Sheet 2 of 2)

Name	Count	Type	Description
SDTR	1	O	Serial data terminal ready
SRI	1	I	Serial ring indicator
SCD	1	I	Serial carrier detect
RpTx(+-)	2	I/O	10/100 Ethernet To rear panel
RpRx(+-)	2	I/O	10/100 Ethernet From rear panel
NEG_O	1	O	Negotiate output to other CMM
NEG_I	1	I	Negotiate input from other CMM
HLY_O#	1	O	Healthy output to other CMM
HLY_I#	1	I	Healthy input from other CMM
PRES_I#	1	I	Other CMM is present (0V)
PRES_O#	1	O	Grounded on CMM.
GA0	1	I	Location Address
CMM_SEL#	1	I	Tells CMM it has been seated. (Backplane grounds this pin)
GND	14	I	CMM GND
IPMB_PWR	8	I	CMM Power
BP_12V, BP_N12V, BP_5V, and BP_3.3V	4	I	Monitor backplane power 12,-12,5,3.3
RPMAC#	1	I	Rear panel major alarm clear
RPMIC#	1	I	Rear panel minor alarm clear
RPCR#	1	O	Rear panel critical alarm relay
RPMAR#	1	O	Rear panel major alarm relay
RPMIR#	1	O	Rear panel minor alarm relay
VIO	1	I	Backplane IO voltage (may not be present in non-PCI system)
RES	18	NC	No Connect on Backplane
TOTAL	220		

2.8.5 JA1 Ethernet Port

JA1 is an RJ-45 Ethernet port providing 10 Mbit (10BASE-T) and 100 Mbit (100BASE-TX) protocols. JA1 connects to the CMM's LAN0 Ethernet channel. A second Ethernet channel, LAN1, is only available at the backplane.

Two LEDs are located in the RJ-45 Ethernet connector:

- Yellow indicates speed
- Green indicates a link/activity

See Table 10 for JA1 Ethernet port pin definitions.

Table 10. JA1 Ethernet Port Pinout

Pin #	Description
1	TX+
2	TX-
3	RX+
4,5	Unused pair; terminated on ZT 7102
6	RX-
7	Unused pair; terminated on ZT 7102

2.8.6 J6 Serial Port

J6 is an RJ-45 connector providing a front-panel RS-232 serial port interface. Serial port signals are also directed out connector J1 to the backplane. See Table 11 for J6 serial port pin definitions.

Table 11. J6 Serial Port Pinout

Pin#	Function	Description
1	SRTS	Serial Request To Send
2	SDTR	Serial Data Terminal Ready
3	STx	Serial Transmit
4	GND	Ground
5	GND	Ground
6	SRx	Serial Receive
7	SDSR	Serial Data Set Ready
8	SCTS	Serial Clear to Send
-	SRI	Serial Ring Indicator (not utilized)
-	SCD	Serial Carrier Detect (not utilized)

2.8.7 J7 Telco Alarm Connector

J7 is a µDB-15 connector providing a front-panel telco alarm interface. See Table 12 for J7 Telco alarm connector pin definitions. Contact Intel for information about obtaining a compatible µDB-15 to DB-15 cable. Contact information is located in Appendix G, “Customer Support”.

For additional information on the Telco Alarm Connector, refer to the *Wiring Telco Alarm Connectors Application Note* posted at the following location:

http://www.intel.com/design/network/applnots/273926.htm

Table 12. J7 Telco Alarm Connector Pinout

Pin #	Function	Description
1	AMIR+	MinorReset +
2	AMIR-	MinorReset -
3	AMAR+	MajorReset +
4	AMAR-	MajorReset -
5	ACNO	CriticalAlarm - NO
6	ACNC	CriticalAlarm - NC
7	ACCOM	CriticalAlarm - COM
8	AMINO	MinorAlarm - NO
9	AMINC	MinorAlarm - NC
10	AMINCOM	MinorAlarm - COM
11	AMANO	MajorAlarm - NO
12	AMANC	MajorAlarm - NC
13	AMACOM	MajorAlarm - COM
14	APRCO	PwrAlarm - NO
15	APRCOM	PwrAlarm - COM
-	GND	Not Utilized
COM=Common NO=Normally Open NC=Normally Closed		

2.9 Electrical and Environmental

The ZT 7102 requires +5 VDC ±5% @ 1.8 A typical. CMM power comes from the backplane's IPMI_PWR rail; other voltages are derived as needed.

Caution: The processor core temperature must **never** exceed 100° C under any condition of ambient temperature or usage. This may result in permanent damage to the processor.

2.9.1 CMM Power

CMM power comes from the backplane's 5 V power (IPMI_PWR). See Table 13 for available voltages.

Table 13. CMM Power Availability

Voltage	Maximum Current	Where Used
5	100 mA	Miscellaneous components that cannot use 3.3 V.
3.3	4 A	80312 and most logic
2.5	700 mA	FPGA core
1.5	500 mA	80200 core

2.9.2 Power Ramp Circuitry

The ZT 7102 features a power controller with power ramp circuitry to allow the board's voltages to be ramped in a controlled fashion. The power ramp circuitry eliminates large voltage or current spikes caused by hot swapping boards. Controlled voltage ramping is a requirement of the *CompactPCI* Hot Swap Specification, PICMG* 2.1, Version 2.0.*

Fault current sensing is also provided. When a board fault (short circuit) or overcurrent condition is detected, the hot swap controller automatically removes power from the CMM components, and the Status LED on the faceplate illuminates red (see Table 3, "Face Plate" on page 20). Fault protection activates when the current exceeds 10 A for longer than 10 ms.

2.9.3 Operating Temperature

The ZT 7102 processor can operate between +5° C and approximately +65° C ambient. It is the users' responsibility to ensure that the ZT 7102 is installed in a chassis capable of supplying adequate airflow. The maximum power dissipation of the processor is 15 W (10 W typical). External airflow **must** be provided at all times.

Caution: The processor core temperature must **never** exceed 100° C under any condition of ambient temperature or usage. This may result in permanent damage to the processor.

2.9.4 Reliability

MTBF: 479,000 hours at 40° Celsius

MTTR: Three minutes (based on hot-swap board replacement), plus system startup

2.9.5 Absolute Maximum Ratings

See Table 14 for absolute maximum ratings.

Note: These are stress ratings only. Do not operate the ZT 7102 at these maximums. See Section 2.9.6, "DC Operating Characteristics" on page 31 for operating conditions.

Table 14. Absolute Maximum Ratings

Signal or Characteristic	Range
Supply Voltage, V_{CC}	5 V ±5% with 50 mV maximum ripple
Storage Temperature	-40° to +85° Celsius
Operating Temperature	+5° to +65° Celsius
Non-Condensing Relative Humidity	<95% at 40° Celsius

2.9.6 DC Operating Characteristics

Table 15. DC Operating Characteristics

Signal	Range
Supply Voltage, V_{CC}	4.85 V minimum to 5.25 V maximum
Supply Current, I_{CC}	1.8 A average (with 733 MHz processor and 32 Mbyte of SDRAM. Peak (short duration) power supply current may be significantly higher (up to 50%) and varies depending upon the application.

2.10 Onboard Switches

The ZT 7102 contains a push-button switch on the faceplate and one bank of DIP switches on the component side of the board. See Table 16 for switch identification and functions. Factory default switch settings are shown in Figure 7.

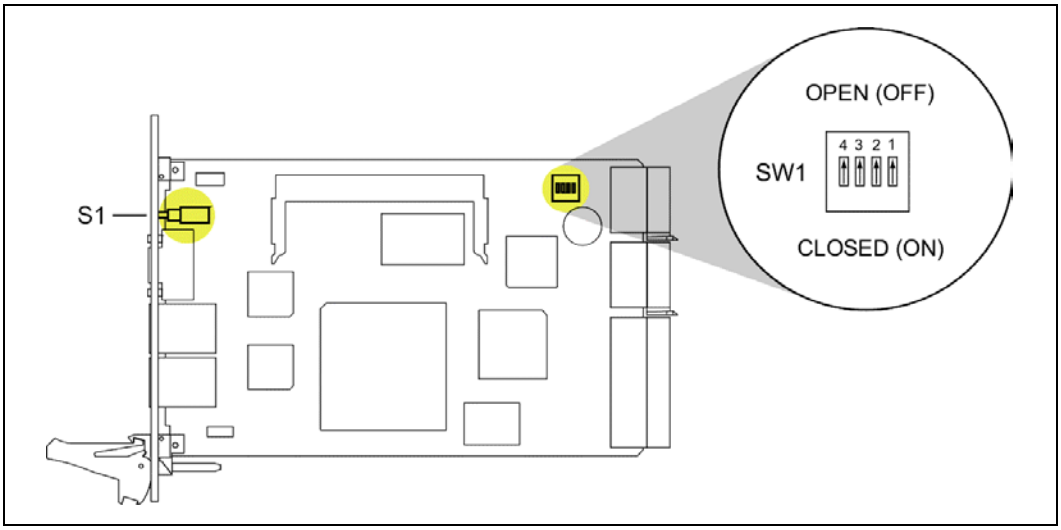
Note: Where switches are referenced in this chapter, "SWx" refers to the switch number and "-N" refers to the switch segment (SW1-2 indicates "switch number 1, segment 2").

Table 16. Switch Cross Reference Table

Switch	Function
S1, "S1 (Alarm Cutoff and Board Reset)" on page 32	Alarm Cutoff and Board Reset
SW1-1, "SW1-1 (Password Reset)" on page 32	Reset Password
SW1-2	Reserved
SW1-3	Reserved
SW1-4	Reserved

See Figure 7 for the default switch configuration.

Figure 7. Default Switch Configuration



2.10.1 Switch Descriptions

The following topics provide a detailed description of each switch.

2.10.1.1 S1 (Alarm Cutoff and Board Reset)

S1 is a push button at the faceplate of the ZT 7102. S1 toggles the CMM's alarm cutoff (ACO) state. When ACO is activated, the active alarm LEDs blink and the alarm relays are deactivated. This button does not clear alarms.

The Alarm Cutoff feature automatically cancels itself under the following conditions:

- When the ACO is engaged for more than 10 minutes without an active alarm present. This prevents the ACO from being accidentally left engaged after an alarm is cleared).
- When the ACO is engaged for more than x minutes, even if an active alarm is present (x can be set in the Command Line Interface with the Data item keyword AlarmTimeOut, see Table 23, “dataitem (-d) Keywords for All Locations” on page 58).

In addition to its ACO function, S1 is used to reset the CMM by pressing and holding the button for about five seconds. Reset is discussed in more detail in Section 4.0, “CMM Redundancy, Synchronization and Failover” on page 39.

2.10.1.2 SW1-1 (Password Reset)

Closing this switch resets the CMM's root logon password. Use this feature if you forget your password or want to restore the default root password for other reasons. For more information about resetting to the default root password, refer to Section 9.0, “Resetting the Password” on page 63.

Table 17. SW1-1 Functions

SW1-1	Function
Open [default]	Does not reset the CMM's root password at reboot.
Closed	Resets the CMM's root password at reboot.

2.11 Processor and Chipset

The Intel® IOP310 chipset contains two devices, the Intel® 80200 processor and the Intel® 80312 I/O companion chip. The 733 MHz 80200 processor features 32 Kbit data and instruction caches. The 80312 companion chip provides a memory controller and a PCI interface for the CMM's Ethernet controllers. Appendix E, “Datasheet Reference” contains a link to the datasheet for the chipset.

The ZT 7102 uses the Intel® IOP310 I/O processor chipset. Based on Intel XScale® microarchitecture, this chipset combines high performance processing with ultra-low power consumption.

2.12 Memory

The ZT 7102 includes 128 Mbytes of SDRAM on a Small Outline Dual Inline Memory Module (SODIMM). The SODIMM used is 16 Mbits x 72 PC133 unbuffered SDRAM with ECC on a 144-pin module.

2.12.1 Removing the SODIMM

The following instructions cover the mechanical aspects of removing the SODIMM from a ZT 7102.

1. Take the necessary precautions to protect the ZT 7102 and the SODIMM from static discharge.
2. Locate the SODIMM socket. Refer to Table 4, “Connector Assignments” on page 22 for the socket's location.
3. Press outward on the clips at either end of the SODIMM socket (press both clips at the same time) until the module ejects from the slot and tilts upward.
4. Pull the SODIMM away from the socket.

2.12.2 Installing Memory

The following instructions cover the mechanical aspects of installing the SODIMM in a ZT 7102.

1. Take the necessary precautions to protect the ZT 7102 and the SODIMM from static discharge.
2. Hold the SODIMM at an angle and push it into the SODIMM socket. Refer to Table 4, “Connector Assignments” on page 22 for the socket's location.
3. Align the notches on the ends of the module with the matching bumps on the latches and gently push the module down until it clicks in place.

2.13 Flash

The ZT 7102 uses an on-board 64 Mbyte flash memory device.

2.14 Dual Ethernet Controllers

The ZT 7102 uses two Intel® 82559 Fast Ethernet Controllers. The 82559 consists of both the Media Access Controller (MAC) and the physical layer (PHY) interface combined into a single component solution.

The CMM provides an RJ-45 connector on the faceplate to connect to out-of-band 10/100 LAN0. This connector is labeled with the Ethernet icon (refer to “Face Plate” on page 20). Care should be taken not to confuse the RJ-45 Ethernet port with the RJ-45 serial port that is next to it. (Note that the Ethernet port has integrated speed and activity LEDs).

2.15 Serial I/O

The ZT 7102 provides an RS-232 compatible RJ-45 serial port connector on the faceplate. Serial port signals are also available for rear panel access.

2.16 Telco Alarm Signal

2.16.1 Alarm Relays

The ZT 7102's alarm relay circuits are capable of carrying 60 VDC or 1 A, with a maximum rating of 30 VAC.

2.16.2 Opto Inputs

The ZT 7102 accepts timed pulse inputs for clearing Minor and Major alarm states (there is no reset for the Critical state). Reset is accomplished by asserting a voltage differential from 3.3 V to 48 V for between 200 and 300 ms. The acceptable voltage range is from 0 to 48 VDC continuous (handles up to 60 VDC at a 50 percent duty cycle). The current drawn by a reset input should not exceed 12 mA.

Caution: Do not apply more than 60 V (maximum) to any pin or combination of pins on the CMM's J7 Telco Alarm connector.

2.17 Real Time Clock

A Dallas Semiconductor* DS1307 Serial Real-Time Clock allows the ZT 7102 to keep track of time for accurate event logging. The DS1307 features 56 bytes of nonvolatile SRAM, a 100-year calendar, and battery backup.

The Real-Time Clock keeps time in UTC only. If the time is set for a different time zone, as in the example below, the difference will be automatically calculated giving the UTC equivalent. In this example, the time will be set 8 hours ahead of the local time.

To set the real-time clock, enter the correct date and time in the following format:

```
setdate 'Thu Apr 01 09:04:29 PST 2004'
```

The above command would set the time and date to the following:

```
Thu Apr 1 17:04:29 UTC 2004
```

Note: The real-time clock should be set to current date and time out of the box.

2.18 Network Time Synchronization

The CMM supports the RFC 868 Time Protocol. This allows the CMM to be configured to retrieve the date and time from a server over a TCP/IP network using the rdate command. To properly configure time synchronization using the RFC 868 Time Protocol, follow the steps below.

Enable RFC 868 Time Protocol on a Linux/Unix Host (server):

- In general this depends upon the OS and version. Please consult your OS documentation.
- The easiest way to enable this service is to use the OS GUI to enable the "time" server. This is part of inetd/xinetd.

Enable RFC 868 Time Protocol on CMM (client):

1. Create file "/etc/rdate.cfg".
2. Edit file "/etc/rdate.cfg" with a valid host name or IP address of an enabled RFC 868 Time Protocol server.
3. Reboot the CMM, and check the new date (use "date") to make sure it is identical to the RFC 868 Time Protocol server (use "busybox rdate -p <host name or IP>".
4. This file will only be preserved during upgrades to same or newer versions.

Note: Since rdate is executed during CMM boot, it is possible that on a chassis powerup the CMM may not be able to sync up with a server if the network path to it is through the backplane Ethernet port. For proper network time synchronization during CMM boot, the CMM should be able to contact the time server through the front Ethernet port.

2.19 Watchdog Timer

The ZT 7102 uses a Maxim* MAX6374KA-T watchdog timer to supervise processor activity. When the software fails to strobe the watchdog within a set period (once every second), the watchdog alerts the CMM's Complex Programmable Logic Device (CPLD), which resets the CPU.

2.20 Battery Backup

The CMM contains a CR 1025 onboard battery to retain certain functions such as the real time clock, even when no standby power to the board is present. If it becomes apparent that the time and date no longer stay current, then the battery should be replaced. The battery should only be replaced with a lithium CR 1025 or equivalent. After the battery is replaced, use the *setdate* command to set the time correctly. See Section 2.17, “Real Time Clock” on page 34 for information on setting the time and date.

Note: Date and time are critical to the operation of the CMM because health events are stamped with this information. Always ensure the date and time of the CMM are kept current.

2.20.1 Replacing the Battery

1. Slide the battery out from the battery retention clip.
2. Slide the new battery into the clip with the positive (+) side up. The metal arm of the retention clip should rest on top of the positive side of the battery.

Table 18. Battery Characteristics

Characteristic	Description
Battery Voltage:	3 V
Battery Capacity:	30 mAh
Real-Time Clock Requirements:	300 uA typical, 500 uA maximum (Vbat = 3 V, Vcc = 0 V)
Real-Time Clock Data Retention:	11.4 years typical/6.8 years minimum (not powered). When powered, RTC takes its power from +5v, not battery.
Electrochemical Construction:	Long-life lithium with solid-state polycarbon monofluoride cathode.

Caution: The ZT 7102 contains a lithium battery. Do not disassemble or recharge the battery. Do not dispose of the battery in fire. When the battery is replaced, the same type or an equivalent type recommended by the manufacturer must be used. Used batteries must be disposed of according to the manufacturer's instructions.

2.21 Running Your System Without A CMM

In CMM-based systems, the CMM controls power to the general-purpose slots in accordance with the PICMG* 2.1 specification. PICMG* 2.1 defines the power control signals (BD_SEL#) as pulled up on general-purpose boards. When the BD_SEL# signals are not asserted (grounded) by external means, the general-purpose boards will not power up. Therefore, to operate a CMM-based system without a CMM board, BD_SEL# must be grounded at each slot on the backplane.

Your system may have jumpers at each slot on the backplane to ground BD_SEL#. Inserting the BD_SEL# jumper will allow the node slot to power up regardless of the CMM. Node slots with the BD_SEL# jumper installed cannot be managed using the CMM. Refer to system-specific documentation for more information about running the system with or without a CMM board.

3.0 Software Specifications

3.1 Operating System

The CMM runs a customized version of embedded BlueCat* Linux* 4.0 on an Intel® 80200 processor with Intel XScale® technology. Development support for BlueCat Linux* is available on the web at http://www.lynuxworks.com.

3.2 Command Line Interface (CLI)

The Command Line Interface (CLI) connects to and communicates with the intelligent management devices of the chassis, boards and the CMM itself. The CLI is an IPMI-based library of commands that can be accessed directly or through a higher-level management application. Administrators can access the CLI through Telnet or the CMM's serial port. Using the CLI, users can access information about the current state of the system including current sensor values, threshold settings, recent events, and overall chassis health.

3.3 SNMP/UDP

The chassis management module supports both queries and traps on SNMP v1 or v3. The SNMP version can be configured through the CLI interface. The default is for SNMP v1. A MIB for the entire platform is included with the CMM.

Along with SNMP traps, the CMM sends UDP alerts to port 10000. The content of these UDP alerts is the same as the SNMP traps.

3.4 Remote Procedure Call (RPC) Interface

In addition to the console command-line interface, the CMM can be administered by custom remote applications via remote procedure calls (RPC).

3.5 Ethernet Interfaces

Eth1 is the Ethernet port going to the backplane on the CMM. Eth1 should remain static at all times. It is through this port that synchronization will be attempted initially. For synchronization to occur on eth1, both Ethernet switches must be present in the chassis.

Eth1:1 is a an IP address used to connect to the active CMM at any time through the chassis backplane and therefore should remain static. Eth1:1 is a static alias of eth1 defined in the /etc/ifcfg-eth1 file as STATICIP2. Because this IP address is synced between the two CMMs, it should only be changed on the active CMM. See Section 7.3.1, “Setting IP Address Properties” on page 54 on how to set the eth1:1 IP address.

Eth0 is the Ethernet interface on the front of the CMM that can connect to an external or private management network through a switch or through a crossover to the other CMM. This is used if no Ethernet switches are available in the chassis. Eth0 can be set to static or DHCP.

In a platform containing redundant CMMs, upon initial power up, both CMMs will have identical IP addresses on all of their Ethernet interfaces. The standby CMM will automatically decrement its IP address on Eth1 by 1 so that synchronization can occur; however, the user must change these IP addresses upon initial login so that an address conflict does NOT occur and synchronization and failover can operate correctly. See Section 7.3.1, “Setting IP Address Properties” on page 54 for further information and instructions on changing the IP addresses. See Section 4.0, “CMM Redundancy, Synchronization and Failover” on page 39 for further information regarding synchronization, failover, and redundancy.

3.6 Sensor Event Logs (SEL)

The chassis management module utilizes a dual domain for hosting the sensor event logs (SEL) of the chassis components and boards used in the platform. With the dual domain architecture, some SEL files are stored locally on the CMM in /etc/cmm, and some SEL files are stored on the individual blades.

For the CMM, backplane, and chassis components, the CMM stores the SELs locally on the CMM in /etc/cmm.

3.6.1 CMM SEL Architecture

The SEL files stored locally on the CMM in /etc/cmm have a maximum size of 8 Kbytes. If a SEL becomes full, the CMM will wrap the SEL to the beginning, and all previous SEL entries will be cleared.

3.6.2 Satellite Management Controller (SMC) Boards

For a board inserted into the platform acting as a satellite management controller (SMC), the CMM will also store the SEL file locally. Some SMC-based boards may also contain their own SEL. Health events generated from SMC-based boards will be stored in the SEL on the CMM as well as on the SMC board if a SEL exists. When the command is issued to get the SEL from an SMC-based board, the CMM will always retrieve the SEL file stored locally on the CMM in /etc/cmm.

3.6.3 Baseboard Management Controller (BMC) Boards

For boards inserted into the system acting as a baseboard management controller (BMC), the SEL file will be stored on that particular board. All health events will be logged in the SEL on that board. When the command is issued to get the SEL from the board, the CMM will retrieve the entire SEL via IPMB from that board.

3.6.4 Retrieving a SEL

The CLI can be used to retrieve the SEL for a location. To retrieve the SEL for a particular location, issue the following get command:

```
cmmget -l [location] -d SEL
```

Where location is the component you wish to retrieve the SEL for (cmm, chassis, bladeN).

4.0 CMM Redundancy, Synchronization and Failover

4.1 Overview

The CMM supports redundant operation with automatic failover in chassis using redundant CMM slots. In systems where two CMMs are present, one acts as the active chassis manager and the other as standby. Both CMMs monitor each other, and either can trigger a failover if necessary.

Data is always synchronized from the active CMM to the standby CMM whenever any changes occur. Existing data on the standby CMM is overwritten. A full synchronization between the active and standby CMM occurs on initial power up or insertion of a new CMM. See Table 19, “CMM Synchronization” on page 40 for a list of data items that are synchronized between the active and standby CMM.

4.2 Hardware Redundancy Specification

The following hardware interfaces exist for the support of redundancy and automatic failover:

- Cross-connected CMM present inputs (PRES_I#) and outputs (PRES_O#)
- Cross-connected CMM health inputs (HLY_I#) and outputs (HLY_O#)
- Cross-connected negotiation inputs (NEG_I) and outputs (NEG_O)

The active CMM monitors its PRES_I# and HLY_I# inputs to determine if it has a healthy standby CMM. The active CMM deasserts its HLY_O# output to trigger a failover to the standby CMM.

The cross-connected negotiation signals are used to assure that only one CMM is active at a time. At any time, the standby CMM can trigger a failover by driving its NEG_O output low.

4.3 Synchronization

To ensure data on the standby CMM matches the data on the active CMM, the active CMM synchronizes its data with the standby CMM, overwriting any existing data on the standby CMM.

The CMMs will initially fully synchronize data from the active to the standby CMM approximately 30 seconds after the CMMs boot (which can take up to 2 minutes total for both CMM boot and synchronization). An insertion of a new CMM will also cause a full synchronization from the active to the newly inserted standby, which also takes 2 minutes. Partial synchronization will also occur any time files are modified or touched via the Linux* “touch” command. Date and time are synched every hour.

Note: During synchronization, the LEDs on the standby CMM may blink on and off as the healthevents that were logged in the SEL are synchronized.

The following items are synchronized between CMMs. During a full synchronization, all of these files and data are synchronized. A change to any of these files results in that file being synched. The active CMM overwrites these files on the standby CMM.

Table 19. CMM Synchronization

File(s) or data	Description	Path:
/etc/cmm.cfg	CMM's main configuration file	Ethernet
/etc/passwd	Password file	Ethernet
/etc/shadow	Password file	Ethernet
/etc/cmm/sel_*	All SEL files except the CMM SEL files	Ethernet
/etc/cmm/*.bin	All SDR Files	Ethernet
/etc/cmm/*.sif	All SIF Files	Ethernet
/etc/cmm/DroneList.cfg	Drone mode compatible forms	Ethernet
/etc/cmm/PWMMap.cfg	PWM Fanspeed Mapping	Ethernet
/etc/cmm/SlotLayout.cfg	Chassis slot layout configuration	Ethernet
/etc/var/snmpd.conf	SNMP configuration files	Ethernet
/etc/snmpd.conf	SNMP configuration files	Ethernet
/home/scripts	Entire user scripts area	Ethernet
date and time	Date and time	IPMB
IP Address Settings	CMM eth1, eth1:1, and eth0 IP address settings	IPMB
Blade power states	Status of blade power states	IPMB
Power Supply Inhibit States	Status of power supplies that are inhibited by the CMM	IPMB

Warning: The /.rhosts file is used for synchronization and should NEVER be modified.

4.3.1 Synchronization Requirements

For synchronization to occur:

- The CMMs must be able to communicate with each other over their dedicated IPMB. The CMMs use a heartbeat via their dedicated IPMB to determine if they can communicate with each other over IPMB.
- An Ethernet connection must exist between the two CMMs. The CMMs must be able to ping each other via Ethernet for synchronization to be successful. This can be a connection through the Ethernet switches in the chassis, which requires both switches to be present in the chassis. A connection can occur through an external Ethernet switch connected to the front ports of the CMM pair, or alternatively, the connection can be a crossover cable connecting the two front ports of the CMM pair. If synchronization fails on Eth1, it will be attempted on Eth0. If the CMMs cannot successfully ping each other via Eth0 or Eth1, synchronization cannot occur.

A failure of synchronization to occur will result in a health event being logged in the chassis SEL, however this will not inhibit a failover from occurring.

4.4 CMM Failover

Once information is synchronized between the redundant CMMs, the active CMM will constantly monitor its own health as well as the health of the standby CMM. In the event of one of the scenarios listed in the sections that follow, the active CMM will automatically fail over to the standby CMM so that no management functionality is lost at any time.

4.4.1 Scenarios That Force a Failover

The following scenarios cause a failover as long as the standby CMM is operational:

- The active CMM is pulled out of the chassis.
- The active CMM's HEALTHY# signal is deasserted.
- A reboot command is issued to the active CMM.
- The front panel reset button on the active CMM is pushed in for more than five seconds.
- The internal synchronization code version of the active CMM is less than that of the standby CMM.

4.4.2 Scenarios That Prevent Failover

The following are reasons a failover cannot occur:

- The active CMM cannot communicate with the standby CMM via their IPMB bus.
- Not all data has been completely synchronized between the CMMs.

To determine the active CMM, use the CLI command:

```
cmmget –d redundancy
```

Eth1:1 will always point to the active CMM, which will allow you to access the active CMM through the chassis at all times.

4.4.3 Scenarios That Failover to a Healthier Standby CMM

The scenarios listed below can only cause a failover if the standby CMM is in a healthier state than the active CMM. The health of the CMM is determined by computing a CMM health score, which is equal to the sum of the weights of the following active conditions. A CMM health score is determined for each CMM whenever any of these conditions occur on the active CMM. The CMM health score is composed of the sum of the weights of any of the three conditions listed below. Each condition has a default weight of 1 assigned to it, causing all conditions to have equal importance in causing failover.

To determine if a failover is necessary when one of these conditions occurs, the active CMM computes its CMM health score, and requests the health score of the standby CMM. If the score of the standby CMM is LESS than the score of the active CMM, a failover will occur. If a failover does not occur, the Chassis SEL will contain an entry indicating the reason failover did not occur.

1. **SNMPTrapAddress1 ping failure:**

   A condition is asserted and a health score is computed if the active CMM cannot ping its first SNMP trap address as defined in the SNMPTrapAddress1 setting. Only a ping failure of the first SNMP trap address (SNMPTrapAddress1) can cause the condition to assert. SNMPtrapaddress2 through SNMPtrapaddress5 do not perform this ping test.

Note: The frequency of the ping to the first trap address can vary from one second to approximately 20 seconds.

2. **Unhealthy Ethernet Switch:**

   A condition is asserted and a health score is computed if the active CMM's corresponding Ethernet switch is not healthy or not present. The switch health is determined by the state of the HEALTHY# hardware signal coming from the Ethernet switch. Refer to the chassis specification to see which switch corresponds to the CMM. If both CMMs have unhealthy switches or are not present in the chassis, then a failover can still occur based on other failover conditions depending on the CMM health scores.

3. **Critical events on the active CMM:**

   A condition occurs if the active CMM has critical events for any of the CMM sensors (not chassis or blade sensors). Critical events are events associated with crossing an upper or lower nonrecoverable threshold of a sensor. If both CMMs have critical CMM events, then the number of major and minor CMM events is examined to decide if a failover should occur. The number of major events is compared, and if they are equal, the number of minor events is used.

4.4.4 Scenarios That Failover to an Equally Healthy CMM

The following conditions will cause a failover only if the health score of the standby CMM is equal to that of the active CMM:

1. The ejector latch on the active CMM is opened.
2. A manual failover is executed on the active CMM.

4.4.5 Failover Timing

Times required to detect different possible failover conditions and perform data synchronization vary. For example, detecting network connection loss can take up to approximately 20 seconds. Complete synchronization typically takes 7 to 30 seconds to occur, assuming both CMMs are fully booted and a healthy Ethernet network connection and IPMB connection exist between the two CMMs). Synchronization with a newly inserted CMM can take two minutes, since a newly inserted CMM needs that time to boot and initialize.

Once the CMM data is initially synchronized, failover happens instantaneously at the hardware level. However, the CMM software requires some time to initialize various components following a failover. Software-based remote management applications accessing the CMM **will** need to reconnect to the newly active CMM. The newly active CMM may respond with unexpected errors while initializing.

4.4.6 Manual Failover

The following command can be issued to the active CMM to cause a failover manually to the standby CMM:

```
cmmset -l cmm -d failover -v 1
```

A manual failover can only be initiated on the active CMM. A failover will only occur if the standby CMM is at least as healthy as the active CMM. Once the command executes, the former standby CMM immediately becomes the active CMM.

If the failover could not occur, the CLI will indicate the reason why the failover could not occur, and a SEL event will be recorded.

5.0 CMM Installation and Removal

This chapter describes the steps necessary to install and set up the ZT 7102 Chassis Management Module (CMM). It includes instructions on unpacking, installing, and removing the CMM. Some systems may come with the CMM(s) already installed.

Caution: The ZT 7102 plugs into a dedicated chassis management slot. System components will be damaged if a standard 3U board is plugged into the chassis management slot.

5.1 Unpacking

Check the shipping carton for damage. If the shipping carton and contents are damaged, notify Intel Customer Support. Retain the shipping carton and packing material for inspection by the carrier. Obtain authorization before returning any product to Intel. Refer to Appendix G, "Customer Support" for assistance information.

Caution: This board must be protected from static discharge and physical shock. Never remove any of the socketed parts except at a static-free workstation. Use the anti-static bag shipped with the product to handle the board. Wear a wrist strap grounded through one of the system's ESD Ground jacks when servicing system components.

5.2 Connectivity

The ZT 7102 is intended to be installed as the CMM in a system's Chassis Management Slot. Where two Chassis Management Slots are available in a system, two CMMs may be installed. (They will automatically negotiate for active and standby roles.)

The ZT 7102 is designed to operate in a backplane providing CompactPCI-style 2 mm hard metric connectors to the board's J1, J2, and J3 backplane interfaces. The backplane's CMM interfaces are specific to the CMM and do not support other 3U CompactPCI boards. See Section 2.8, "Connectors" on page 22 for connector descriptions and pinouts.

5.3 Installing the CMM

The following instructions cover the mechanical aspects of installing the ZT 7102 CMM in a compatible system. For a better understanding of what happens when a CMM is absent from a system, see "Removing the CMM" on page 44.

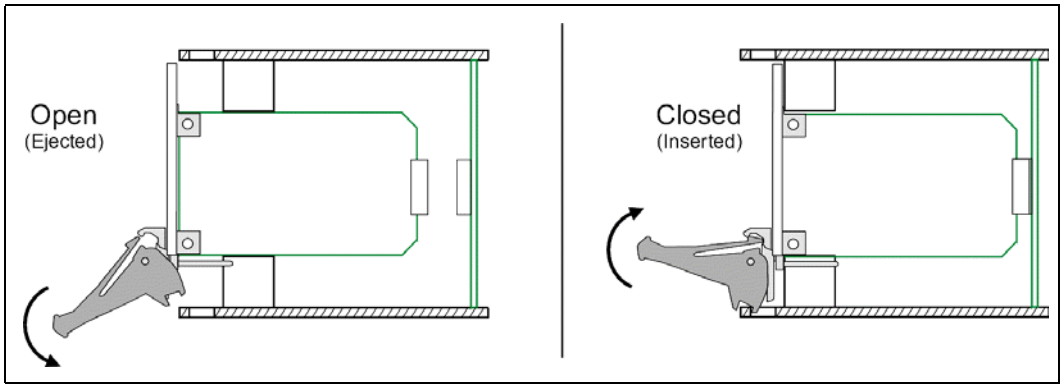
Warning: When the system is plugged in, high voltages are present on the backplane. Do not reach into the enclosure.

Warning: Static electricity can damage electronic components. Wear a wrist strap grounded through one of the system's ESD ground jacks when servicing system components.

1. Take the necessary precautions to protect the ZT 7102 from static discharge. System power does not need to be off to install a CMM board.
2. Locate the system's dedicated CMM slot and remove the filler panel or existing CMM (see Section 5.4 for removal instructions).

3. Prepare the new/replacement CMM by opening its ejector handle (refer to Figure 8).
4. Carefully align the edges of the board with the card guides in the CMM slot. It may be helpful to look into the enclosure to verify correct alignment of the rails in the guides.
5. Taking care to keep the board aligned in the guides, slide the board in until the injector/ejector mechanism engages the retention bar.
6. Simultaneously push in the board and rotate the ejector handle to its closed position (rotate inward) to seat the backplane connectors.
7. If system power is on, the CMM will boot and its Status LED will light green (active CMM) or blink green (standby CMM).
8. Screw in the board retention screw to anchor the board in the chassis. This screw is located at the opposite end of the faceplate from the ejector handle. Refer to Figure 3, “Face Plate” on page 20 for the screw's location.
9. Use the CMM's Command Line Interface (CLI) to configure the CMM.

Figure 8. Ejector Handle Operation



5.4 Removing the CMM

Note: The CMM should only be removed when the blue hot swap LED is lit.

In CMM-based systems, the CMM controls power to every slot in the system via BD_SEL#. If a system's only CMM is removed, all the boards in the system lose power. When hot swapping a CMM, it performs a controlled shutdown of itself but not the other boards in the system. Therefore, you should ensure that the entire system is in a “safe” state before removing the CMM. The CMM's hot swap LED will not light while other boards in the system are powered on.

If a redundant CMM is available, the CMM being removed automatically fails over to the standby CMM if it is active when the ejector latch is opened. If the active CMM is being removed, the CMM status light will begin to blink indicating that a failover has occurred. The other CMM’s status light will be solid green. If a failover does not occur, check the Chassis SEL to see why a failover cannot occur. In a redundant system, the CMM's removal does not cause a loss of system power or an interruption in service (you must still wait for the blue hot swap LED to light before removing the CMM).

If the CMM loses power or is removed suddenly, the CMM's flash memory could become corrupted. If this happens, restore the board to operation using the instructions in “Updating Software” on page 127.

Note: Removing the CMM before safely shutting it down is highly discouraged. Corruption of the flash could occur if the CMM is yanked suddenly during a flash operation.

To remove the ZT 7102 CMM from a system:

Note: Take the necessary precautions to protect the ZT 7102 from static discharge. System power does not need to be off to remove a CMM board. However, other boards in the system should be shut down.

1. Unscrew the board retention screw that fastens the board to the enclosure. This screw is located at the opposite end of the faceplate from the ejector handle. Refer to Figure 3, “Face Plate” on page 20 for the screw's location.
2. If system power is off or the CMM's blue hot swap LED is on, proceed to the next step. If not, the CMM needs to be in a “safe” state before it can be removed. Signal the CMM that it is about to be removed by partially unlatching its ejector. Do not fully open the ejector as this levers the board out of the enclosure and breaks its backplane connection before the board can shut down properly.
3. Wait for the blue hot swap LED to light. It will take a few seconds to light up. The LED will not light if other boards in the system are powered on. Refer to Figure 3, “Face Plate” on page 20 for the LED's location.
4. Open the ejector handle fully, rotating it outward until the board disengages from the backplane (refer to Figure 8 for ejector handle instructions).
5. Slide the board evenly out of the enclosure.
6. Install a replacement CMM or cover the empty slot with a filler panel to maintain the system’s shielding and cooling performance.

6.0 Built-In Self Test

The CMM provides for a Built-In Self Test (BIST). The test will be run automatically after power-up or front-panel reset. This test will detect flash corruption as well as other critical hardware failures.

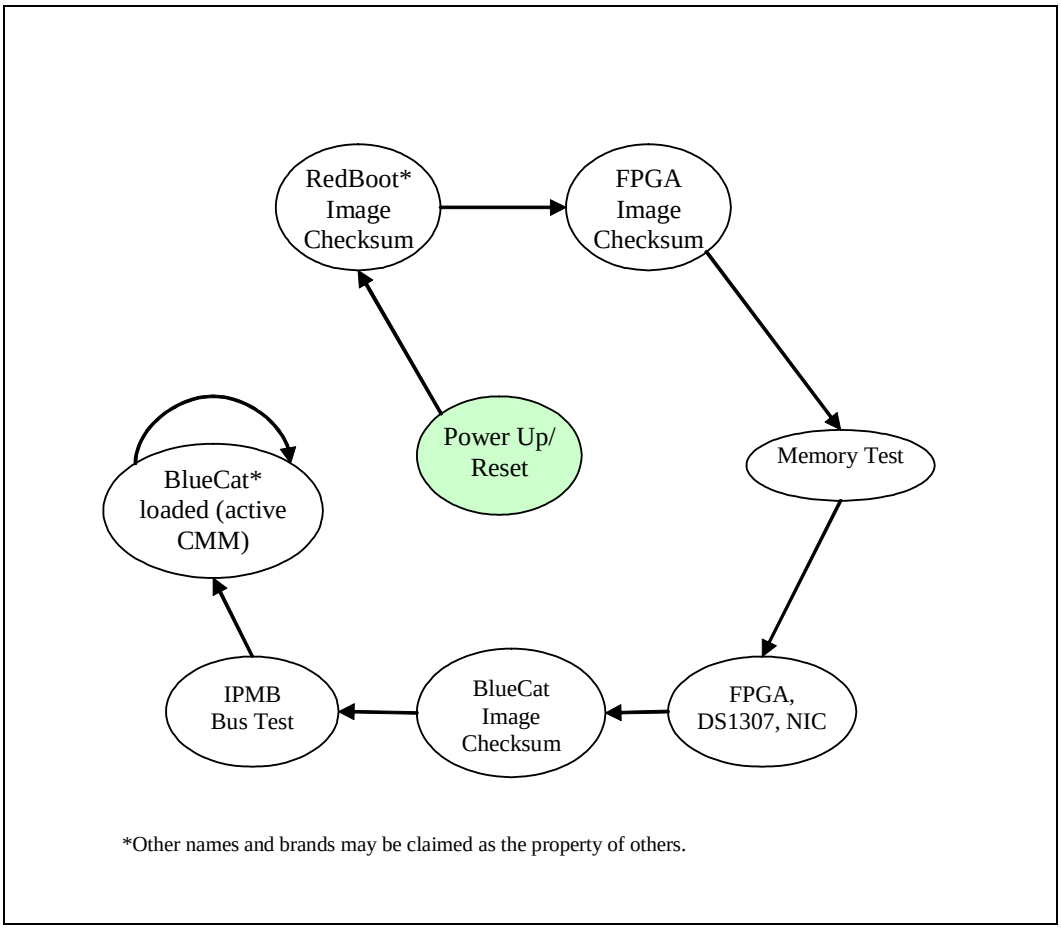
Results for the BIST are displayed on the console through the serial port during boot time. Results of BIST will also be available through the CLI if the OS successfully boots.

If the BIST detects a fatal error, the CMM will not be allowed to function as an active CMM.

6.1 BIST Test Flow

The following state diagram shows the order of the tests RedBoot* runs following a power-up or front-panel reset. On every state before reaching active CMM, if there is an error, RedBoot will log the error event into the EEPROM, route the error message to the serial port, and continue booting. If the execution hangs before the OS loads due to the nature of the error, the CMM hangs. If the OS successfully boots, it will alert users to any errors that occurred during boot.

Figure 9. BIST Flow Chart



The BIST is broken down into stages consisting of groups of tests run at serious times throughout the boot process. Table 20 shows the different BIST stages and the tests associated with each stage:

Table 20. BIST Implementation

Boot-BIST	Early-BIST	Mid-BIST	Late-BIST
RedBoot image checksum	Strobe watchdog timer to extend timeout period	Extended memory test	BlueCat image checksum
FPGA image checksum		FPGA version check	IPMB bus test
Base memory test		DS1307 RTC test	
		Local PCI bus / NIC presence test	

6.2 Boot-BIST

The codes in Boot-BIST will be executed at the very early stage of the RedBoot bootstrap, which is just before the FPGA programming and memory module initialization. Boot-BIST will perform checksum checking over the RedBoot image and the FPGA image. Checksum errors will be detected if there is a mismatch of between the calculated checksum and the stored checksum in FIS directory.

Boot-BIST will also perform a Base Memory Test for the first 1 Mbyte of memory. Whenever there is an error, BIST will inform the user by prompting a warning message through the console terminal and log the event to event-log area.

6.3 Early-BIST

The early BIST stage will extend the reset timeout period on the watchdog timer (MAX6374) by strobing GPIO7 on FPGA1. This prevents any possible hardware reset during the BIST process. The watchdog timer will be enabled after the ADM1026 GPIO initialization, and disabled once it reaches the RedBoot console. The OS will enable the watchdog timer again and start the strobing thread at the kernel level.

6.4 Mid-BIST

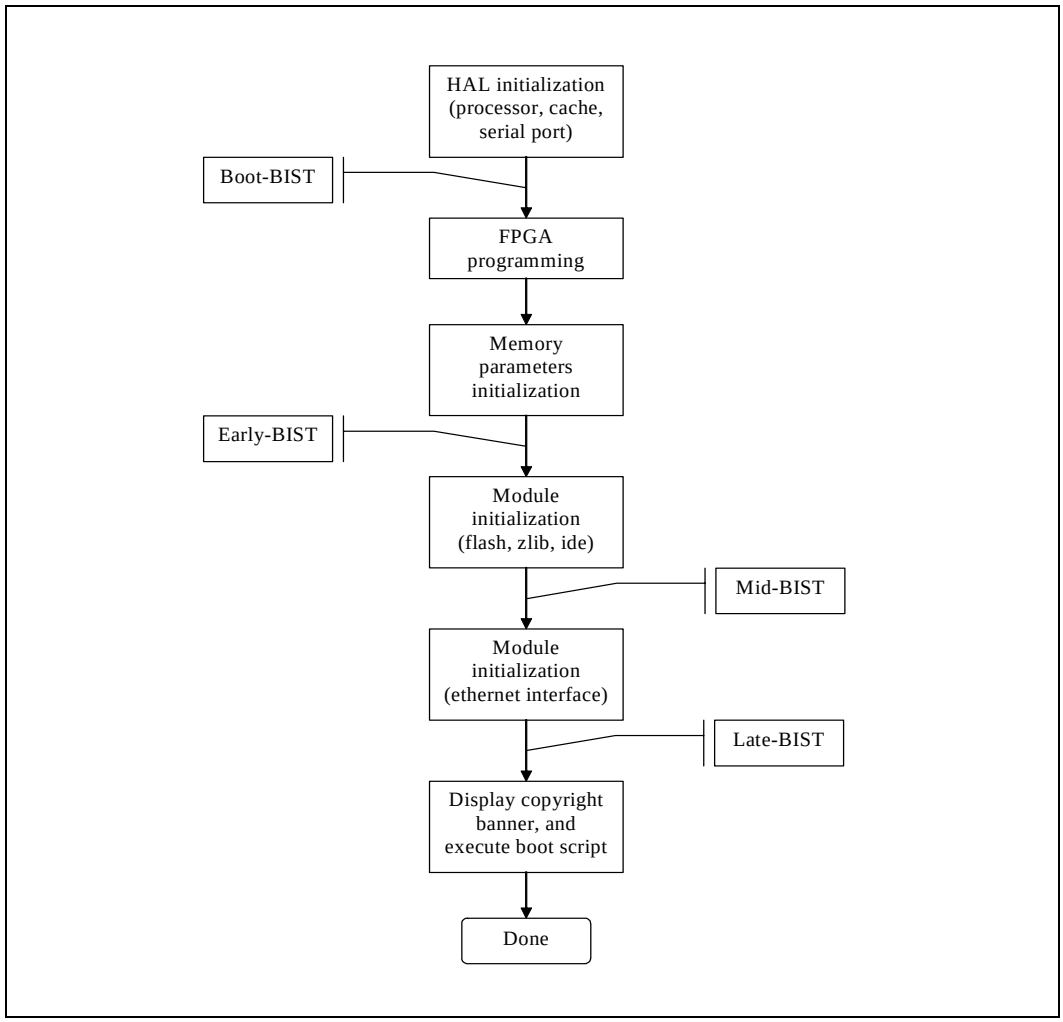
Mid-BIST will be started just after some basic modules initialization, such as flash and zlib compression. In this stage of BIST, the Extended Memory Test will be performed to scan and diagnose the possible bit errors in the memory. It starts scanning from 1 MByte to the maximum available memory size on the board. It will not test the memory below 1 MByte because portion of RedBoot has already loaded and resided on it. The memory test includes: the walking ones test, 32-bit address test, and 32-bit inverse address test. Furthermore, voltage and temperature ratings will be verified to lie within the hardware tolerable ranges. The FPGA firmware version will be checked and will alert if an older version of an FPGA image has been detected. Also, system date and time will be read out from the real time clock and displayed through the console terminal. NIC presence is also checked here. NIC self-test is not performed here because it is performed in the existing NIC driver module.

6.5 Late-BIST

Late-BIST involves disabling the watchdog timer to indicate a successful boot of RedBoot. It will also verify the checksum of the OS image stored in the FIS directory before proceeding with the boot script execution.

Figure 10 shows at which times during the boot cycle the various stages of BIST are performed.

Figure 10. Timing of BIST Stages



6.6 Event Log Area and Event Management

Errors detected by the BIST are stored in an event log. The event-log area is designed to have 269 entries. Each entry is 14 bytes. The event-log area is located in EEPROM on the CMM. The BIST can place entries into the event log until it becomes full. Once full, any new entries will be lost. The BIST event log is cleared by the OS once the OS logs any BIST errors into the SEL.

At OS start-up, the CMM will read the contents of BIST results in the reserved event log area and store the errors as entries in the CMM SEL. This will allow the CMM application to take the appropriate action based upon the SEL events as a result of RedBoot BIST tests. If there is not enough space to log the events in the CMM SEL, no results are logged to the CMM SEL.

The BIST event log will be erased only after the event log is stored into the CMM SEL. Event strings for BIST events are listed in Section 11.0, "Health Event Strings" on page 67.

6.7 OS Flash Corruption Detection and Recovery Design

The OS is responsible for flash content integrity at runtime. Flash monitoring under the OS environment can be divided into two parts: monitoring static images and monitoring dynamic images.

Static images refer to the RedBoot image, FPGA image and BlueCat image in flash. These images should not change throughout the lifetime of the CMM unless they are purposely updated or corrupted. The CRC for these files are written into flash when the images are uploaded.

Dynamic image refers to the OS Flash File System (JFFS2). This image will dynamically change throughout the runtime of the OS.

6.7.1 Monitoring the Static Images

A static test is run at specified time intervals during CMM operation. The interval is specified on the command line in the CMM startup script. The default interval is 24 hours. A value of zero will turn off the test. The static test will read each static image (RedBoot, FPGA, BlueCat), calculate the image checksum, and compare with the checksum in the RedBoot configuration area (FIS). If the check sum test fails, the error will be logged to the CMM SEL.

6.7.2 Monitoring the Dynamic Images

For monitoring the dynamic images, the CMM leverages the corruption detection ability from the JFFS(2) flash file system. At OS start-up, the CMM executes an initialization script to mount the JFFS(2) flash partitions (/etc and /home). If a flash corruption is detected, an event will be logged to the CMM SEL.

During normal OS operation, flash corruption during file access can also be detected by the JFFS(2) and/or the flash driver. If a flash corruption is detected, an event will be logged to the CMM SEL.

6.7.3 CMM Failover

If during normal OS operation a critical error occurs on the active CMM, such as for a flash corruption, the standby CMM will be checked to see if it is in a healthier state. If the standby CMM is in a healthier state, then a failover will occur.

6.8 BIST Test Descriptions

6.8.1 Flash Checksum Test

This test is targeted to verify the RedBoot image and FPGA image are not corrupted.This test will calculate the CRC32 checksum from the RedBoot image, and then compare with the image checksum stored in the FIS directory. If one mismatches another, BIST will switch to the backup image. If checksum mismatch was found from the FPGA image, BIST will load the backup image to program the FPGA device.

6.8.2 Base Memory Test

This test will write the data pattern of 55AA55AA into every 4-byte of memory below 1 Mbyte. Its objective is to verify wire connectivity of address and data pins between the memory module and the processor. The test will write the data pattern into the complete first 1 Mbyte, then verify the written data pattern by reading them out from the memory module. If the data pattern mismatches, the test will log the error event into event-log area and route error message to serial port.

6.8.3 Extended Memory Tests

6.8.3.1 Walking Ones Test

This test is targeted to verify the data bus wiring by testing the bus one bit at a time. The data bus passes the test if each data bit can be set to 0 and 1 independently of the other data bits.

6.8.3.2 32-Bit Address Test

This test is targeted to verify the address bus wiring. The smallest set of addresses that will cover all possible combinations is the set of “power-of-two” addresses. These addresses are analogous to the set of data values used in the walking 1's test. The corresponding memory locations are 0001h, 0002h, 0004h, 0008h, 0010h, 0020h, and so on. In addition, address 0000h must also be tested. The possibility of overlapping locations makes the address bus test harder to implement. After writing to one of the addresses, we must check that none of the others has been overwritten.

Note that not all of the address lines can be tested in this way. Part of the address-the left most bits-selects the memory chip itself. Another part-the right most bits-may not be significant if the data bus width is greater than eight bits. These extra bits will remain constant throughout the test and reduce the number of test addresses. For example, if the processor has 32 address bits, it can address up to 4 Gbytes of memory. If we want to test a 128 Kbyte block of memory, the 15 most-significant address bits will remain constant. In that case, only the 17 rightmost bits of the address bus can actually be tested. (128 K is 1/32,768th of the total 4 Gbyte address space.)

To confirm that no two memory locations overlap, first write some initial data value at each power-of-two offset within the device. Then a new value is written-an inverted copy of the initial value, to the first test offset. It is then verified that the initial data value is still stored at every other power-of-two offset. If a location is found, other than the one just written, that contains the new data value, you have found a problem with the current address bit. If no overlapping is found, the procedure is repeated for each of the remaining offsets.

6.8.3.3 32-Bit Inverse Address Test

This test is similar to the 32-bit address test except the addresses are tested in the inverse direction. The test helps identify a broader scope of possible addressing errors inherent in memory modules.

6.8.4 FPGA Version Check

This test is targeted to verify the correct FPGA image programmed into both FPGA chips. This test will display the FPGA version on both FPGAs. Both versions should be the same, or the user will be prompted with a warning message. If the programmed version is older than the expected, user will be prompted to upgrade to the latest FPGA image.

6.8.5 DS1307 RTC Test

This test is targeted to verify the functionality of DS1307 RTC chip. This test will display the date/time settings from the RTC and validate the readings. If any readings found to be non-BCD format, user will be prompted with warning message. This test will also capture current time, sleep a while, then compare the previously captured time and new time. If they differ, it means the RTC is working. Otherwise, user will be prompted with warning message.

6.8.6 NIC Presence/Local PCI Bus Test

This test generates the PCI bus transaction by scanning the PCI buses available on the board. This test will detect the two Ethernet devices and verify each device has the valid Vendor ID and Device ID in the PCI configuration space. NIC internal self-test will not be performed here, as the self-test will be executed when loading the Ethernet driver.

6.8.7 OS Image Checksum Test

This test is targeted to verify the OS image stored in the flash is not corrupted. This test will calculate the CRC32 checksum from the OS image, then compare with the image checksum stored in the FIS directory. If one mismatches the other, BIST will log an error event to the event-log area and route the error message to serial port.

6.8.8 CRC32 Checksum

CRC32 is the 32-bit version of Cyclic Redundant Check technique designed to ensure the bits' validity and integrity within the data.

It first generates the diffusion table, which consists of 256 entries of double-word; each entry is known as a unique diffusion code. The checksum calculation is started by fetching the first byte in data buffer, exclusive-OR with the temporary checksum value. The resulting value will be AND-ed with 0xFF to restrict an index from 0 to 255 (decimal). That index will be used to fetch a new diffusion code from the table. Next, the newly fetched diffusion code will be exclusive-OR with the most significant 24 bits of the temporary checksum value (effectively 8 bits left-shifting the checksum value). The resulting value is the new temporary checksum value. The calculation process is repeated until the last byte in the data buffer. The final temporary checksum value becomes the final checksum value.

6.8.9 IPMB Bus Busy/Not Ready Test

This test identifies any potential FPGA lockup before loading BlueCat. If the FPGA is detected to be locked up, an event indicating which bus actually failed will be logged into the Event log.

7.0 CMM Connection and Setup

7.1 CLI Overview

The Command Line Interface (CLI) connects to and communicates with the intelligent management devices of the chassis, boards, and the CMM itself. The CLI is an IPMI-based library of commands that can be accessed directly or through a higher-level management application. Administrators can access the CLI through Telnet or the CMM's serial port. Using the CLI, users can access information about the current state of the system including current sensor values, threshold settings, recent events, and overall chassis health.

Note: Commands should always be issued to the CLI on the active CMM. The standby CMM can only respond to commands to the CMM location.

7.2 Connecting to the CLI

The CMM provides two connections on its front panel (see Figure 3, “Face Plate” on page 20)

- An Ethernet connection via an RJ-45 connector
- An RS-232 serial port interface also via an RJ-45 connector

Either of these interfaces can be used to log into the CMM, as well as the Ethernet interface provided through the backplane of a chassis. Use Telnet to log into the CMM over a Ethernet connection, or use a terminal application or serial console over the RS-232 interface.

If logging in for the first time to set up IP addresses, use the serial port console interface to perform configuration.

7.2.1 Connecting through a Console

Connect an RS-232 serial cable with an RJ-45 connector to the serial console port on the front of the CMM. Set your terminal application settings as follows:

- Baud – 115200
- Data Bits – 8
- Parity – None
- Stop Bits – 1
- Flow Control – Xon/Xoff or none

Connect using your terminal application.

7.2.2 Telnet into the CMM

To telnet into the CMM, point your console or telnet application to the IP address of the CMM you wish to telnet to. If you wish to telnet to the active CMM through the backplane, you can point the telnet application to the Eth1:1 IP address.

7.2.3 FTP Into the CMM

Using an FTP client, FTP to the IP address of the CMM you wish to transfer files to or from.

7.3 Initial Setup - Logging in for the First Time

The default username for the cmm is ***root***. The default password is ***cmmrootpass***.

At the login prompt, enter the username: ***root***

When prompted for the password, enter: ***cmmrootpass***

The root password can be changed using the ***passwd*** command. For information on resetting the CMM password back to default, refer to Section 9.0, “Resetting the Password” on page 63.

7.3.1 Setting IP Address Properties

By default, the CMM assigns IP addresses statically. Eth0, the front panel Ethernet port, is configured with the static IP address 10.90.90.91. Eth1, the Ethernet port to the backplane, is configured with a static IP address of 192.168.100.92. Eth1:1, an alias of Eth1 is used to always point to and be active on the active CMM, is configured with a static IP address of 192.168.100.93.

On initial power up of a chassis with two CMMs, both CMMs will have the same IP addresses assigned by default. When the chassis is powered up, the standby CMM automatically decrements its eth1 IP address by one less than the active CMM.

Example:

1. A dual CMM Chassis is powered up.
2. Active CMM is assigned IP address of 192.168.100.92 to Eth1 on the active CMM.
3. Standby CMM is assigned IP address of 192.168.100.91 to Eth1 on the standby CMM.

At this point, the static IP addresses must be changed to appropriate values for their network configuration. Ensure that the two CMMs do not contain duplicate IP addresses on Eth0 and eth1 to avoid address conflicts on the network.

eth0 can also be set using DHCP. eth1, and eth1:1 will always remain static.

Note: eth0 should always be set to a different subnet than eth1. Failure to set eth0 to a different subnet than eth1 will cause network errors on the CMM and redundancy will be lost.

Note: A DHCP server must be present on the network for the CMM to get a valid IP address. The network reload command will refresh the IP addresses on both network interfaces.

7.3.1.1 Setting IP Information (IP Address, Netmask, and Gateway)

Eth0

1. Open the /etc/ifcfg-eth0 file using the vi editor
2. Ensure the BOOTPROTO variable is set to static.

Note: Linux is case sensitive, so ensure that the BOOTPROTO variable is entered in lower case letters in the step above.

3. Set the STATICIP variable to the IP address you want to assign to that interface.
4. Set the NETMASK variable to the appropriate netmask for your network.
5. Set the GATEWAY variable to the appropriate value for the gateway on your network
6. To activate the changes, at the user prompt, type:

```
/etc/rc.d/network reload
```

Eth1 and Eth1:1

Note: Eth1 and Eth1:1 are static IP addresses only. They do not support DHCP.

1. Open the /etc/ifcfg-eth1 file using the vi editor
2. Set the STATICIP1 variable to the IP address you want to assign to that interface.
3. Set the STATICIP2 variable to the IP address you want to assign to the active CMM on the network. This value should ONLY be set on the active CMM, as it will be synchronized to and overwritten on the standby CMM.
4. Set the NETMASK variable to the appropriate netmask for your network.
5. Set the GATEWAY variable to the appropriate value for the gateway on your network.
6. To activate the changes, at the user prompt, type:

```
/etc/rc.d/network reload
```

Note: The Eth1:1 address should only be changed on the active CMM. The new address will be synchronized to the standby CMM automatically when the /etc/rc.d/network reload command is executed.

7.3.1.2 Setting Eth0 to DHCP

1. Using the vi editor, change the BOOTPROTO variable in the /etc/ifcfg-eth0 file to **dhcp**.

Note: Linux is case sensitive, so ensure that the BOOTPROTO variable is entered in lower case letters in the step above.

2. To activate the changes, the user can reboot the CMM, or at the user prompt, type:

```
/etc/rc.d/network reload
```

Note: eth1 and eth1:1 should not be set to dhcp. The BOOTPROTO variable does not exist in these files.

7.3.2 Setting a Hostname

The hostname of the CMM is a logical name that is used to identify a particular CMM. This name is shown at login time and advertised to any DNS servers on a network.

To change the hostname:

1. Using the vi editor, change the HOSTNAME variable in /etc/HOSTNAME to the desired name.
2. To activate the changes, at the user prompt, type: `/etc/rc.d/network reload`

Note: Executing **network reload** also causes the network interfaces to reload their IP addresses. If DHCP is being used on a network interface, then it is possible that the IP address on that interface will change.

7.3.3 Setting Auto-Logout Time Limit

For security purposes, the CMM automatically logs the user out of the current console session after 15 minutes (900 seconds). This auto-logout time can be changed by editing /etc/profile and changing the TMOUT value to the desired setting. The timeout (TMOUT) value is set in seconds (900 seconds is the default). A setting of TMOUT=0 will disable the automatic logout. This can also be set at the command line, however doing so at the command line will not be persistent across a reboot of the CMM.

7.3.4 Rebooting the CMM

To reboot the CMM, type the ***reboot*** command in the CLI. Telnet sessions will have to be reestablished after the CMM is rebooted.

Do not use the “init 0” or “init 6” command to reboot the CMM as problems may result.

8.0 Command Line Interface (CLI) Syntax and Arguments

The command line interface on the CMM supports two types of commands: cmmgets and cmmsets. Cmmgets are used to query for information, whereas cmmsets are used to write information.

8.1 cmmget and cmmset syntax

The syntax for calling the CLI from the command line is as follows:

```
cmmget [-h] [-l location] [-t target] -d dataitem
cmmset [-h] [-l location] [-t target] -d dataitem -v value
```

Where **cmmget** and **cmmset** are the CLI executables. The parameters can be in any order. The CLI is case insensitive, except for the executable name. Parameters shown in brackets are optional.

Any attribute value that contains a space must be enclosed in quotes. This happens often when specifying targets. For example, to get the current value of a sensor called *Brd Temp* on the CMM, the command would be:

```
cmmget –l cmm –t “Brd Temp” –d current
```

8.2 Help Parameter: -h

If the Help parameter is given, then the rest of the parameters are ignored, and the help text is output to the user.

8.3 Location Parameter: -l

The Location parameter is the location in the system that the user is executing the **cmmget** or **cmmset** on. If no location is given then the default location is the CMM. The Location keywords are shown in the following table.

Table 21. Location (-l) Keywords

Keyword	Function
cmm	The Chassis Management Module.
bladeN	One of the CPU boards in the chassis. N refers to the chassis slot number into which the CPU board is inserted. Refer to the chassis documentation for slot information.
system	The entire platform.
chassis	Chassis specific items excluding the boards (such as fan trays and power supplies).
fantrayN	One of the fan trays in the system. N refers to the number of the fan tray in the system.
powersupplyN	(Only works with IPMI power supplies.) One of the power supplies in the system where N is a number between 1-8.

If no location parameter is given, then the default is CMM.

8.4 Target Parameter: -t

The Target parameter is the sensor or variable that the **cmmget** or **cmmset** acts on. The Target keywords are shown in the following table.

Note: If the target is not specified, then it is assumed that Dataitem is an attribute of Location.

Table 22. Target (-t) Keywords

Keyword	Description
fru	Used to query the FRU or individual FRU fields for a specific location. For more information on querying a component's FRU and individual FRU fields, see Section 16.0, "Obtaining FRU Information" on page 89.
SDR Sensors	The sensor name as defined in its Sensor Data Record (SDR) for that component. Sensor names can be retrieved for an IPMI based location using the ListTargets Dataitem.
none	Same as not entering a target.

8.5 Dataitem Parameter: -d

The Dataitem parameter is the Target or Location attribute that the user is getting or setting. Dataitem must be given for every CLI command. The Dataitem keywords are shown in the following table.

Table 23. dataitem (-d) Keywords for All Locations (Sheet 1 of 2)

Data Item	Description	Get/ Set	Valid Set Values:
criticalaction majoraction minoraction normalaction	Used to configure user defined actions when events occur. This dataitem is used with a target (–t) parameter specified sensor and a value (-v) parameter. When an event happens for that particular sensor, then the script defined in the –v parameter will be executed. The script to be executed must be located in the /home/scripts directory on the CMM and the /home/scripts path should be omitted when specifying the script. See Section 18.0, "CMM Scripting" on page 93 for more information.	Both	Valid filename of a script that resides in /home/scripts none
current	The current value of a sensor.	Get	N/A
health	Returns the health of the location and if any events exist.	Get	N/A
healthevents	Returns the specific health events that are occurring on the location if any exist.	Get	N/A
listdataitems	Used to find the data items available on a location. The output lists dataitems that the location supports.	Get	N/A
listtargets	Used to find what sensors or targets are available on the location. This is the list of sensors defined by the SDR for that particular location.	Get	N/A
presence	Used to find out if a particular location is occupied or present in the chassis. This can be blades or intelligent fan trays.	Get	N/A
sel	Returns a location's System Event Log.	Get	N/A

Table 23. dataitem (-d) Keywords for All Locations (Sheet 2 of 2)

Data Item	Description	Get/ Set	Valid Set Values:
thresholdsall	All thresholds of a sensor. This includes lower non-recoverable, lower critical, lower non-critical, upper non-critical, upper critical, and upper non-recoverable.	Get	N/A
uppercritical uppernoncritical lowercritical lowernoncritical uppernonrecoverable lowernonrecoverable	Used to query individual thresholds for a value based sensor, such as temperature or voltage.	Get	N/A
eventaction	Used to trigger a script based on event code of a health event. Refer to Section 18.0, “CMM Scripting” on page 93	Set	The event code of the event

Table 24. dataitem (-d) Keywords for System location

Data Item	Description	Get/ Set	Valid Set Values
unhealthylocations	Output: Critical: CritList Major: MajList Minor: MinList where each list is a list of locations having that level of health events (space separated). In the output, PSn refers to PowerSupplyN (i.e., PS1 = PowerSupply1, PS2 = PowerSupply2, etc.)	Get	N/A

Table 25. dataitem (-d) Keywords for BladeN Locations

Data Item	Description	Get/ Set	Valid Set Values:
ignorespcirst	Used to add IPMI capable drone mode compatible blades to the list of recognized drone mode boards. For further information on drone mode see Section 14.0, “Drone Mode SBC Support” on page 85.	Set	1 - Add board to drone mode list 0 - Delete board from drone mode list
powerstate	Used to query or control the power state of a board. This is also used to reset a board.	Both	poweron poweroff reset offline activate

Table 26. dataitem (-d) Keywords for Chassis Location

Data Item	Description	Get/ Set	Valid Set Values:
fanspeed	Used to set or query the fan speed of all the fans in the system as a percentage of their full speed. Some fan trays may prevent the fan speed from going to zero. Querying the fan speed does not show actual fanspeed, but reflects what the CMM fan speed output is set to as a percentage of the actual full fan speed. See Section 17.0, "Fan Control and Monitoring" on page 91 for further information. Note: In case of a temperature upper threshold violation, the fans will be automatically set by the CMM to run at full speed until the violation deasserts.	Both	100 (default) 80 0
healthyrampuptime	Used to set or query the value configured in the CMM for the maximum amount of time the CMM will wait for the HEALTHY # hardware signal to be asserted by an SBC.	Both	2 (default) through 60 in seconds
location	Used to set or query the location field in the chassis FRU.	Both	Location String of up to 16 characters
maxpowerupattempts	Used to set or query the value for the maximum number of tries the CMM will attempt to power up a board during the power sequencing loop.	Both	" -1" (Default) 0 through 2000000000
psinhibit[1..8]	Used to inhibit the power supplies in the chassis.	Both	1 - Enable 0- Inhibit
slotinfo	Queries chassis slot layout information. This will tell the user what slots are system slots, peripheral slots, or busless slots. It will also show which slots are occupied by blades.	Get	N/A

Table 27. dataitem (-d) Keywords for CMM Location (Sheet 1 of 2)

Data Item	Description	Get/ Set	Valid Set Values:
alarmcutoff	Used to engage the Alarm Cutoff (ACO) or get its value. When engaged, the ACO silences active alarms and blinks the alarm LEDs on the CMM. This Dataitem is only valid when used with the CMM as the location.	Both	1 - Set cut off 0 - Unset cut off
alarmtimeout	Used to set the Alarm Cutoff timeout value (in minutes) or get its value. This is the length of time the CMM waits before automatically canceling an engaged ACO (if the user fails to cancel it manually). A value of 0 disables the timeout. This dataitem is only valid when used with the CMM as the location.	Both	Number of minutes: 0-1000. Value of 0 disables the timeout.
criticalled minorled majorled	Used to turn on or off the critical, major and minor LEDs.	Set	1 - Turn On LED 0 - Turn Off LED
Ethernet	Used to change to the eth0 direction to either the front panel or the rear panel IO card.	Both	front (default) rear

Table 27. dataitem (-d) Keywords for CMM Location (Sheet 2 of 2)

Data Item	Description	Get/ Set	Valid Set Values:
failover	Used only from the active CMM to failover to the standby CMM. This command will only successfully execute if the standby CMM is in a healthy enough state to accept a failover.	Set	1 - Failover
redundancy	Used to query CMM redundancy information. Indicates CMM(s) presence, which CMM is active, which CMM is stand by, and which one the user is logged into.	Get	N/A
version	The version of the CMM software.	Get	N/A
snmptrapcommunity	Set or query the SNMP trap community string.	Both	Valid SNMP community name: public (default)
snmpenable	Sets or queries the SNMP trap enabled status.	Both	Enable (default) Disable
snmptrapaddress[1..5]	The IP address of the machines receiving SNMP traps from the CMM.	Both	IP Address in the form: x.x.x.x
snmptrapport	Sets or queries the TCP/IP port that the SNMP trap will be sent from.	Both	Valid port number: 0-65536 162 (default)
snmptrapversion	Sets or queries the SNMP trap version.	Both	V1(default) V3
update	Used with the cmmget command to update the CMM firmware on the CMM. The syntax of the command is: cmmset -l cmm -d update -v [Update Package path and filename] [force] [overwrite] [ftp:server:user:password] Refer toChapter 25.0, “Updating Software” for more info.	Set	Starts the update of the firmware on that CMM.

8.6 Value Parameter: -v

The Value parameter specifies the new value for a Dataitem. This parameter is required for all **cmmset** commands and is only used for **cmmset** commands. Valid value parameters are shown in with their corresponding data items in the data item tables listed above.

8.7 Sample CLI Operations

Sample CLI Operations can be found in Appendix D, “Example CLI Commands”.

8.8 Generating a System Status Report

The CLI includes an executable script (cmmdump) that is used to generate a system status report for use in communicating system health and configuration information to technical support personnel. This is useful in helping technical support successfully troubleshoot any issues that may be affecting the system. Cmmdump outputs system information to the screen by default or to a file. To send the output to a file use the following command:

```
cmmdump > [filename]
```

The filename should refer to a file that is in a valid directory (i.e., /home/cmmdump.txt). The file can then be retrieved off the CMM using FTP (see Section 7.2.3, “FTP Into the CMM” on page 54).

9.0 Resetting the Password

It may become necessary to reset the CMM password to its default of ***cmmrootpass***. The CMM has an onboard dip switch labeled SW1-1 to perform this action. See Section 2.10, "Onboard Switches" on page 31 for the location of the switch. Setting the switch and powering up the CMM will cause the password to reset to its default. The CMM then must be removed, and the switch must be turned off again.

9.1 Resetting the Password in a Single CMM System

For non-redundant systems that contain only a single CMM, resetting the password requires removing the CMM. This causes any boards that are power controlled by the CMM to power off.

1. If necessary, safely shut down and power off boards being power controlled by the CMM.
2. Remove the CMM from the system.
3. Turn on the dip switch SW1-1. The dip switch has a label indicating which way is on.
4. Reinsert the CMM into the system and allow the CMM to fully boot into Bluecat Linux*.
5. Once at the login prompt, the password should now be reset to its default of ***cmmrootpass***.
6. Log in to the CMM to ensure the password was reset.
7. Remove the CMM from the system.
8. Turn off dip switch SW-1 by moving it back to its original off position.
9. Reinsert the CMM into the system and allow the CMM to fully boot into Bluecat Linux.
10. Log in to the CMM and operate as normal.

9.2 Resetting the Password in a Dual CMM System

In redundant systems containing dual CMMs - one active, one standby - the password should be reset on the standby CMM. Once reset to its default, the default password will synchronize itself to the active CMM. This prevents the need to perform the reset on both CMMs and a failover.

1. Open the ejector latch on the standby CMM and wait for the blue hot swap LED to illuminate, indicating the CMM is safe to remove from the system.
2. A health event will occur on the active CMM indicating redundancy has been lost. A SEL entry will be recorded, and a trap will be sent out.
3. Remove the standby CMM from the chassis.
4. Turn on the dip switch SW1-1. The dip switch has a label indicating which way is on.
5. Reinsert the CMM into the system and allow the CMM to fully boot into Bluecat Linux.
6. A health event will occur indicating that the passwords on both CMMs have been reset and was synched from the standby CMM.
7. Once at the login prompt, the password should now be reset to its default of ***cmmrootpass***.
8. Login to the active CMM to ensure the password was reset.

9. Open the ejector latch on the standby CMM and wait for the blue hot swap LED to illuminate, indicating the CMM is safe to remove from the system.
10. A health event will occur on the active CMM indicating redundancy has been lost. A SEL entry will be recorded, and a trap will be sent out.
11. Remove the standby CMM from the chassis.
12. Turn off the dip switch SW1-1.
13. Reinsert the CMM into the system and allow the CMM to fully boot into Bluecat Linux.
14. The password can now be set on the active CMM and it will get synched to the standby CMM and the CMM can now be operated normally.

10.0 Sensor Types

The CMM provides access to and can log events from different IPMI sensor types. These sensors can be threshold based sensors or discrete sensors, depending on the type of reading and events they generate. For more information on sensors and sensor types, refer to the *Intelligent Platform Management Interface Specification v1.5*.

10.1 Threshold-Based Sensors

Threshold based sensors are sensors that generate or change an event status based on comparing a current value to a threshold value for a given hardware monitor device. Examples of threshold based sensors are temperature, voltage, and fan tachometer sensors.

10.1.1 Threshold-Based Sensor Events

Threshold-based sensors generate events when a current value for a device becomes greater than or less than a given threshold value. The IPMI specification defines six thresholds that can be assigned to a given sensor. The sensor will generate an event when its current reading goes above or below any of these thresholds. The severity of the event generated depends on which threshold is crossed. These six thresholds and their associated event severity are as follows:

Upper non-recoverable---> Critical Event

Upper critical--->Major Event

Upper non-critical--->Minor Event

Lower non-critical--->Minor Event

Lower critical--->Major Event

Lower non-recoverable--->Critical Event

10.2 Discrete Sensors

Discrete sensors are sensors that have a predefined set of states. An examples of a discrete sensor would be board presence, where a board is either present or not present.

10.2.1 Discrete Sensor Events

Discrete sensors can generate events on state changes. The severity of the event is determined by the CMM. Discrete sensor events for the CMM are located in the next section, Chapter 11.0, “Health Event Strings”.

11.0 Health Event Strings

This section describes the health event strings that are associated with CMM events and includes the syntax and severity of these strings. Refer to Section 10.0, “Sensor Types” on page 65 for more information on the different types of sensors.

Note: For this section, “health event” (two words) refers to any event that is generated that reports the health of a sensor. “healthevents” (all one word) refers specifically to the “healthevents” dataitem or the output of that dataitem (healthevents query).

11.1 Syntax of Health Event Strings

Health event string is a string that is associated with the particular health event occurring. Active events are displayed in a healthevents query using the following command in the CLI:

```
cmmget -l [location] [-t [target]] -d healthevents
```

Active health events are also returned when healthevents queries are executed over SNMP. In addition, all health events are logged in the SEL and sent out as SNMP Traps.

Note: SEL entries and SNMP traps do not include the severity of the event. Only healthevents query results display the severity of an event.

11.1.1 “healthevents” Query Event Syntax

The following is the syntax of a string returned by a healthevents query for an associated active health event. The \n represents newline characters (seen as <0A> in SNMP queries).

```
[Timestamp]\n
[Severity] Event : [SDR Sensor Name] [Health Event String]\n
```

- Timestamp is in the format: `[Day] [Month] [Date] [HH:MM:SS] [Year].` For example, Thu Dec 11 22:20:03 2003
- Severity is either Minor, Major, or Critical.
- SDR Sensor Name is the name given to the sensor in the Sensor Data Record (SDR).
- Health Event Strings are listed in Section 11.4, “List of Possible Health Event Strings” on page 70.

11.1.2 SEL Event Syntax

The following is the syntax of each SEL entry for an associated health event. The \n represents newline characters and the \t represents tab characters.

```
[Timestamp]\n
\t[SDR Sensor Name]\t[Health Event String]\n_____\n
```

- Timestamp is in the format: `[Day] [Month] [Date] [HH:MM:SS] [Year].` For example, Thu Dec 11 22:20:03 2003
- SDR Sensor Name is the name given to the sensor in the Sensor Data Record (SDR).
- Health Event Strings are listed in Section 11.4.

11.1.3 SNMP Trap Event Syntax

The following is the syntax of each SNMP trap for an associated event:

```
Time : [TimeStamp] , Location : [ChassisLocation] , Chassis Serial # :
[ChassisSN] , Board : [Location] , Sensor : [SDR Sensor Name] , Event :
[Health Event String]
```

- Timestamp is in the format: `[Day] [Month] [Date] [HH:MM:SS] [Year].` For example, Thu Dec 11 22:20:03 2003
- ChassisLocation is the chassis location information recorded in the chassis FRU.
- ChassisSN is the chassis serial number recorde in the chassis FRU.
- Location is the location where the sensor generating the event is located (i.e., CMM)
- SDR Sensor Name is the name given to the sensor in the Sensor Data Record (SDR).
- Health Event Strings are listed in Section 11.4, “List of Possible Health Event Strings” on page 70.

11.2 Sensor Targets

Available sensors for a location can be retrieved with the listtargets dataitem argument of the cmmget command. To view a list of sensor targets on the cmm using the listtargets command:

```
cmmget -l cmm -d listtargets
```

The CMM lists the following sensors that can be queried for health events:

Table 28. CMM Sensor Targets

Sensor Target	Description
Brd Temp	Board Temperature Sensor
+1.5V	+1.5 Volt Sensor
+2.5V	+2.5 Volt Sensor
+3.3V	+3.3 Volt Sensor
+5V	+5 Volt Sensor
Eth0 Interface	Eth0 LAN Sensor
Eth1 Interface	Eth1 LAN Sensor
Eth1:1 Interface	Eth1:1 LAN Sensor
TASensor1	Telco Alarm Input Sensor 1
TASensor2	Telco Alarm Input Sensor 2
BIST	Built-in Self Test Sensor Type

For complete lists of sensors on other components (i.e., voltage sensors on blade1), refer to the corresponding Technical Product Specification for that product.

11.3 Healthevents Queries

Healthevents queries, whether done from the CLI or through SNMP, output the same event strings and syntax for active events. This section explains the various types of output that are retrieved from healthevents queries. For more information on using the healthevents dataitem, refer to Section 8.0, “Command Line Interface (CLI) Syntax and Arguments” on page 57.

11.3.1 HealthEvents Queries for Individual Sensors

Executing a healthevents query on a particular sensor target returns all active healthevents for that sensor target in a concatenated string. One sensor may have multiple events. For example, running the following healthevents query on the CMM BIST sensor:

```
cmmget -l cmm -t BIST -d healthevents
```

might return multiple events that are active on the BIST sensor in a concatenated string like this:

```
Mon Feb 2 19:51:05 2004
Major Event : BIST RTC Not Working
Mon Feb 2 19:51:09 2004
Major Event : BIST Both Ethernet interfaces are not working.
```

11.3.2 HealthEvents Queries for All Sensors on a Location

Executing a healthevents query on the “cmm” location in the CLI without a target specified (*cmmget -l cmm -d healthevents*) returns all the healthevents for all CMM sensors in a concatenated string. This includes all BIST, LAN, Telco Alarm, Voltage, and Temp sensors on the CMM. This ability to retrieve all healthevents on a location also applies to the chassis, bladeN, FantrayN and powersupplyN* locations.

* - PowersupplyN healthevents are dependent on the presence of IPMI-compliant power supplies. The “pwr supply N” target on the chassis location is used for querying non-IPMI capabilities of power supplies (i.e., Failure Detected). Querying the healthevents on the chassis location will return all active events for all “pwr supply N” targets.

The CMM’s SNMP interface provides a “cmm” target that can be found in the listtargets ouput. This target is used to execute healthevents queries for all active events on the CMM. Since each OID in the MIB requires a target on the location, the “cmm” target was implemented in SNMP to give the user the ability to do a healthevents query of the cmm location equivalent to the CLI command “cmmget -l cmm -d healthevents”. Similar SNMP-specific targets also exist for chassis and bladeN that allow users to retrieve all active events for that location. Refer to Section 19.0, “SNMP” on page 97 for more information on SNMP.

11.3.3 No Active Events

When a healthevents query is executed in the CLI on a target that doesn’t have active events a string will be returned that is a single line string with no timestamp or severity. Only this string will be returned and won’t be concatenated with any other strings. For example, doing a healthevents query from the CLI of a location or target that doesn’t have any active healthevents will return the following string:

“[Target] has no problems.” - i.e., “chassis has no problems” or “temp snsr 1 has no problems”

Executing a healthevents query through SNMP on a target with no active events returns different values than the CLI does. When a healthevents query is executed in SNMP on:

- A location target (i.e., in SNMP, the cmm target on the cmm location) that has no active events, the value returned is a zero length string.
- An individual sensor (i.e., the +3.3V sensor on the CMM) that has no active events, the value returned is a null value.

Note: Attempting to execute a healthevents query on the "FRU" target will return a "CLI Invalid Data Item Error." string since "FRU" is not a sensor.

11.3.4 Not Present or Non-IPMI Locations

Doing a healthevents query of a blade or powersupply, or a target on a blade or powersupply, that is not present or non-IPMI compliant will return one of the following:

"BPM Blade Not Present." - if querying an empty blade slot.

"BPM Non IPMI Blade." - if querying a blade that is present but doesn't support IPMI.

"Power Supply Not Present." - if querying an empty power supply slot or a power supply that doesn't support IPMI.

11.4 List of Possible Health Event Strings

Table 29 through Table 40 list possible event strings for sensors on the CMM, chassis, fan tray, and powersupply. Each event will log an entry in the SEL and generate an SNMP Trap, and can be retrieved through a healthevents query on a target or location. Locations and sensor names are listed as well.

The event severity is not displayed in the SEL entry or the SNMP trap. The severity of the event (Minor, Major, or Critical) is displayed in a healthevents query while the event is active. OK HealthEvents are displayed in healthevents queries for only a limited time when they are asserted.

For each event, the event code is listed in base 10 format for use with event scripting, which is documented in Section 18.0, "CMM Scripting" on page 93.

The table for threshold-based sensors is common to other threshold-based sensors on other components (e.g., voltage, temp, current).

11.4.1 All Locations

Table 29. Threshold-Based Sensor Event Strings (Voltage, Temp, Current, Fan)

Event String	Event Code	Event Severity
"Lower non-critical going low asserted "	16	Minor
"Lower non-critical going high asserted "	17	Minor
"Lower critical going low asserted"	18	Major
"Lower critical going high asserted"	19	Major
"Lower non-recoverable going low asserted"	20	Critical
"Lower non-recoverable going high asserted"	21	Critical
“Upper non-critical going low asserted"	22	Minor
“Upper non-critical going high asserted"	23	Minor
"Upper critical going low asserted"	24	Major
"Upper critical going high asserted"	25	Major
"Upper non-recoverable going low asserted"	26	Critical
"Upper non-recoverable going high asserted"	27	Critical
"Lower non-critical going low deasserted "	28	OK
"Lower non-critical going high deasserted "	29	OK
"Lower critical going low deasserted"	30	OK
"Lower critical going high deasserted"	31	OK
"Lower non-recoverable going low deasserted"	32	OK
"Lower non-recoverable going high deasserted"	33	OK
"Upper non-critical going low deasserted"	34	OK
"Upper non-critical going high deasserted"	35	OK
"Upper critical going low deasserted"	36	OK
"Upper critical going high deasserted"	37	OK
"Upper non-recoverable going low deasserted"	38	OK
"Upper non-recoverable going high deasserted"	39	OK

Note: For Threshold-based sensor events, strings for SEL entries are the same except they begin with the word “Crossed” (i.e., “Crossed Upper critical going high asserted")

11.4.2 PowerSupplyN Location (IPMI-Compliant Supplies Only)

Table 30. Power Supply Event Strings (PowerSupplyN)

Event String	Event Code	Event severity
"Power Supply feed lost"	51	Critical
“Power Supply feed lost or out of range”	52	Critical
"Recovered from error conditions"	48	OK

Note: “Power Supply feed lost” event string is currently not implemented.

11.4.3 CMM Location

Table 31. BIST Event Strings (BIST) (Sheet 1 of 2)

Event String	Event Code	Event Severity
“Boot is Successful.”	112	OK
“Event-log area full.”	113	Critical
“RedBoot image corrupted.”	114	Major
“Backup RedBoot image corrupted.”	115	Critical
“FPGA image corrupted. Using backup image.”	116	Major
“Backup FPGA image corrupted.”	117	Critical
“OS image corrupted.”	118	Critical
“Base memory test failed.”	119	Critical
“Extended memory test failed.”	120	Critical
“FPGA1 firmware outdated.”	121	Minor
“FPGA2 firmware outdated.”	122	Minor
“FPGA 1+2 version mismatched.”	123	Minor
“RTC date is invalid.”	124	Major
“RTC time is invalid.”	125	Major
“RTC is not working.”	126	Critical
“One of the Ethernet interfaces is not working.”	127	Minor
“Both Ethernet interfaces are not working.”	128	Major
“CMM Boot.”	129	OK
“IPMB bus [0-29] busy/not ready.”	130 - 159	Major
“Software update successful.”	160	OK
“Update of RedBoot failed.”	161	Critical
“Update of FPGA failed.”	162	Critical
“Update of Bluecat OS failed.”	163	Critical
“Update of /etc failed.”	164	Critical
“Restore of /etc files failed.”	165	Critical
“Software update failed.”	166	Critical
“Standby CMM has completed FPGA re-programming.”	179	OK
“FPGA re-programmed 2 times and no further lockup detected.”	180	Minor
“FPGA re-programmed 3 times and no further lockup detected.”	181	Minor
“FPGA re-programming has failed.”	182	Critical
“FPGA re-programmed more than 3 times and lockup still detected.”	183	Critical
“Dynamic FLASH image corruption detected.”	184	Critical
“Static Redboot image is corrupt.”	185	Critical
“Static OS image is corrupt.”	186	Critical
“Static FPGA image is corrupt.”	187	Critical
“Primary FRU unreadable - using Secondary"	188	Minor

Table 31. BIST Event Strings (BIST) (Sheet 2 of 2)

Event String	Event Code	Event Severity
"FRU unreadable!"	189	Critical
"RTC battery power lost."	190	Critical
"I2C failure while reading RTC."	191	Critical
"Primary FRU is corrupted."	272	Major
"Secondary FRU is corrupted."	273	Major
"FRU is corrupted!"	274	Critical

Table 32. Telco Alarm Sensor Event Strings (TASensor[1,2])

Event String	Event Code	Event Severity
“asserted.”	96	Major
“deasserted.”	97	OK

Table 33. LAN Sensor Event Strings (Eth[0,1,1:1] Interface)

Event String	Event Code	Event Severity
“Duplicate IP address detected on the network.”	84	Major
“Duplicate IP address corrected.”	85	OK

11.4.4 Chassis Location

Table 34. Fan Tray Presence Event Strings (Fan Tray [1-3] Presence)

Event String	Event Code	Event Severity
"Fan Tray Removed"	64	Major
"Fan Tray Inserted"	65	OK

Table 35. Chassis Power Supply Event Strings (Pwr Supply N)

Event String	Event Code	Event Severity
"Recovered from error conditions"	48	OK
"Power Supply Failure detected"	49	Critical
"Power Supply Degraded"	50	Minor
“Power Supply detected”	53	OK
“Power Supply removed”	54	Major

Note: Chassis power supply detected and removed events are only generated with IPMI power supplies.

Table 36. Slot Event Strings (Slot N Event)

Event String	Event Code	Event Severity
"Blade Removed"	192	OK
“Blade Presence Detected”	193	OK
"Blade Deasserted HEALTHY#"	194	OK
"Blade Asserted HEALTHY#"	195	OK
“Blade Failed HEALTHY# After Maximum Attempts”	196	OK
“Blade Powered On”	197	OK
“Blade Powered Off”	198	OK
“Blade Reset”	199	OK
“Blade set to Offline”	200	OK
“Blade set to Active”	201	OK
“Blade is not IPMI capable”	202	OK
"Blade is IPMI capable"	203	OK

Table 37. CMM Redundancy Event Strings (CMM [1-2] Redundancy)

Event String	Event Code	Event Severity
"Regained"	208	OK
"Established"	209	OK
"Lost due to switch failure"	210	Major
"Lost due to unhealthy signal"	211	Major
"Lost due to failed network connectivity between the CMM and the Primary SNMP trap destination (SNMPTrapAddress1)."	212	Major
"Lost due to CMM removal"	213	Major
"Lost due to CMM reboot"	214	Major
"Lost due to critical event on the CMM"	215	Major
"Lost due to inability to communicate with the other CMM over its management bus."	216	Major
"Lost due to incompatible firmware versions on the CMMs. Please upgrade the CMMs to the same version."	239	Major

Table 38. CMM Failover Event Strings (CMM Failover)

Event String	Event Code	Event Severity
"Failover occurred."	217	OK
"Failover occurred because of switch failure."	218	OK
"Failover occurred because of failed network connectivity between the CMM and the Primary SNMP trap destination."	219	OK
"Failover occurred because of critical CMM health events."	220	OK
"Failover occurred because of a bad hardware signal from the other CMM."	221	OK
"Failover occurred because it was forced by the user."	222	OK

Table 38. CMM Failover Event Strings (CMM Failover)

Event String	Event Code	Event Severity
"Failover occurred because the other CMM rebooted."	223	OK
"Failover cannot occur because the other CMM has a bad switch."	224	OK
"Failover cannot occur because the other CMM has lost network connectivity with its Primary SNMP trap destination."	225	OK
"Failover cannot occur because the other CMM has critical health events."	226	OK
"Failover cannot occur because the other CMM is not responding over its management bus."	227	OK
"Failover cannot occur because the critical items have not been synced."	228	OK
“Failover cannot occur because the other CMM has a bad hardware signal.”	229	OK
“Failover occurred because the active CMM had older firmware than the newly active CMM.”	240	OK

Note: In the CMM Redundancy and Failover event strings, the phrases “bad switch” and “switch failure” refer to the switch that is associated with that CMM having deasserted its HEALTHY# signal. The “bad hardware signal” refers to the CMM itself having deasserted its HEALTHY# signal.

Table 39. CMM File Sync Event Strings (CMM File Sync)

Event String	Event Code	If HealthEvents, its severity
"Could not copy the /etc/CMM.cfg file to the standby CMM."	230	OK
"Could not copy the /etc/passwd and /etc/shadow files to the standby CMM."	231	OK
“Default password files synced to active CMM. Passwords have been reset to default.”	232	OK
"Could not copy the /etc/sel_* files to the standby CMM."	233	OK
"Could not copy the /etc/snmpd.conf and /etc/var/snmpd.conf files to the standby CMM."	234	OK
"Could not copy the SDR file to the standby CMM."	235	OK
"Could not copy the /etc/scripts directory to the standby CMM."	236	OK
"All files successfully synced."	237	OK

Table 40. Chassis Sensor Event Strings

Event String	Event Code	Event Severity
“No PCI connectivity record found in chassis FRU information.”	256	Critical
“cmm.sif file not found. Cannot scan CMM sensors.”	257	Critical
“chassis.sif file not found. Cannot scan Chassis sensors.”	258	Critical

12.0 Slot Power Control

The chassis management module controls power to the node and fabric slots of a chassis using the BD_SEL# and HEALTHY# hardware signals. The CMM can power up, power down, and reset a board in a particular slot, as well as to place a slot into an offline mode. The CMM can also be used to query the power state of a board at any time.

12.1 Difference between a Board's HEALTHY# Signal and a Board's Health

Confusion may arise as to the difference between the HEALTHY# hardware signal coming from a board and the board's health retrieved using the CLI command:

```
cmmget -l bladeN -d health
```

A board's HEALTHY# signal is a PICMG-specified hardware signal on the CompactPCI connector. The PICMG 2.0 specification defines HEALTHY# to be a signal generated by a board's hot swap control logic. At a minimum, HEALTHY# should be asserted by the hot swap control when all of the voltages are good. Additional logic may be implemented above this to generate the HEALTHY# signal. The CMM uses the HEALTHY# hardware signal to determine when a board is powered up completely.

The board's health, as returned by the CLI command listed above, is used to determine whether there are any active health events occurring on a particular board, such as temperature or voltage violations.

12.2 Slot Power-Up Sequence

The CMM constantly polls the status of the BD_SEL# lines coming from the node and fabric slots. Boards inserted into the chassis will pull the BD_SEL# line to Vcc, indicating insertion of the board.

12.2.1 Assertion of BD_SEL#

Upon detection of board insertion, the CMM determines whether to assert BD_SEL# based on the type of board being inserted and the type of slot the board is being inserted to.

12.2.1.1 System Master Boards

For system master boards in system master slots, the CMM will always assert BD_SEL#.

12.2.1.2 Peripheral Boards

For peripheral boards inserted into peripheral slots, the CMM will ONLY assert BD_SEL# if a system master board is present for that PCI segment. If a peripheral board is inserted into a system master slot, or into a peripheral slot in a PCI segment with no system master, the CMM will not assert BD_SEL# for that board.

12.2.1.3 Fabric and PCI-Less Slots

For node slots containing no PCI bus (PCI-less slots) as well as fabric slots, the CMM will always assert BD_SEL#.

12.2.1.4 Drone Mode Boards

For drone mode capable boards that have been added to the CMM, the CMM will always assert BD_SEL# regardless of the type of node slot where they are inserted. For more information on drone mode and drone mode boards, see Section 14.0, “Drone Mode SBC Support” on page 85. All Intel SBCs are recognized as drone mode capable.

12.2.2 Assertion of HEALTHY# During Power-Up

Upon assertion of BD_SEL#, the CMM will wait a minimum of two seconds for the board to assert the HEALTHY# signal. The amount of time the CMM will wait for HEALTHY# to be asserted by an SBC can be configured from the CMM interfaces from 2 to 60 seconds and is referred to as the HEALTHY# ramp up time. See Section 12.5, “Power Sequencing Commands and Policies” on page 79 for instructions on how to set and retrieve the HEALTHY# ramp-up time.

A board should generate a HEALTHY# signal indicating all of its onboard voltages are good per the PICMG 2.1 R2.0 specification. Depending on the logic of the board, HEALTHY# can also be generated by additional logic, such as from a BIST, OS boot-up, etc.

If within the defined HEALTHY# ramp-up time the CMM detects HEALTHY# being asserted, the CMM will leave BD_SEL# asserted, and the board will go through its normal power-on sequence. BD_SEL# will remain asserted unless the HEALTHY# signal gets deasserted for that board or the command to power down a board is issued.

If within the defined HEALTHY# ramp-up time the CMM does not detect HEALTHY# being asserted, the CMM will deassert BD_SEL# to that slot, and that board will not power up. If the board remains inserted in the slot, the process will repeat on the next polling cycle of that slot. This process will continue until either the board powers up, the board is removed from the slot, or the maximum number of power up attempts is reached.

The maximum number of attempts the CMM will try to power on a board can be configured through the CMM interfaces. See Section 12.5, “Power Sequencing Commands and Policies” on page 79 for further information on how to set and retrieve the maximum power-up attempts.

12.3 Obtaining the Power State of a Board

The CMM can obtain the power state information of a board at any time by issuing the following command:

```
cmmget -l bladeN -d powerstate
```

Where N is the number of the slot in which the blade you are querying resides. This command will give information on whether the blade is present, the state of the HEALTHY# signal (asserted/ deasserted), whether the board has been powered up or not, and whether the slot is in the active or offline mode. Output will look similar to this:

```
Board is present
```

```
Board HEALTHY# signal is asserted

Board has been powered up by the CMM

Board is in active mode.
```

For further information on active and offline mode see Section 15.0, “Active vs. Offline Slots” on page 87.

12.4 Controlling the Power State of a Slot

The CMM can power a slot off, on, reset the slot, or place a slot in an offline mode. Placing a slot in offline mode allows the CMM to ignore the state of the HEALTHY# signal from that slot.

12.4.1 Powering Off a Board

The following command will power a board off:

```
cmmset -l bladeN -d powerstate -v poweroff
```

N is the number of the slot the blade to be powered down resides in. Once issued, the command will ask for confirmation by entering “y” before continuing.

12.4.2 Powering On a Board

The following command will power a board on:

```
cmmset -l bladeN -d powerstate -v poweron
```

N is the number of the slot the blade to be reset resides in.

12.4.3 Resetting a Board

The following command will reset a board:

```
cmmset -l bladeN -d powerstate -v reset
```

N is the number of the slot the blade to be reset resides in. Once issued, the command will ask for confirmation by entering “y” before continuing.

12.5 Power Sequencing Commands and Policies

The following section describes how to set and retrieve the HEALTHY# ramp up time and maximum power up attempts for the chassis.

HEALTHY# ramp up time is used to determine the maximum amount of time the CMM will wait for an SBC to generate the HEALTHY# hardware signal during the current power sequencing loop. If the amount of time defined in the *healthyrampuptime* expires, the CMM deasserts BD_SEL# to the slot, and continues through the power up sequence loop. For example, on initial power up, if the *healthyrampuptime* is set to two seconds, a blade can be reset during its powerup sequence. If set to 60 seconds, a reset command won’t take effect until 60 seconds after power on in which time the blade will typically be booted.

The maximum power-up attempts variable is used to configure the number of times the CMM will attempt to power on a board during the power sequencing loop. Once the *maxpowerupattempts* is exceeded for a particular board, the CMM will no longer attempt to power on the board by asserting the BD_SEL# signal during the power sequencing loop.

Setting these variables applies to the entire chassis, not just a single board or slot.

12.5.1 Retrieving the Healthy# Ramp-Up Time

Issuing the following command will retrieve the value configured in the CMM for the maximum amount of time the CMM will wait for the HEALTHY # hardware signal to be asserted by an SBC.

```
cmmget -l chassis -d healthyrampuptime
```

This returns the time (in seconds) the CMM will wait for an SBC to generate the HEALTHY# hardware signal for ALL node slots in the chassis. This time can range from 2 to 60 seconds. The default is 2 seconds.

12.5.2 Setting the Healthy# Ramp-Up Time

The following command can be used to set the maximum amount of time in seconds the CMM will wait for the HEALTHY# hardware signal to be asserted all SBCs in the chassis.

```
cmmset -l chassis -d healthyrampuptime -v [time]
```

Where *time* is a value between 2 and 60 seconds. The default is 2 seconds.

Setting this value applies to all boards in the chassis. For every newly inserted board, or every board being powered on, the CMM will always wait the allotted *healthyrampuptime*, even if the board asserts HEALTHY# before *healthyrampuptime* expires.

12.5.3 Retrieving the Maximum Power-Up Attempts

The following command is used to retrieve the configured value for the maximum number of tries the CMM will attempt to power up a board during the power sequencing loop.

```
cmmget -l chassis -d maxpowerupattempts
```

The default value for maxpowerupattempts is -1, and indicates an infinite number of attempts will be made by the CMM to power up boards in the chassis. This value applies to all boards in the chassis.

12.5.4 Setting the Maximum Power-Up Attempts

The following command is used to set the value for the he maximum number of tries the CMM will attempt to power up boards in the chassis during the power sequencing loop.

```
cmmset -l chassis -d maxpowerupattempts -v "[# of attempts]"
```

Where *# of attempts* is the desired number of attempts the CMM should try to power up boards in the chassis enclosed in quotes. The default is a value of " -1" and indicates the CMM will try an infinite number of times to power up boards.

Note: To set the maxpowerupattempts back to the default value of -1, you need to have a space between the first quote and the number 1 (i.e., " -1").

Setting a value of "0" will cause the CMM to discontinue attempting to power up boards in the chassis. Rebooting the chassis when the value is set to "0" will prevent the CMM from powering up any blades in the chassis.

The maximum value that can be set for maxpowerupattempts is 2000000000.

This value applies to all boards in the chassis. Changing this value to a greater value with boards that failed to power by reaching the previous maximum power up attempts will cause the CMM to incrementally attempt to power up the board with the difference between the two maximums. For example, for a board that failed to power up with a maxpowerupattempts of 5, changing the maximumpowerupattempts to a value of 12 will cause the CMM to attempt to power up that board an additional seven more times.

Failure of a board to power after reaching the maximum number of attempts will cause a SEL event to be logged indicating this.

12.5.5 Power Sequencing SEL Events

The following power sequencing events will cause an event to be recorded in the chassis SEL.

1. CMM detects the presence or removal of a board

```
Slot X Event  Blade Presence Detected
```

```
Slot X Event  Blade Removed
```

2. CMM detects the assertion/deassertion of the HEALTHY# signal from the board

```
Slot X Event  Blade Asserted HEALTHY#
```

```
Slot X Event  Blade Deasserted HEALTHY#
```

3. CMM detects a board has failed to power up after reaching the maximum power up attempts

```
Slot X Event  Blade Failed HEALTHY# After Maximum Attempts
```

4. CMM detects the powering on or off of a board

```
Slot X Event  Blade Powered On
```

```
Slot X Event  Blade Powered Off
```

5. CMM detects a board has reset

```
Slot X Event  Blade Reset
```

6. CMM detects a slot has been placed in offline or active

```
Slot X Event  Blade set to Offline
```

```
Slot X Event  Blade set to Active
```

13.0 Power Supplies

13.1 Power Supply Detection

Unlike blades, there is no hardware support for the detection of power supply presence. Therefore, an additional process is used to periodically scan for the presence of IPMI power supplies. Specifically, every five seconds, the process sends an IPMI GetDeviceID to each of the possible power supply slots. If a response is received, then it is known that an IPMI power supply is present.

13.2 Inhibit, Degrade, and Fail Signals

The CMM and power supplies do share several hardware signals that provide some limited functionality. These signals work with existing (non-IPMI) power supplies and are expected to work in the same way for the IPMI power supplies. The Inhibit signal is an output from the CMM that can be used to effectively turn off a power supply. Querying the Inhibit signal will indicate whether the CMM has inhibited the supply or not. When a power supply degrades or fails, it signals these conditions to the CMM using the Degrade and Fail inputs.

The CMM can detect if any power supply is Degraded or Failed, then it can be deduced that there is a power supply present. The new feature provided by IPMI supplies is the ability to detect the presence of a power supply when the power supply is operating normally.

13.3 Inhibiting Power Supplies

The CMM has the ability to inhibit any of the power supplies in the chassis. Each power supply can be inhibited individually.

The state of inhibited power supplies is synchronized from the active to the standby CMM. If a failover were to occur, the inhibited power supplies would remain inhibited.

The state of inhibited power supplies is not persistent across chassis power cycles.

13.4 Precautions For Inhibiting Power Supplies

Care should be taken when inhibiting power supplies. Inhibiting certain supplies or groups of supplies may cause the loss of power to portions of the chassis, including power to the CMMs. It is also possible to inhibit all of the power supplies, which causes a loss of power to the entire chassis, including the CMMs.

Once the CMMs lose power, they are no longer able to drive the INHIBIT# line to the power supplies, which causes all of the supplies to power up again. Losing power to both CMMs is equivalent to power cycling the chassis.

Care should be taken to prevent inhibiting all of the supplies in a chassis as well as losing power to both CMMs so that the reset condition does not occur.

13.5 Inhibiting Power Supplies

Power supplies can be inhibited through the CMM. This is accomplished using the following command:

```
cmmset -l chassis -d psinhibitN -v 0
```

Where N is the number of the power supply that is to be inhibited.

Note: When a non-IPMI supply is inhibited, a SEL entry is logged indicating a failure condition. When the non-IPMI supply is enabled a SEL entry is logged indicating that the power supply has recovered from a failure condition. This does not occur in current IPMI-compliant power supplies.

13.6 Enabling Power Supplies

To enable a supply which has been inhibited, issue the following command:

```
cmmset -l chassis -d psinhibitN -v 1
```

Where N is the number of the power supply that is to be enabled.

Note: When a non-IPMI supply is enabled a SEL entry is logged indicating that the power supply has recovered from a failure condition. This is because when a non-IPMI supply is inhibited, a SEL entry is logged indicating a failure condition. This does not occur in current IPMI-compliant power supplies.

14.0 Drone Mode SBC Support

An SBC is said to be “drone mode capable” if it can be configured to ignore the PCI_RST# signal coming from the PCI bus when inserted in a peripheral slot in a CompactPCI chassis. In this “drone mode,” the board is operating independently of the PCI bus.

Because it is not acting as a peripheral on the PCI bus, and therefore not dependant on the system master for correct initialization and power up, the normal methods used by the CMM for power sequencing a peripheral board do not apply to the drone board in that PCI segment. This allows two things to occur for drone boards that normally wouldn’t occur for standard peripheral boards:

- A drone mode capable board can be powered up in a peripheral slot without the system master slot being populated and running for that particular PCI segment.
- If the system slot board is extracted/goes unhealthy, the drone mode capable boards in that particular PCI segment should not be powered off.

14.0.1 Adding a Drone Mode Capable SBC

The CMM maintains an internal table of drone mode capable SBCs. For the CMM to recognize a drone mode capable board, it must also support IPMI 1.0 or above in addition to being able to operate in a drone mode. To add a drone mode capable SBC to the table issue the following CLI command, substituting the appropriate blade number for N:

```
cmmset -l bladeN -d IgnoresPCIRst -v 1
```

Once this command is issued, the CMM will send a GetDeviceID command to the blade and retrieve its manufacturer and product ID. It then adds this information to its internal table of drone mode capable SBCs. That drone mode capable SBC can now be configured to operate in drone mode and placed in any node slot within the chassis thereafter.

The CMM will then treat any board whose manufacturer and product ID match the one listed in the table as drone mode capable.

14.0.2 Removing a Drone Mode Capable SBC

To remove a drone mode SBC from the internal table on the CMM, issue the following command:

```
cmmset -l blade5 -d IgnoresPCIRst -v 0
```

This removes the listing for that manufacturer and product ID from the internal table on the CMM, and any boards matching this manufacturer and product ID thereafter will be treated as standard CompactPCI system or peripheral boards.

15.0 Active vs. Offline Slots

A slot operating normally is in what is defined as an active mode. The CMM has the capability to place a slot in an offline mode. In an offline mode the CMM ignores the HEALTHY # signal from the board in terms of power sequencing, and also does not send any health events that occur on the board. Placing a slot into offline mode allows for troubleshooting and debugging an unhealthy board.

If the board has an IPMI controller on it that is powered by IPMB power (standby), the CMM can access the IPMI controller and hardware monitors connected to it even if the board itself has powered down. A slot in which a board goes unhealthy can be placed into the offline mode so that the CMM can get information from the IPMI controller and hardware monitors on the board which can be used to diagnose problems with the board.

15.1 Determining the State of a Slot

Slots are set by default to the active mode, which allows for normal CompactPCI operation. A cmmget of the powerstate of a slot can be used to determine if a slot has been placed into an offline mode. See Section 12.3, “Obtaining the Power State of a Board” on page 78 for more information on obtaining the powerstate of a slot.

15.2 Setting a Slot to Offline Mode

The following command will place a slot in offline mode:

```
cmmset -l bladeN -d powerstate -v offline
```

N is the number of the slot to be placed in an offline mode.

15.3 Setting a Slot to Active Mode

The following command will place a slot into the active mode for normal operation:

```
cmmset -l bladeN -d powerstate -v activate
```

N is the number of the slow to be placed into the active mode.

15.4 Limitations of the Offline Mode

The CMM does not deassert BD_SEL# and power to an offline slot. Thus, when presence for an offline slot is queried, it will return as present even with the board extracted from the slot.

Also, because the CMM does not recognize board extractions/insertions in an offline slot due to BD_SEL# remaining asserted, the slot should be placed into active mode before a new board is inserted into a slot in the offline mode. If this is not done, the CMM retains all of the information of the board that was in the slot when it was placed in the offline mode.

16.0 Obtaining FRU Information

The CMM has the ability to obtain FRU information from IPMI 1.0 and higher compliant devices. The entire FRU can be retrieved, or individual fields from the FRU.

16.1 FRU Information

The FRU of an IPMI compliant device should be compliant with the "Platform Management FRU Information Storage Definition v1.0." In this specification, the FRU can contain information in any, all or none of the following areas: chassis info, board info, and product info.

The CMM can query the entire FRU of a device, entire areas of a FRU, or individual fields in the different areas of the FRU.

For detailed information on FRUs, please refer to the "Platform Management FRU Information Storage Definition v1.0."

16.2 FRU Query Syntax

To query FRU information, the CMM uses FRU as its target (-t) parameter. The format for querying the FRU of a particular location is:

```
cmmget -l [location] -t fru -d [FRUdataitem]
```

Where location is the component for which the FRU information is to be retrieved from, and FRUdataitem is the field(s) of the FRU which will be retrieved.

Table 41 lists the various FRU data items and the information they retrieve.

Table 41. Dataitems Used With FRU Target (-t) to Obtain FRU Information (Sheet 1 of 2)

Data Item	Description
all	Returns all FRU information for the location (Not supported over SNMP)
boardall	Lists all board area FRU information for the location (Not supported over SNMP)
boarddescription	Lists the description field in the FRU board area for the location
boardmanufacturer	Lists the manufacturer field in the FRU board area for the location
boardpartnumber	Lists the part number field in the FRU board area for the location
boardserialnumber	Lists the serial number field in the FRU board area for the location
boardmanufacturedatetime	Lists the manufacture date and time field in the FRU board area for the location
productall	Lists all product area FRU information for the location (Not supported over SNMP)
productdescription	Lists the description field in the FRU product area for the location
productmanufacturer	Lists the manufacturer field in the FRU product area for the location
productmodel	Lists the model field in the FRU product area for the location
productpartnumber	Lists the part number field in the FRU product area for the location

Table 41. Dataitems Used With FRU Target (-t) to Obtain FRU Information (Sheet 2 of 2)

Data Item	Description
productserialnumber	Lists the serial number field in the FRU product area for the location
productrevision	Lists the revision field in the FRU product area for the location
chassisall	Lists all chassis area FRU information for the location
chassispartnumber	Lists the part number field in the FRU chassis area for the location
chassisserialnumber	Lists the serial number field in the FRU chassis area for the location
chassislocation	Lists the location field in the FRU chassis area for the location
chassistype	Lists the type field in the FRU chassis area for the location
listdataitems	Displays a list of all of the FRU dataitems that can be queried for the FRU target

16.3 Chassis FRU Validity Check

Chassis FRU validity is read on a periodic basis (every 24 hrs). A reuse/redefinition of two existing BIST FRU error messages in place when one or both of the Chassis FRU cannot be read. When the FRU (usually primary) cannot be read, and another message for when both primary and secondary cannot be read

"Primary FRU unreadable - using Secondary"

"FRU unreadable!"

These two error messages are now used if the chassis FRU is detected to be corrupt will be a SEL entry and an error message will go into /home/log/error.log

17.0 Fan Control and Monitoring

The CMM controls the fan speeds and the fan tray LEDs. In a healthy state (no events) the LED changes to a green color. If any of the fan tray sensors (temperature, voltages, fan tachs) are in an unhealthy state, the LED changes to orange.

17.1 Setting Fan Speed

The factory default fan speed is 100 percent. The user can set the fan speed to 100 percent, 80 percent and 0 percent of the full speed through any of the supported interfaces. Setting the fan speed from the CMM changes the fan speeds for all of the fans in the chassis. Setting the fan speeds to 0 percent turns off the fans.

Note: The fan speed setting is persistent between CMM reboots.

The following command can be used to set the fan speed:

```
cmmset -l chassis -d fanspeed -v [100, 80, 0]
```

Where the value is 100 for 100 percent, 80 for 80 percent, or 0 for off.

Note: Querying fanspeed on the CMM will always reflect what the fan speed value was set to, not what the actual fan speed is.

17.1.1 Considerations for Turning Off the Fans

Some fan tray designs will not allow the fans to be turned off and will override the CMM setting with hardware on the fan trays. The CMM cannot detect fan tray hardware that provides such an override. Consult the fan tray or chassis documentation for further information on setting fan speeds.

For fan trays that do not allow fans to be turned off, querying the fanspeed on the CMM after setting it to zero will still show the fanspeed as set to zero because this is what is being output by the CMM.

Also, setting the fan speed to 0 (off) will override the automatic fan control of the CMM. Temperature and fan tray events will no longer cause the CMM to drive the fans to full speed.

17.2 Automatic Fan Control

The CMM will drive the fans to full speed (100 percent) under any of the following conditions:

- An upper temperature threshold is violated in the platform, including blades.
- A fan tray is removed from the chassis.
- A fan tachometer crosses a lower threshold.
- A fan board controller has a failure.

The fans are also driven to full speed automatically if both CMMs are removed from the chassis.

Note: Once the CMM automatically drives the fan speeds to 100 percent because of one of the above health events, it is still possible to set the fan speeds to 80 or 0 through the supported interfaces (CLI, etc). Once the automatic override of the fan speeds has been deactivated by resetting the fan speed to 80 or 0, the fans will remain at the newly set speed even after the condition that caused the automatic override has cleared.

17.3 Querying Fan Tray Sensors

To query the fan trays and fan tray sensors, use *fantrayN* as the location (-l) of the cmmget command, where N is the number of the fan tray being queried. For example, to query the current value of FANTK1 on fantray1, issue the command:

```
cmmget -l fantray1 -t “FANTK1” -d current
```

18.0 CMM Scripting

18.1 CLI Scripting

In addition to calling the CLI directly, commands can easily be called through scripts using “bash” shell scripting. These scripts could be tailored to create a single command from several CLI commands or to give more detailed information. For example, you may want to display all of the fans and their speeds on the server. A script could be written that would first call the CLI to find out what fan trays are present. Next it would find out what fan sensors are in each fan tray. Finally it would call the CLI to get the current speeds of each of the fans.

Scripts can be written directly on the CMM and should be saved on the CMM as a file in flash in the /home/scripts directory.

18.1.1 Script Synchronization

Script files are not synchronized automatically when copied or changed (by deleting, renaming, altering its chown/chmod/chgrp permissions, etc) in the /home/scripts directory on the active CMM. The Linux “touch” command should be issued to the /home/scripts directory on the active CMM to synchronize the scripts. Scripts are automatically synchronized on any full synchronization (chassis power up, insertion of a new CMM) as well as when they are edited (i.e., using vi and writing the file).

Scripts need to be deleted from both CMMs manually. Deleting a script on the active CMM does not automatically delete the script on the standby.

Note: Scripts are always synchronized from the active CMM to the standby CMM. Adding, editing, or modifying scripts on the standby CMM should be done with caution, and will need to be manually copied to the active CMM.

18.2 Associating Scripts to Event Severity

Health events triggered on the CMM can be used to execute scripts stored locally in the /home/scripts directory. Any level of an event can be used as a trigger: normal, minor, major, and critical.The CLI command for associating a script with an event is:

```
cmmset –l [location] –t [target] –d [action type] –v [script]
```

Location is the component in the chassis that the health event is associated with.

Target is the sensor to be triggered on.

Action type is NormalAction, MinorAction, MajorAction, or CriticalAction based upon the severity of the event to be triggered on.

Script is the script file or executable in the /home/scripts directory to be run including parameters to be sent to the script. The script and parameters should be enclosed in quotes. The *script* parameter should not include the /home/scripts path.

Note: If applicable, remember to set a script for when a sensor returns to normal (NormalAction). A script will not automatically stop running when a sensor returns to a normal setting (no alarms or events).

For example, if you wanted to run a blade powerdown script called “powerbladedown” stored in /home/scripts when the ambient temperature triggered a major event for a blade 4, the command would be:

```
cmmset –l blade4 –t “Ambient Temp” –d MajorAction –v “bladepowerdown 4”
```

In this example, the /home/scripts/bladepowerdown script or executable will be executed with “4” as a parameter when blade4’s Ambient Temp sensor generates a major health event.

18.2.1 Listing Scripts Associated With Events

The scripts that are associated with events are put into the /etc/cmm.cfg file. To display a list of scripts associated with events, cat the /etc/cmm.cfg file.

For further information on event scripting commands and its arguments, see Section 7.0, “CMM Connection and Setup” on page 53.

18.2.2 Removing Scripts From an Associated Event

To remove scripts from occurring for an event in which they have been associated, issue the following command:

```
cmmset –l [location] –t [target] –d [action type] –v none
```

Location is the component in the chassis that the health event is associated with.

Target is the sensor that was being triggered on.

Action type is NormalAction, MinorAction, MajorAction, or CriticalAction based upon the severity of the event that was being triggered on.

18.3 Setting Scripts for Specific Individual Events

The CMM allows scripts to be associated with specific events which may not necessarily be health related, such as the insertion or removal of a board from the chassis. This allows any single event that can occur on the CMM to have an associated script, or scripts with it.

18.3.1 Event Codes

To allow the user to set scripts based on any event, a unique event code is assigned to each event that can occur on the CMM. The list of events and the codes associated with each event is detailed in Section 11.0, “Health Event Strings” on page 67.

18.4 Setting Event Scripts

Setting event scripts can be done using any of the standard CMM interfaces (CLI, SNMP, RPC). The format for the CLI command is as follows:

```
cmmset  -l [location] -t [Sensor Name] -d EventAction -v [Event
Code]:[script to be run]
```

This setting gets written to the cmm.cfg file and is synced to the standby CMM. It is persistent across boots.

18.5 Slot Events

The CMM treats slot related events by assigning a sensor to each slot. These sensors will be called "*Slot X Event*" where X is the slot number. These will be discrete sensors and will show up in the chassis "listtargets" command. This allows an event script like to be associated to slot related events like any other sensor type.

For example, to launch a script called "*Notify*" when blade 5 is removed, the user would execute:

```
cmmset -l chassis -t "Slot 5 Event" -d EventAction -v [Removal Event
Code]:Notify
```

Assigning sensors to slot based events affects the strings shown in SNMP traps and SEL entries for slot events. The SEL entry and trap messages will now reflect the following:

```
Slot [Blade Number] Event [Event]
```

Where *Event* will be a string such as "Blade inserted", "Slot powered off", etc.

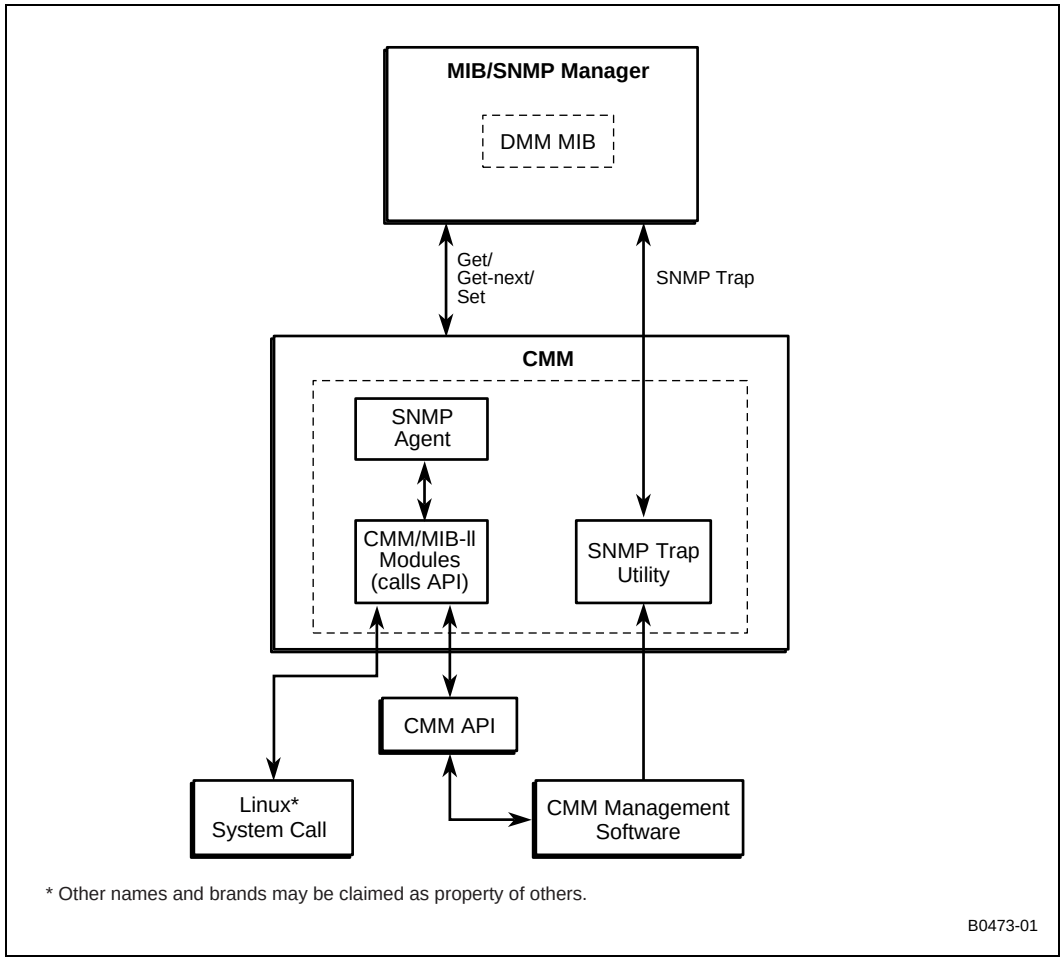
19.0 SNMP

The CMM supports the Simple Network Management Protocol (SNMP). The CMM can support SNMP queries and send SNMP traps in v1 or v3. The SNMP interface on the CMM very closely mirrors that of the CLI in both syntax and function.

Note: Like the CLI, SNMP commands should be issued to the active CMM. The standby CMM will only respond to commands for using the CMM location parameter.

The following diagram shows the high-level layout of the SNMP/MIB components and the flow of the data via SNMP protocol.

Figure 11. High Level SNMP/MIB Layout



19.1 CMM MIB

The CMM comes with a text file (ZT 7102.mib) that describes the CMM objects to be managed, as well as all of the managed objects in the platform. A remote application such as an SNMP/MIB manager can compile and read this file and utilize this information to manage the sensor devices on CMM, chassis, and server blades currently present. The ZT 7102.mib is located in the */usr/local/cmm* directory in the CMM. Users can utilize ftp or rcp to get this file from the CMM.

19.2 MIB Design

The CMM supports the following MIB II objects. The writeable objects can be set in their respective fields in the /etc/snmpd.conf file.

Table 42. MIB II Objects - System Group

Object	Syntax	Size	Access	Value
sysDscr	DisplayString	0..127	read-only	"Linux ZT 7102 2.4.2-1 #156 [DATE] armv5" Date is the actual current date.
sysObjectID	OBJECT IDENTIFIER		read-only	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).intel(343).products(2).Chassis-Management(14).ZT 7102(1)
sysUpTime	Days and Time		read-only	Time since ZT7102 began running
sysContact	DisplayString	0..127	read-write	max. 128 bytes
sysName	DisplayString	0..8	read-write	Default:"ZT 7102"
sysLocation	DisplayString	0..127	read-only	

Table 43. MIB II - Interface Group

Object	Syntax	Size	Access	Value
ifDscr	DisplayString	0..127	read-only	10/100BASE-TX

19.2.1 CMM MIB Objects

Detailed descriptions of the MIB objects can be found in the ZT7102.mib file located on the CMM.

19.2.2 Querying Non-IPMI Compliant Blades and Power Supplies

Slot control is a target that is specific to SNMP that is displayed in response to a listtargets command when used on an unoccupied slot in the chassis. This target is not available if the slot is occupied. The Slot Control target allows the user to perform other commands against that slot without an IPMI-compliant blade or IPMI-compliant power supply present.

19.3 SNMP Agent

The SNMP agent (snmpd) listens to SNMP v1 queries (gets/sets) by default, evokes the corresponding MIB Module to process the request, and sends the SNMP response with return data to the SNMP/MIB manager. All SNMP "set" queries are logged in the command log file.

Note: Changing IP addresses when using SNMP v3 will require a restart of the snmpd process. This is due to SNMP v3 using the IP address to generate the encryption keys. To restart the snmpd process follow steps 2 through 4 in Section 19.3.2.

19.3.1 Configuring the SNMP Agent Port

The SNMP agent is set up to use port 161 by default. The port can be configured to use a different port number by adding the following line to the snmpd.conf file:

```
agentaddress port_number
```

19.3.2 Configuring the Agent to Respond to SNMP v3 Requests

The agent can also be configured to respond to SNMP v3 queries, in which case it checks the security, decodes the request, then evokes the corresponding MIB Module to process the request, and sends the SNMP response with return data to the SNMP/MIB manager. SNMP v3 adds support for strong authentication and private communication.

To change the SNMP agent to respond to SNMP v3 queries:

1. Copy /etc/snmpdv3.conf to /etc/snmpd.conf.
2. Find the process ID of the snmpd process by issuing the command:

```
ps -ax
```

3. Look for the snmpd process. There may be more than one, so use the lowest process ID, which is the first instance of the process. The listing will look similar to this:

```
121 ?        SN     0:00 /usr/sbin/snmpd -c /etc/snmpd.conf
```

4. Restart the snmpd agent by issuing the following command:

```
kill -s SIGHUP [snmpd_process_id]
```

Where snmpd_process_id is the process ID of the snmpd process found in the step above.

19.3.3 Configuring the Agent Back to SNMP v1

To reconfigure the agent back to SNMP v1, follow the same steps as above, substituting the /etc/snmpdv1.conf for /etc/snmpdv3.conf. The rest of the process is the same.

19.3.4 Setting up an SNMP v3 MIB Browser

To manage the CMM via SNMPv3 MIB browser/manager (mg-soft, net-snmp, etc.), the user needs to configure the browser with the following parameters:

1. Load and compile the ZT 7102.MIB file
2. Set the SNMPv3 security parameters:
   - Set SNMPv3 agent user
   - At default, User: **root**
   - Set the MD5 Authentication password: **cmmrootpass**
   - Set the DES Encryption password: **cmmrootpass**

19.4 SNMP Trap Utility

The SNMP trap utility is utilized by the CMM management software to send SNMP trap messages to a remote application regarding any abnormal system events. When enabled, the CMM will issue SNMP v1 traps on port 162. The CMM can also be configured to issue SNMP v3 traps. The SNMP trap port can also be configured. For a description of all the traps that are issued, refer to Section 11.0, “Health Event Strings” on page 67.

19.4.1 Configuring the SNMP Trap Port

To configure the SNMP Trap Port to a different port number, issue the command:

```
cmmset -l cmm -d SNMPTrapPort -v [Port Number]
```

Where Port Number is the desired SNMP Trap Port number.

19.4.2 Configuring the CMM to Send SNMP v3 Traps

To configure the CMM to send SNMP v3 Traps, issue the command:

```
cmmset -l cmm -d SNMPTrapVersion -v v3
```

19.4.3 Configuring the CMM to Send SNMP v1 Traps

To configure the CMM to send SNMP v1 Traps, issue the command:

```
cmmset -l cmm -d SNMPTrapVersion -v v1
```

19.5 Configuring and Enabling SNMP Trap Addresses

The CMM allows up to five SNMP trap addresses destinations, SNMPTrapAddress1-5. In redundant CMM systems, **SNMP Trap Address 1 MUST be set to a valid IP address on the network that the CMM can ping to at all times.** This is used as a test of network connectivity as well as being the first SNMP Trap Address.

19.5.1 Configuring an SNMP Trap Address

To configure an SNMP trap address, issue the command:

```
cmmset -l cmm -d SNMPTrapAddressN -v [IP address]
```

where N is the number of the trap address from 1 to 5 that is being set, and IP address is the IP address of the trap receiver.

19.5.2 Enabling and Disabling SNMP Traps

SNMP Trap addresses are enabled by default. To disable SNMP traps, issue the command:

```
cmmset -l cmm -d SNMPenable -v disable
```

To enable SNMP trap addresses, issue the command:

```
cmmset -l cmm -d SNMPenable -v enable
```

To check the status of SNMP traps, issue the command:

```
cmmget -l cmm -d SNMPenable
```

19.5.3 Alerts Using SNMPv3

The CMM utilizes the SNMPtrap utility to send the SNMPv3 'trap' message**.** For receiving the SNMPv3 Trap, the remote application such as the trap listener needs to:

1. Set the SNMPv3 Trap User. The default trap user is **root**.
2. Set the MD5 Authentication password. The default MD5 Authentication password: is **publiccmm**.
3. Set the DES Encryption password. The default DES Encryption password is **publiccmm**.

Note: To change the passwords (MD5 and DES) for SNMPv3 trap, change the snmpTrapCommunity from the CLI interface or from the SNMP manager console.

19.5.4 Alert Using UDP Alert

The CMM Management software also sends the alert via UDP port 10000. The IP address to receive this alert is the same as the SNMP Trap IP addresses. The UDP alert is enabled by default when the user enables SNMP traps.

19.6 Security Model

19.6.1 SNMPv3 Security: Authentication Protocol and Privacy Protocol

The CMM supports the highest security level for SNMPv3. The MD5 is used for authentication Protocol in the CMM. The DES is used for Privacy Protocol in the CMM. When in this mode, the user needs to specify each password (authKey, privKey) for these protocols. The SNMPv3 packet is securely encrypted during the transmission.

This is the default security level of the CMM when configured for SNMP v3.

The following fields are defined to handle all SNMPv3 security levels:

Table 44. SNMPv3 Security Fields

SecurityName	user name	Default Value:
AuthProtocol	authentication type (MD5)	
AuthKey	authentication password	**publiccmm**
PrivProtocol	privacy type (DES)	
PrivKey	privacy password	**publiccmm**

19.7 snmpd.conf

For more information regarding SNMP configuration, and the snmpd.conf file, please visit:

http://www.net-snmp.org/man/snmpd.conf.html

20.0 RPC

In addition to the command-line interface, the CMM may be administered by custom remote applications through an RPC.

RPC calls are useful for managing the CMM from:

- A remote machine through the management network
- Another blade in the same chassis as the CMM, through the chassis backplane network
- An application running on the CMM itself

The RPC interface supports the same functionality as the command-line interface.

Note: System Event Log (SEL) information is currently not available through the RPC interface.

20.1 Setting Up the RPC Interface

Before you can use RPC calls in a custom application, you must download the following RPC source code files from the Intel developer web site located at:

http://developer.intel.com/design/network/products/cbp/zt7102.htm

- rcliapi.h
- rcliapi_xdr.c
- rcliapi_clnt.c
- cli_client.c
- cli_client.h

The first three files should be compiled and linked into the application under development. These files implement the RPC calling subsystem for use in an application.

The file cli_client.c contains a small sample program for calling the CMM through RPC. These files currently support Linux and VxWorks*. Future revisions may support Microsoft* Windows*.

The last file, cli_client.h, contains declarations and function prototypes necessary for interfacing with the RPC calling subsystem. Include the file in a '#include' statement in all application files that make RPC calls.

20.2 Using the RPC Interface

The RPC interface may be used to manage the CMM whether the calling application is on a remote network, on a blade in the same chassis as the CMM, or even running on the CMM itself.

The following two functions are defined by the RPC subsystem for calling the CMM through RPC:

- GetAuthCapability()
- ChassisManagementApi()

20.2.1 GetAuthCapability()

The following is the calling syntax for GetAuthCapability():

int **GetAuthCapability(**
char* *pszCMMHost*,
char* *pszUserName*,
char* *pszPassword*
);

Parameters
pszCMMHost:
[in] IP Address or hostname of CMM
pszUserName:
[in] A valid CMM user name
pszPassword:
[in] Password corresponding to *pszUserName*

Return Value
>0
Authentication successful. The return value is the authentication code.

-1
Invalid username / password combination.

E_RPC_INIT_FAIL
RPC initialization failure.

E_RPC_COMM_FAIL
RPC communication failure.

GetAuthCapability is used to authenticate the calling application with the remote CMM. The remote CMM will not respond to RPC communications until the application has successfully authenticated. To authenticate, the application will need to pass the CMM's current IP address, username and password to GetAuthCapability(). The default username and password are 'root' and 'cmmrootpass'. When the authentication is successful authentication, GetAuthCapability() will return an authentication code for use with all further RPC communication.

Note: Clients will need to re-authenticate to the CMM whenever the CMM is reset. Re-authentication is necessary when ChassisManagementApi() returns E_ECMM_SVR_AUTH_CODE_FAIL.

20.2.2 ChassisManagementApi()

The following is the calling syntax for ChassisManagementApi():

int **ChassisManagementApi**(
char* *pszCMMHost*,
int *nAuthCode*,
unsigned int *uCmdCode*,
unsigned char* *pszLocation*,
unsigned char* *pszTarget*,
unsigned char * *pszDataItem*,
unsigned char * *pszSetData*,

```
void **          ppvbuffer,
unsigned int *   uReturnType
);
```

Parameters

pszCMMHost
[in] IP Address or DNS hostname of CMM.

nAuthCode
[in] Authentication code returned by GetAuthCapability().

uCmdCode
[in] The command to be executed (CMD_GET or CMD _SET as defined in cli_client.h).

pszLocation
[in] The location that contains the data item that *uCmdCode* acts upon, such as system, CMM, or blade1.

pszTarget
[in] The target that contains the attribute that *uCmdCode* acts upon, such as the sensor name as listed in the Sensor Data Record (SDR). When not applicable, use "NA" (such as when *pszDataItem* is an attribute of the *pszLocation* rather than *pszTarget.*

pszDataItem
[in] The attribute that *uCmdCode* acts upon, either an attribute of *pszLocation* or *pszTarget.*

pszSetData
[in] The new value to set. When not applicable, use "NA".

ppvbuffer
[out] A pointer to the buffer containing the returned data.

uReturnType
[out] The type of data that *ppvbuffer* points to (see the defines in cli_client.h).

The value definitions of the return codes may be found in Table , “” on page 106.

Once the application has authenticated, it may proceed to get and set CMM parameters by calling ChassisManagementApi(). For each call to ChassisManagementApi(), the calling application must pass in the authentication code returned from GetAuthCapability(). The get and set commands available through ChassisManagementApi() are the same as those available on the command-line interface with cmmget and cmmset.

Note: System Event Log (SEL) information is currently not available through the RPC interface. This limitation may be removed in a future CMM firmware release.

For more information on the command-line interface, see Section 8.0, “Command Line Interface (CLI) Syntax and Arguments” on page 57.

Table 45. Error and Return Codes for the RPC Interface (Sheet 1 of 4)

Code #:	Error Code String	Error Code Description
0	E_SUCCESS	Success
1	E_BPM_BLADE_NOT_PRESENT	Blade isn't in the chassis
2	E_ECMM_SVR_COMMAND_UNSUPPORTED	ECMM_SVR: Unsupported Command Error
3	E_CLI_MSG_SND	CLI Send Message Error
4	E_CLI_INVALID_TARGET	Not a valid -t parameter.
5	E_CLI_INVALID_LOCATION	Not a valid -l location.
6	E_CLI_INVALID_DATA_ITEM	Not a valid -d parameter.
7	E_CLI_INVALID_SET_DATA	Not a valid -v parameter.
8	E_CLI_INVALID_REQUEST	CLI Invalid Request Error
9	E_CLI_MSG_RCV	CLI Receive Message Error
10	E_CLI_NO_MORE_DATA	No data found to retrieve.
11	E_CLI_DATA_TYPE_UNSUPPORTED	CLI Data Type Unsupported
12	E_ECMM_CLIENT_CONNECT_ERROR	ECMM_CLIENT: RPC Connect Error
13	E_ECMM_SVR_AUTH_CODE_FAIL	Invalid auth code passed to RPC interface
14	E_CLI_STANDBY_CMM	Operation cannot Be Performed on Standby CMM
15	E_WP_INITIALIZING	The CMM is Initializing and Not Ready. Wait a few seconds and try again.
16	E_BPM_NON_IPMI_BLADE	Blade does not support IPMI
17	E_BPM_STANDBY_CMM	BPM Operation cannot Be Performed on Standby CMM
18	E_BPM_NO_MORE_DATA	Couldn't delete a board from the drone mode list
19	E_BPM_INVALID_SET_DATA	Not a valid -v parameter
20	E_CLI_INVALID_BUFFER	Internal CMM Error
21	E_CLI_INVALID_CMM_SLOT	Internal CMM Error
22	E_CLI_NO_MSGQ_KEY	Internal CMM Error
23	E_CLI_NO_MSGQ	Internal CMM Error
24	E_CLI_NO_MSGQ_LOCK	Internal CMM Error
25	E_CLI_NO_MSGQ_UNLOCK	Internal CMM Error
26	E_CLI_FILE_OPEN_ERROR	Internal CMM Error
27	E_CLI_CFG_WRITE_ERROR	CMM Config File Error
28	E_IMB_NO_MSGQ	Internal CMM Error
29	E_IMB_NO_MSGQ_KEY	Internal CMM Error
30	E_IMB_SEND_TIMEOUT	Internal CMM Error
31	E_IMB_DRIVER_FAILURE	Internal CMM Error
32	E_IMB_REQ_TIMEOUT	A blade is not responding to IPMI requests.
33	E_IMB_RECEIVE_TIMEOUT	A blade is not responding to IPMI requests.

Table 45. Error and Return Codes for the RPC Interface (Sheet 2 of 4)

Code #:	Error Code String	Error Code Description
34	E_IMB_COMPCODE_ERROR	An IPMI request returned with a non successful completion code. User should try the command again.
35	E_IMB_INVALID_PACKET	Invalid IPMI response. Blade may be returning invalid data.
36	E_IMB_INVALID_REQUEST	Invalid IPMI response. Blade may be returning invalid data.
37	E_IMB_RESPONSE_DATA_OVERFLOW	Invalid IPMI response. Blade may be returning invalid data.
38	E_IMB_DATA_COPY_FAILED	Internal CMM Error
39	E_IMB_INVALID_EVENT	Internal CMM Error
40	E_IMB_OPEN_DEVICE_FAILED	Internal CMM Error
41	E_IMB_MMAP_FAILED	Internal CMM Error
42	E_IMB_MUNMAP_FAILED	Internal CMM Error
43	E_IMB_RESP_LEN_ERROR	Invalid IPMI response. Blade may be returning invalid data.
44	E_NEM_SNMPTRAP_ERROR	Error setting snmptrap parameters. Retry command
45	E_NEM_SYSTEMHEALTH_ERROR	Internal CMM Error
46	E_NEM_GETHEALTH_ERROR	Internal CMM Error
47	E_NEM_SNMPENABLE_ERROR	Internal CMM Error
48	E_NEM_SENSOR_HEALTH_ERROR	Internal CMM Error
49	E_NEM_FILTER_SEL_ERROR	Internal CMM Error
50	E_NEM_INITIALIZE_ERROR	Internal CMM Error
51	E_NEM_SENSOR_EVENT	Internal CMM Error
52	E_NEM_SENSOR_ERROR	Internal CMM Error
53	E_NEM_SNMP_PROCESS_EVENT_ERROR	Internal CMM Error
54	E_NEM_SNMP_DEST_ADDR_ERROR	SNMP Trap address that the user is setting is invalid
55	E_NEM_SNMP_COMMUNITY_STRING_ERROR	SNMP Community that user is setting is invalid
56	E_NEM_SNMP_TRAP_VERSION_ERROR	SNMP Trap version that the user is setting is invalid
57	E_NEM_SNMP_TRAP_PORT_ERROR	SNMP Trap port that the user is setting is invalid
58	E_NEM_SNMP_CFG_ERROR	Cannot read SNMP config parameter. Config file may be corrupted.
59	E_NEM_SEND_SNMP_TRAP_ERROR	Internal CMM Error
60	E_SFS_INVALID_TRANSACTION	Internal CMM Error
61	E_SFS_LOCK_SDR	Can't read SDRs. Blade may be busy, try again.
62	E_SFS_ENTITY_ID	Internal CMM Error
63	E_SFS_DEVICE_LOCATOR_NULL	Internal CMM Error

Table 45. Error and Return Codes for the RPC Interface (Sheet 3 of 4)

Code #:	Error Code String	Error Code Description
64	E_SFS_NO_MEMORY	Internal CMM Error
65	E_SFS_UNSUPPORTED_DEVICE	Internal CMM Error
66	E_SFS_RESPONSE_LENGTH	Internal CMM Error
67	E_SFS_RESPONSE_DATA	Internal CMM Error
68	E_SFS_POWER_SUPPLY_FRU	Internal CMM Error
69	E_SFS_PATTERN_FOUND	Internal CMM Error
70	E_SFS_SEMAPHORE_FAILED	Internal CMM Error
71	E_SFS_CALLBACK_NOT_FOUND	Internal CMM Error
72	E_SFS_END_OF_DATA	Internal CMM Error
73	E_SFS_NO_SEL_ENTRY	Internal CMM Error
74	E_SHEM_INTERNAL_ERROR	Internal CMM Error
75	E_SHEM_INVALID_DATA_ITEM	Not a valid -d parameter
76	E_SHEM_STANDBY_CMM	Can't execute this command on the standby CMM
77	E_SNSR_STATUS_UNSUPPORTED	Internal CMM Error
78	E_SNSR_UNSUPPORTED	Internal CMM Error
79	E_SNSR_CATEGORY	Internal CMM Error
80	E_SNSR_NO_MEMORY	Internal CMM Error
81	E_SNSR_NOT_FOUND	Internal CMM Error
82	E_SNSR_ACTION_UNSUPPORTED	Internal CMM Error
83	E_SNSR_NON_FIRMWARE	Internal CMM Error
84	E_SNSR_SHARE_CODE	Internal CMM Error
85	E_SNSR_LOW_STORAGE	Internal CMM Error
86	E_SNSR_EVENT_TYPE	Internal CMM Error
87	E_SNSR_INVALID_REQUEST	Internal CMM Error
88	E_SNSR_OS_ERROR	Internal CMM Error
89	E_SNSR_PROCESSOR_NOT_PRESENT	Internal CMM Error
90	E_SNSR_THRESHOLD_UNSUPPORTED	The sensor being queried doesn't support a particular threshold
91	E_SNSR_CAPABILITY_UNSUPPORTED	Internal CMM Error
92	E_SNSR_SCANNING_DISABLED	Internal CMM Error
93	E_SNSR_MAX_RETRIES	Internal CMM Error
94	E_SNSR_TRIGGER_TYPE	Internal CMM Error
95	E_SNSR_STATE	Internal CMM Error
96	E_SNSR_EVENT_DEREGISTER	Internal CMM Error
97	E_SNSR_SEL_EVENT_FUNCTION	Internal CMM Error
98	E_SNSR_BASE_INDEX	Internal CMM Error
99	E_SNSR_PRESENCE_DETECTED	Internal CMM Error

Table 45. Error and Return Codes for the RPC Interface (Sheet 4 of 4)

Code #:	Error Code String	Error Code Description
100	E_SNMP_CMD_UNSUPPORTED	Internal CMM Error
101	E_SNMP_ERROR	Internal CMM Error
102	E_SNSR_VALUE_OUT_OF_RANGE	Internal CMM Error
103	E_SNSR_AUTH_ERROR	Internal CMM Error
104	E_WP_INITIALIZE_LIBS	Internal CMM Error
105	E_WP_CFG_READ_ERROR	CMM config file may be corrupted
106	E_WP_CFG_WRITE_ERROR	CMM config file may be corrupted
107	E_WP_THRESHOLD_UNSUPPORTED	The sensor being queried doesn't support a particular threshold.
108	E_WP_INVALID_TARGET	The sensor doesn't support a "current" value. This happens when querying a current value on a discrete sensor type.
109	E_WP_INVALID_LOCATION	Not a valid -l location
110	E_WP_INVALID_DATA_ITEM	Not a valid -d parameter
111	E_WP_INVALID_SET_DATA	Not a valid -v parameter
112	E_WP_CMD_UNSUPPORTED	Not a supported command
113	E_WP_STANDBY_CMM	Can't execute this command on the standby CMM.
114	E_WP_I2C_ERROR	Internal CMM Error
115	E_FT_SEM_GET_FAILURE	Internal CMM Error
116	E_DRONE_NOT_FOUND	Internal CMM Error
117	E_INTERNAL_ERROR	Internal CMM Error
118	E_BPM_PWR_SUPPLY_NOT_PRESENT	Power Supply Not Present Error
119	E_NEM_INTERNAL_FAILURE	Internal CMM Error
120	E_WP_CMM_RESET	Internal CMM Error
121	E_UPDATE_INPROGRESS	Update in Progress
122	E_MSGQ_START	Initilization message

Note: Error codes 100 and 101 will never be returned by the CMM when using RPC. If you receive these error codes, they are most likely being returned by the RPC client. Consult your RPC client documentation or code to find the description of these errors.

20.2.3 ChassisManagementApi() Threshold Response Format

The following table documents the format of ChassisManagementApi() queries that return data of type DATA_TYPE_ALL_THRESHOLDS.

Table 46. Threshold Response Formats

Dataitem	Return format	Example
thresholdsall	Data is returned in the THRESHOLDS_ALL structure as defined in cli_client.h. All structure fields are valid. If a particular threshold is not supported, the structure field will contain an empty string. Each supported and valid field is a null-terminated string. **Syntax:** [*Value*] [*Units*] /n /0	5.400 Volts 5.200 Volts 5.100 Volts 4.600 Volts 4.800 Volts 4.900 Volts
uppernonrecoverable uppercritical uppernoncritical lowernonrecoverable lowercritical lowernoncritical	Data is returned in the THRESHOLDS_ALL structure defined in cli_client.h. Only the structure field corresponding to the dataitem requested is valid. If a particular threshold is not supported, the structure field will contain an empty string. A valid field is a null-terminated string. **Syntax:** [*Value*] [*Units*] /n /0	5.160 Volts

20.2.4 ChassisManagementApi() String Response Format

The following table documents the format of ChassisManagementApi() queries that return data of type DATA_TYPE_STRING.

Table 47. String Response Formats (Sheet 1 of 4)

Dataitem	Return format	Example
current	Null-terminated string showing the current value of a sensor. **Syntax:** Value [Units] /0	23.000 Celsius
Ethernet	Null-terminated string showing the orientation of the eth0 Ethernet port: **Syntax:** [*front/back*] /0	front
fanspeed	Null-terminated string giving the current CMM fan speed setting: **Syntax:** [*0 or 80 or 100*] /0	80

Table 47. String Response Formats (Sheet 2 of 4)

Dataitem	Return format	Example
healthevents	List of human-readable health events. Lines are separated by line feeds with a null-terminator at the end. "(null)" or "" if there are no healthevents Refer to Section 11.0, "Health Event Strings" on page 67 **Syntax:** [*Critical/Major/Minor*] Event: [*Health String*] /n /0	Major Event: Fan Tray 3 Pres Fan Tray Removed Major Event: FANTK9 Lower critical going low asserted Major Event: FANTK8 Lower critical going low asserted Major Event: FANTK7 Lower critical going low asserted Minor Event: BP +12V Upper non-critical going high asserted
ListDataItems	List of available dataitems. Lines are separated by line feeds and a null-terminator at the end. **Syntax:** [*Dataitem*] /n /0	presence listtargets listdataitems health healthevents sel snmpenable snmptrapcommunity snmptrapaddress1 snmptrapaddress2 snmptrapaddress3 snmptrapaddress4 snmptrapaddress5 redundancy powerstate
ListTargets	List of available targets. Targets represent the sensor data records (SDRs) for a particular component. Lines are separated by line feeds with a null-terminator at the end. **Syntax:** [*Sensor Name*] /n /0	Brd Temp +1.5 V +2.5 V +3.3 V +5 V
location	Null-terminated string containing the user-specified physical location of the CMM, 16 characters maximum. **Syntax:** [*Location String*] /0	Server room 3
powerstate	Human readable powerstate information containing the target blade powerstate information. Lines are separated by line feeds with a null-terminator at the end. **Syntax:** Board is [*present or not present*] /n Board HEALTHY# signal is [*asserted or not asserted* /n Board has [*not*] been powered up by the CMM /n Board is in [*active or offline*] mode. /n /0	Board is present Board HEALTHY# signal is asserted Board has been powered up by the CMM Board is in active mode.

Table 47. String Response Formats (Sheet 3 of 4)

Dataitem	Return format	Example
redundancy	Human-readable redundancy information containing the current CMM redundancy status. Lines are separated by line feeds with a null-terminator at the end. **Syntax:** Top CMM (CMM 1): [*Present or Not Present*] ([*active or standby*]) [** or no star*] /n Bottom CMM (CMM 2): [*Present or Not Present*] ([*active or standby*) [** or no star*] /n * = The CMM you are currently logged into. /n /0	Top CMM (CMM 1): Present (active) * Bottom CMM (CMM 2): Not Present (standby) * = The CMM you are currently logged into.
slotinfo	Human-readable slot information, containing a list of System slots, Peripheral slots, Busless slots, and Occupied slots. If there are no slots in a particular category, "None" is reported. Lines are separated by line feeds with a null-terminator at the end. Each colon is followed by one tab (for Peripheral and Busless slots) or two tabs (for System and Occupied slots) and a space-delimited list of slot numbers. **Syntax:** System Slot(s): [*None or slot numbers*] /n Peripheral Slot(s): [*None or slot numbers*] /n Busless (Switch) Slot(s): [*None or slot numbers*] /n Occupied Slot(s): [*None or slot numbers*] /n /0	System Slot(s): None Peripheral Slot(s): 2 3 4 5 6 7 8 13 14 15 16 17 18 19 20 21 Busless (Switch) Slot(s): 2 19 20 21 Occupied Slot(s): 2 5 21
snmptrapaddress[*1..5*]	Null-terminated string containing a dotted-quad IP address **Syntax:** XXX.XXX.XXX.XXX /0	10.10.240.81
snmptrapcommunity	Null-terminated string containing the snmptrapcommunity name **Syntax:** [*SNMP Trap Community Name String*] /0	publiccmm
snmptrapport	Null-terminated string showing the SNMP trap port. **Syntax:** [*port number*] /0	161
snmptrapversion	Null-terminated string showing the version of SNMP traps the CMM is currently set for. **Syntax:** **[v1 *or* v3] /0**	v3

Table 47. String Response Formats (Sheet 4 of 4)

Dataitem	Return format	Example
unhealthylocations	Human-readable unhealthy blade, cmm and/or chassis information, containing a list of blades, cmms and/or chassis with a health status of Critical, Major, and Minor. If there are no items in a particular category, "None" is reported. Lines are separated by line feeds with a null-terminator at the end. Each colon is followed by one space and a space-delimited list of blade numbers and the word "chassis" for chassis health conditions. **Syntax:** Critical: [*None or Blade# and/or chassis and/or cmm*] /n Major: [*None or Blade# and/or chassis and/or cmm*] /n Minor: [*None or Blade# and/or chassis and/or cmm*] /n /0	Critical: None Major: None Minor: 8 chassis
version	Null-terminated string containing the version of the CMM firmware. **Syntax:** **VXX.XXX /0**	V03.00a

20.2.5 ChassisManagementApi() Integer Response Format

The following table documents the format of ChassisManagementApi() queries that return data of type DATA_TYPE_INT.

Table 48. Integer Response Formats (Sheet 1 of 2)

Dataitem	Return format	Example
alarmcutoff	Integer value indicating alarm cutoff status: 0 = alarm not cut off 1 = alarm cut off	0
alarmtimeout	Integer value indicating the alarm cutoff timeout in minutes.	5
health	Integer value corresponding to the health of the location queried: 0 = OK 1 = Minor 2 = Major 3 = Critical	2
presence	Integer value corresponding to the absence or presence of the location queried: 0 = Not present 1 = Present **NOTE:** If a blade is not present, ChassisManagementApi() returns E_BLADE_NOT_PRESENT.	1

Table 48. Integer Response Formats (Sheet 2 of 2)

Dataitem	Return format	Example
psinhibit	Integer returned is a 1 or 0. 1 = Power supply is not inhibited 0 = Power supply is inhibited	1
snmpenable	Integer value indicating SNMP status: 0 = disabled 1 = enabled	0

20.2.6 FRU String Response Format

Querying an individual FRU field returns a null terminated string with a single line feed.

Querying individual FRU areas returns a null terminated string with the following:

Board Area: 3 Linefeeds

Product Area: 2 Linefeeds

Chassis Area: 2 Linefeeds

Querying the entire FRU (all) will return all of the above areas present in the FRU with the line feeds listed above.

Table 49. FRU Data Items String Response Format (Sheet 1 of 2)

Data Item	Description
all	Null-terminated string containing all FRU information for the location
boardall	Null-terminated string containing all board area FRU information for the location
boarddescription	Null-terminated string containing the description field in the FRU board area for the location
boardmanufacturer	Null-terminated string containing the manufacturer field in the FRU board area for the location
boardpartnumber	Null-terminated string containing the part number field in the FRU board area for the location
boardserialnumber	Null-terminated string containing the serial number field in the FRU board area for the location
boardmanufacturedatetime	Null-terminated string containing the manufacture date and time field in the FRU board area for the location
productall	Null-terminated string containing the product area FRU information for the location
productdescription	Null-terminated string containing the description field in the FRU product area for the location
productmanufacturer	Null-terminated string containing the manufacturer field in the FRU product area for the location
productmodel	Null-terminated string containing the model field in the FRU product area for the location
productpartnumber	Null-terminated string containing the part number field in the FRU product area for the location

Table 49. FRU Data Items String Response Format (Sheet 2 of 2)

Data Item	Description
productserialnumber	Null-terminated string containing the serial number field in the FRU product area for the location
productrevision	Null-terminated string containing the revision field in the FRU product area for the location
productmanufacturedatetime	Null-terminated string containing the manufacturer date and time field in the FRU product area for the location
chassisall	Null-terminated string containing all chassis area FRU information for the location
chassispartnumber	Null-terminated string containing the part number field in the FRU chassis area for the location
chassisserialnumber	Null-terminated string containing the serial number field in the FRU chassis area for the location
chassislocation	Null-terminated string containing the location field in the FRU chassis area for the location
chassistype	Null-terminated string containing the type field in the FRU chassis area for the location.
listdataitems	Null-terminated string containing the list of all of the FRU dataitems that can be queried for the FRU target

20.3 RPC Sample Code

Sample code for interfacing with the CMM through RPC is available in the file cli_client.c. The sample code compiles into a command-line executable for use with Linux or a .o file for use with VxWorks. To select the appropriate target, remove the comment from the appropriate #define in the source code.

The sample code first authenticates to the CMM through GetAuthCapability(). When the authentication is successful, the user's command-line arguments (for Linux) or calling parameters (for VxWorks) are passed to the CMM through ChassisManagementApi(). The return code is then checked and the result is printed to the console.

20.4 RPC Usage Examples

Table 50 presents examples of using RPC calls to get and set fields on the CMM. Data returned by RPC calls are returned in the *ppvbuffer* and *uReturnType* parameters to ChassisManagementApi().

Table 50. RPC Usage Examples (Sheet 1 of 3)

Example	ChassisManagementApi() [in] parameters	ChassisManagementApi() [out] parameters
Get the chassis temperature.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: Chassis *pszTarget*: TempSensorName *pszDataItem*: current	*uReturnType*: DATA_TYPE_STRING *ppvbuffer*: A null-terminated string of the format: Value [Units]
Get the fan tray presence.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: fantray1..3 *pszTarget*: NA *pszDataItem*: presence	*uReturnType*: DATA_TYPE_INT *ppvbuffer*: Integer value indicating presence 1 = Present 0 = Not Present
Get the CPU temperature of blade 5.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: blade5 *pszTarget*: CPUTempSensorName *pszDataItem*: current	*uReturnType*: DATA_TYPE_STRING *ppvbuffer*: A null-terminated string of the format: Value [Units]
Determine if a certain blade is present.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: blade[1-n] *pszDataItem*: presence	*uReturnType*: DATA_TYPE_INT *ppvbuffer*: 1 = Present The call to ChassisManagementApi() returns E_BLADE_NOT_PRESENT if the selected blade is not present.
Get all thresholds for the +3.3 V sensor on blade 2.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: blade2 *pszTarget*: 3.3vSensorName *pszDataItem*: ThresholdsAll	*uReturnType*: DATA_TYPE_ALL_THRESHOLDS *ppvbuffer*: A THRESHOLDS_ALL structure as defined in cli_client.h
Get the overall system health.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: system *pszDataItem*: health	*uReturnType*: DATA_TYPE_INT *ppvbuffer*: Integer value denoting health state 0 = OK 1 = Minor 2 = Major 3 = Critical
Get a list of blades with problems.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: system *pszDataItem*: unhealthylocations	*uReturnType*: DATA_TYPE_STRING *ppvbuffer*: List of all blades with problems
Get the temp1 sensor's health on blade 5.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: blade5 *pszTarget*: Temp1SensorName *pszDataItem*: health	*uReturnType*: DATA_TYPE_INT *ppvbuffer*: Integer value denoting health state 0 = OK 1 = Minor 2 = Major 3 = Critical

Table 50. RPC Usage Examples (Sheet 2 of 3)

Example	ChassisManagementApi() [in] parameters	ChassisManagementApi() [out] parameters
Get the CMM's overall health.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: CMM *pszDataItem*: health	*uReturnType*: DATA_TYPE_INT *ppvbuffer*: Integer value denoting health state 0 = OK 1 = Minor 2 = Major 3 = Critical
Get a blade's overall health.	*pszCMMHost*: localhost *uCmdCode* CMD_GET *pszLocation*: blade[1..n] *pszDataItem*: health	*uReturnType*: DATA_TYPE_INT *ppvbuffer*: Integer value denoting health state 0 = OK 1 = Minor 2 = Major 3 = Critical
Get the version of software on the CMM.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: CMM *pszDataItem*: version	*uReturnType*: DATA_TYPE_STRING *ppvbuffer*: A human-readable null-terminated version string.
Power off one of the blades.	*pszCMMHost*: localhost *uCmdCode*: CMD_SET *pszLocation*: blade[1-19] *pszDataItem*: powerstate *pszSetData*: poweroff	*uReturnType*: not used *ppvbuffer*: not used The return code from ChassisManagementApi() indicates success or failure.
Power on one of the blades.	*pszCMMHost*: localhost *uCmdCode*: CMD_SET *pszLocation*: blade[1-19] *pszDataItem*: powerstate *pszSetData*: poweron	*uReturnType*: not used *ppvbuffer*: not used The return code from ChassisManagementApi() indicates success or failure.
Reset a blade.	*pszCMMHost*: localhost *uCmdCode*: CMD_SET *pszLocation*: blade[1-19] *pszDataItem*: powerstate *pszSetData*: reset	*uReturnType*: not used *ppvbuffer*: not used The return code from ChassisManagementApi() indicates success or failure.
Determine what sensors are on blade 3.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: blade3 *pszDataItem*: ListTargets	*uReturnType*: DATA_TYPE_STRING *ppvbuffer*: A list of sensor names as defined in the SDR.
Determine what may be queried or set on a blade.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: blade3 *pszDataItem*: ListDataItems	*uReturnType*: DATA_TYPE_STRING *ppvbuffer:* A list of commands to be used as data items.
Determine what may be queried on the blade4 +3.3 V sensor.	*pszCMMHost*: localhost *uCmdCode*: CMD_GET *pszLocation*: blade4 *pszTarget*: +3.3SensorName *pszDataItem*: ListDataItems	*uReturnType*: DATA_TYPE_STRING *ppvbuffer*: A list of commands to be used as data items.

Table 50. RPC Usage Examples (Sheet 3 of 3)

Example	ChassisManagementApi() [in] parameters	ChassisManagementApi() [out] parameters
Enable the SNMP Traps.	*pszCMMHost*: localhost *uCmdCode*: CMD_SET *pszLocation*: chassis *pszDataItem*: SNMPEnable *pszSetData*: enable	*uReturnType*: not used *ppvbuffer*: not used The return code from ChassisManagementApi() indicates success or failure.
Set the SNMP Target.	*pszCMMHost*: localhost *uCmdCode*: CMD_SET *pszLocation*: chassis *pszDataItem*: SNMPTrapAddress[1-5] *pszSetData*: 134.134.100.34	*uReturnType*: not used *ppvbuffer*: not used The return code from ChassisManagementApi() indicates success or failure.
Set the SNMP Community.	*pszCMMHost*: localhost *uCmdCode*: CMD_SET *pszLocation*: chassis *pszDataItem*: SNMPCommunity *pszSetData*: public	*uReturnType*: not used *ppvbuffer*: not used The return code from ChassisManagementApi() indicates success or failure.
Set the Telco Alarm on.	*pszCMMHost*: localhost *uCmdCode*: CMD_SET *pszLocation*: CMM *pszDataItem*: TelcoAlarm *pszSetData*: 1	*uReturnType*: not used *ppvbuffer*: not used The return code from ChassisManagementApi() indicates success or failure.
Light Major LED on the CMM.	*pszCMMHost*: localhost *uCmdCode*: CMD_SET *pszLocation*: CMM *pszDataItem*: MajorLED *pszSetData*: 1	*uReturnType*: not used *ppvbuffer*: not used The return code from ChassisManagementApi() indicates success or failure.

21.0 Command Logging

All **CMMSET** commands from all of the CMM interfaces (CLI, RPC, and SNMP) are logged by the CMM under */var/log*. The size of the log file is 8 kbytes. When the command log becomes full, the log file is compressed and archived using gzip, then stored in */home/log*. The filenames for the log files will be *user.log.N.gz*, where N is the number of the log file from 1 to 4. The CMM will keep up to four archives of log files. If the log file becomes full and there are already four existing command log archives, the oldest archive will be deleted to make room for the newest archive.

Note: Archived files should **NEVER** be decompressed on the CMM as the resulting prolonged flash file writing could disrupt normal CMM operation and behavior. The files should be transferred and decompressed on a different machine. Files can be decompressed by any application that supports the decompression of gzip (.gz) file types.

Note: The /home/log directory should not be deleted or change. the CMM requires that directory exist to log errors.

22.0 Telco Alarm Sensors

The faceplate of the CMM contains a µDB15 Telco alarm connector. See Section 2.6, “Face Plate” on page 20 and Section 2.16, “Telco Alarm Signal” on page 34 for more information on the Telco Alarm connector. By default, pins 1 through 4 are used by the CMM to clear major or minor alarms. The current state of pins 1 and 3 inputs are mapped to a GPIO on the CMM and can therefore be reconfigured as discrete sensor inputs.

22.1 Configuring the Telco Alarm Connector Pins

To configure the telco alarm sensors as discrete inputs or to clear events, the following CLI command is used:

```
cmmset -l cmm -d TelcoInputs -v [sensors or clearEvents]
```

Where the value (-v) parameter is either:

sensors - Configures the telco alarm inputs as discrete input sensors. Or

clearEvents - Configures the telco alarm inputs to be used to clear telco alarms.

22.2 Obtaining the Configuration of the Telco Alarm Sensor Connector Pins

To retrieve the configuration of the Telco Alarm Sensor pins, use the following CLI command:

```
cmmget -l cmm -d TelcoInputs
```

This command will return a value of:

“sensors” - If the telco alarm input pins are being used as discrete sensor inputs.

“clearEvents” - If the telco alarm input pins are being used to clear minor and major alarms.

22.3 Telco Alarm Connector Sensor Naming

The default name of the two telco alarm sensors is *TAsensor1* and *TAsensor2*. These names are used and reported by the various CMM interfaces for the Telco Alarm Sensors. The sensors can be renamed using the following CLI command:

```
cmmset -l cmm -t TAsensorN -d sensorname -v [NewSensorName]
```

Where:

N - Is either “1” or “2” for TAsensor1 and TAsensor2, respectively.

NewSensorName - The new name that will be associated with the corresponding sensor.

22.3.1 Sensor Names using SNMP

The telco alarm sensor names that will be shown via SNMP will be those that are defined at CMM startup. If a telco alarm sensor name has been changed after startup, the new name will not be reflected via SNMP until the CMM is restarted.

22.4 Telco Alarm Sensors and Redundancy

In dual CMM systems, the telco alarm sensors are individual to each CMM in the system, and are not synchronized between the two CMMs. SEL entries from sensors on the active CMM will not be synched to the standby CMM.

If it is desired to monitor the same two inputs on both CMMs, the two inputs must be split into four and connected to the telco alarm connectors on both CMMs.

Telco alarm sensor names are synched between the two CMMs. Changing the name of the sensors on the active CMM will cause the same names to be assigned on the standby CMM.

23.0 Application Hosting

The ZT 7102 allows applications to be hosted locally. This is useful as a method for adding small custom management utilities to the CMM.

23.1 System Details

The CMM runs a customized version of embedded BlueCat* Linux* 4.0 on an Intel® XScale® microarchitecture 80200 processor. Development support for BlueCat Linux is available at:

http://www.lynuxworks.com

23.2 Startup and Shutdown Scripts

The CMM is capable of running user created scripts automatically on boot up or shutdown.This can be accomplished by editing the /etc/rc.d/init.d/userScripts file with a text editor. This is a standard shell file just like /etc/rc.d/init.d/cmm. By default it just echos "Starting user services" and "Shutting down user services". The user can add scripts or what they want to the file. The Ramdisk version of /etc/inittab calls "/etc/rc.d/init.d/userScripts start" right after its starts the cmm software. When you type reboot, it calls "/etc/rc.d/init.d/userScripts stop”.

23.3 System Resources Available to User Applications

Since the CMM has firmware of its own running at all times, user applications must adhere to certain resource and directory constraints to avoid disrupting the CMMs operation. Specifically, restrictions are placed on an application’s consumption of file system storage space, RAM, and interrupts. Exceeding these guidelines may interfere with proper operation of the CMM.

23.3.1 File System Storage Constraints

The following file system locations are available for storage by user applications. Storing files in locations other than those explicitly listed here is not supported.

23.3.1.1 Flash Storage Locations

Applications should not perform excessive amount of flash file I/O at runtime, because this will impair performance of the CMM.

/etc - Useful for storing configuration files. Do not add more than 100 Kbytes to /etc.

/etc/rc.d/init.d - Useful for storing startup scripts. Do not add more than 100 Kbytes to /etc.

/home/scripts - Useful for storing user scripts. Do not add more than 3 Mbytes to /home.

/home - Useful for storing application binaries. Do not add more than 3 Mbytes to /home.

23.3.1.2 RAM-Disk Storage Locations

Caution: Files in this location are stored in RAM, and will be lost during CMM reboots.

Due to the constraints of writing to flash memory, larger file operations such as decompressing an archive should be performed on RAM-disk in the following directory:

/usr/local/cmm/temp - Useful for storing temporary files. Applications should make a subdirectory for use with their temporary files. Do not add more than 5 Mbytes to this location.

23.3.2 RAM Constraints

Up to 32 Mbytes of RAM are available for use by user applications. RAM usage at runtime, on average, should not exceed this quantity.

23.3.3 Interrupt Constraints

User applications should not make use of interrupts. All interrupts are reserved for use by the CMM firmware.

23.4 Ram Disk Directory Structure

The following directories are stored on a RAM-disk. These directories store information critical to the CMM and are size constrained. These directories should not be modified in any way.

/usr – Executables including the CMM executables (/usr/local/cmm/bin). Also contains some libraries and temp files.

/lib - Shared library files and drivers

/bin – Linux binary files like ping, mount

/dev – Device driver files

/proc – Special linux area that contains info about memory, drivers, etc.

/root – Empty root user area.

/sbin – Binary files like ifconfig, route

/tmp – Empty

/var – Temp location for log files and other temp files.

24.0 Busless Backplane Support

The ZT 7102 supports busless PICMG* 2.16 backplanes. Busless backplanes are PICMG 2.16 backplanes that do not have the CompactPCI* bus routed or enabled. When the CMM is inserted into a busless backplane, the ZT7102 does not perform a check for system or peripheral slots, and powers up the boards in any slot regardless. Boards used in this type of chassis should be system master, or drone mode enabled boards. Special care should be taken to not use peripheral only boards in such a chassis as these boards rely on a system master and CompactPCI bus to operate properly.

25.0 Updating Software

When new CMM updates are available, they are packaged in a zip file and posted to the Intel web site located at:

http://www.intel.com/design/network/products/cbp/zt7102.htm

Please follow the instructions provided with the software update package to perform the update

The CMM is capable of having its firmware and critical system files updated when new update packages become available. The update process allows these updates to occur over the wire without losing the active cmm in a redundant configuration.

25.1 Key Features of the Firmware Update Process

- Updates can be done remotely over the front or back Ethernet ports on the CMM
- Current CMM configuration data is preserved across the update
- Critical CMM data such as the SEL and command history is preserved across an update
- Redundant CMMs can be updated without interrupting management of the chassis
- Update files are verified and checked for corruption
- Update components have associated version numbers
- Update events are logged to the SEL
- Updates can be triggered using the CLI
- Update packages can be located locally on the CMM, or pulled from a mounted NFS or remote FTP server

25.2 Update Process Architecture

The update process consists of the following components:

- User Client - The client is used to trigger the update process, and can be located anywhere on the network.
- Update Package - The update package contains the new software components and other files necessary for the update. The update package can be pulled from a remote server or pushed locally onto the CMM.
- CMM Update Request Handler - The update request handler is CMM software that processes incoming update requests from and responses to the various interfaces provided by the CMM.
- CMM Update Script - The update script is stored on the CMM and is used to process the contents of the update package and perform the update.
- Update Package Server (Optional) - The update package server can be used to store update packages remotely from the CMM. This can be an FTP or NFS server.

25.3 Critical Software Update Files and Directories

The following table is a list of files and directories important to the software update process.

Table 51. List of Critical Software Update Files and Directories

File or Directory Name:	Description:
/usr/local/cmm/temp	Mount point for ramdisk
/usr/local/cmm/temp/update/package	Directory into which the update package is copied and unzipped. The update process will delete and recreate this directory. If it is desired to copy the update package to the CMM before updating, then put the package in /usr/local/cmm/temp/update. **NOTE:** Do NOT place the update files into the /usr/local/cmm/temp/update/package directory. The update process uses this directory and will erase and overwrite files placed here.
/usr/local/cmm/temp/update/etc	Backup copy of /etc is located here
[package file].zip	Zip file containing update package files
[package file].md5	MD5 checksum of the [package file].zip file
/etc/versions	At the end of an update, this file contains a list of the package version, and each component version
/etc/versions.<version>	The update process will move the "old" /etc/versions file by appending a file extension to it which is the package version string - i.e., .. /etc/versions.V00.30a
/home/update/scripts/K*	These optional scripts are supplied by the user. The update process will execute them just prior to shutting down the cmm applications. The primary function of these scripts is to allow the user a method of shutting down any user process which are using the flash file systems.
/home/update/scripts/S*	These optional scripts are supplied by the user. The update process will execute these scripts just before processing the save list - which copies files and directories from /usr/local/cmm/temp/update/etc to /etc
/home/update/backup	If the update direction is "backward", then files and directories that are priority 2 will be copied here instead of /etc
/usr/local/cmm/temp/update/update.log	Log file to which output is sent while the update process is operating - this is on a ramdisk.
/etc/cmm/update.log	The log file is copied when the update process completes successfully

25.4 Update Package

The update package will be distributed by Intel and may contain the following components:

Table 52. Contents of the Update Package (Sheet 1 of 2)

File Name	Description
CMM_RB.bin	Redboot image to be stored in flash
CMM_FPGA.bin	FPGA image
CMM_FFS.bin	/etc image

Table 52. Contents of the Update Package (Sheet 2 of 2)

File Name	Description
CMM_OS.bin	OS image
README	Text file containing release notes for the update package
Update_Metadata	File containing info on the update package and how it should be installed on the CMM
Utilities	Sub-directory containing any utilities that might be required for the install
validationfile	File containing MD5 check sums for the files and scripts present in the update package
saveList	List of configuration files in /etc to be preserved across software updates
Other	Other components such as FRU files that may be necessary for the update
version_history	List of all versions of firmware available for the platform in sequential order, with the newest versions at the end of the list.
ZT7102.MIB	MIB file for SNMP.

The update package can be placed locally on the CMM in the /usr/local/cmm/temp directory, or it can reside on a server on the network.

The files and directories that make up the Update Package are packaged and delivered as a zip file. Associated with the zip file is a file containing the md5 sum of the zip file. For example:

[*PackageFileName]*.zip file containing all files and subdirectories of the update package

[*PackageFileName]*.md5 file containing md5 checksum of PackageFileName.zip

Arguments for the location of the update package will be given in the CLI command. It is here that you can point to a remote server, or to a local directory. The update script will then remove the /usr/local/cmm/temp/update/package directory and recreate it with the new package files.

Note: If an NFS server is mounted to the CMM, the argument in the update script will be similar to a file located locally on the CMM.

If the package fails to copy or transfer to /usr/local/cmm/temp/update/package, then the update process will terminate.

25.4.1 Update Package File Validation

The first level of update package validation is checking that the md5sum of the zip file matches the md5sum in the [*PackageFileName*].md5 file of the update package.

Update packages are validated during the update process by computing the md5sum of each file and comparing against the check sums stored in the check sums stored in the validation_file included with the update package.

The update package is also checked to ensure it is valid for the platform that it is being used on. For example, an update package with firmware for a CPCI CMM will not work on an ATCA CMM.

25.4.2 Update Firmware Package Version

The firmware update package has a firmware version associated with the entire package. This firmware version is the same version that can be retrieved using the CLI command:

```
cmmget -l cmm -d version
```

To determine if the update is a new, old or same version, the update package will contain a version_history file which contains a list of all software builds that have occurred, listed in sequential order. Newer builds are at the bottom of the list.

25.4.3 Component Versioning

The components contained in the update package will contain versioning information which will allow the update process to ensure the updated components installed on the CMM are all compatible at the end of an update and match those contained in the update package.

If an update package contains a component version that is already installed on the CMM, then that component will not be updated. A method is also provided to force all components to update regardless of the version installed.

Version info for files currently installed on the CMM will be able to be obtained through the various interfaces and is stored in the /etc/cmm/versions file. Version info for the files in the update package will also be made available.

The CMM checks the md5sum of /etc/versions during boot. In the event that the checksum fails, the CMM applications will fail to start and a message will appear on the console as follows "Startup of CMM applications has terminated due to detection of an inconsistent software load." This functionality exists to prevent a CMM with an incorrect /etc/versions file, which may have occurred during an incomplete update, from synchronizing incorrectly with the other CMM.

25.5 saveList and Data Preservation

The update process will preserve user configuration data as well as critical system configuration information across updates. These files are located in the /etc directory. The list of files or directories to be preserved across updates is contained in the software update package and is called SaveList. Each entry in the saveList will contain a directory or file to be saved as well as a priority assigned to it. Priority for the file can be either a 1 or a 2, and is used to determine if certain files or directories should be saved when updating to an earlier firmware version.

Priority 1 is assigned to files or directories that should be saved in all cases of an update, including going forward or backward in firmware version. Priority 1 files are considered files critical to the CMM operation and access to the CMM, such as Ethernet configuration.

Priority 2 is assigned to files or directories which should not be saved if an update is being performed to an earlier version of firmware.

During an update, the CMM will copy over the current files in /etc to ram disk in /usr/local/cmm/temp/update/etc. The CMM then uses the saveList file to determine which configuration files or directories to copy back into the new /etc partition. When updating to a previous version of firmware, the CMM makes a backup of any files designated priority 2 in the save list to flash in the /home/update/backup/etc directory.

Table 53. saveList Items and Their Priorities (Sheet 1 of 2)

File	Priority
/etc/*.cfg	2
/etc/*.ini	2
/etc/cmm/*.ent	2

Table 53. saveList Items and Their Priorities (Sheet 2 of 2)

File	Priority
/etc/passwd	1
/etc/shadow	1
/etc/cmm/sel_*	1
/etc/cmm/cmm_sel	1
/etc/snmp*.conf	1
/etc/ifcfg-eth*	1
/etc/HOSTNAME	1
/etc/ftp*	1
/etc/group	1
/etc/profile	1
/etc/versions*	1

25.6 Update Mode

To perform the update, the CMM will enter an update mode to prevent interference from other processes that may write to the flash and to unmount the JFFS2 drives that may need to be repartitioned.

In the update mode, all CMM applications will be stopped. Any user defined processes that are running must also be stopped. Hooks in the update process will be provided to allow these processes to be stopped by calling user scripts located in /home/update/scripts/ if they exist. The update process will execute any scripts that match the following pattern: /home/update/scripts/K*. Scripts will be execute in alpha-numeric sort order. Getting a directory listing using the “ls” command will display the order the scripts will run. User scripts should follow the convention of returning a “0” if successful and a non-zero for failure. The update process will not fail if a script returns a failure, however, the update process will fail if the flash drives cannot be unmounted when necessary.

Before unmounting the JFFS2 drives (/etc and /home), /etc will be copied to ram disk /usr/local/cmm/temp/update/etc to retain user and system configuration information.

Because all CMM applications are stopped, systems running with redundant CMMs will lose redundancy during the firmware update process.

25.7 Update_Metadata File

The Update_Metadata file included in the update package is used by the update process to determine the platform, firmware package version, files, sequence, update method, location, and any other data required to update the individual components in the update package.

25.8 Synchronization

Following a firmware update to the standby CMM in a redundant CMM, some level of synchronization will occur. The firmware contains an internal synchronization version which is used to determine what level of synchronization occurs following an update. The internal synchronization number will increment each time a firmware package is released.

Following a firmware update, the CMM with the newest synchronization version will become or remain the active CMM. If the synchronization version is the same after an update, a full synchronization occurs.

Note: User scripts are only synchronized if the synch code versions remain the same.

The following table describes the behavior following an update to a different synch code version.

Table 54. Synchronization Behavior for Differing Synch Versions

	Local Synch Version > Other Synch Version	Local Synch Version < Other Synch Version
Local CMM is Active	Synch to Standby: Board Power States	Synch to Standby: Board Power States, SEL, SNMP trap addresses, Password **Note:** /home/scripts is NOT synchronized.
Local CMM is Standby	Synch from Active: Board Power States, SEL, SNMP trap addresses, Password **Note:** /home/scripts is NOT synchronized.	Synch from Active: Board Power States.

Board power states are synchronized in all cases (synch versions equal or not equal).

The synchronization process does not synchronize the alarms (SEL) when the synch versions are not equal. SEL files are saved by the update process. During an update to an earlier version of a redundant setup, alarms received while the Standby is being updated won't be synchronized.

25.8.1 Updating to a Newer Synchronization Version

Following an update on the standby CMM to firmware that has a newer synchronization version than the active CMM, the active CMM will synchronize critical information, such as board power states, the SEL, SNMP trap addresses and configuration, and password files. Once synchronization completes, the newly updated CMM will then become the active CMM because it has a newer synchronization version.

25.8.2 Updating to an Older Synchronization Version

Following an update on the standby CMM to firmware that has an older synchronization version than the active CMM, no synchronization will occur and the newly updated CMM will remain as standby because it has an older synchronization version.

25.8.3 Updating to the Same Synchronization Version

Following an update on the standby CMM to firmware that has the same synchronization version as the active CMM, a full synchronization will occur upon completion of the update (as if a new CMM has been inserted into the chassis). The newly updated CMM will remain as the standby CMM following synchronization.

25.9 Single CMM System

When updating CMMs in a system that only contains a single CMM, once the CMM enters the update mode, any components in that system the CMM manages power to may power down because the CMM applications are stopped. Consult your system and board documentation to determine the effect of updating or removing the CMM from the system.

25.10 Redundant CMM Systems

In systems with redundant CMMs, the update should be triggered on the standby CMM. An update cannot occur on the active CMM.

Once the CMM enters the update mode, the chassis will lose redundancy for the duration of the update process.

25.11 CLI Support

The CLI will support a command for an update request. The syntax of the command is as follows:

```
cmmset -l cmm -d update -v “[Update Package path and filename] [force]
[overwrite] [ftp:server:user:password]”
```

Where:

Update Package path and filename - The path and file name of the update package file without the .zip or .md5 extension. For example: “builds/build105b/CMM_P00.105b”

force - Optional argument that causes all components to update regardless of version. This option cause the environment variable FORCE_UPDATE to be set to TRUE.

overwrite - Optional argument that prevents data in the saveList from being preserved. Also, data restore user scripts will not be executed. This option cause the environment variable FORCE_OVERWRITE to be set to TRUE.

The following arguments are used if the update package is located on a remote FTP server. If *ftp* is supplied as an argument, *server* and *user* are also required. *Password* is optional. If *password* is not supplied, then ftp will prompt for a password.

ftp - Optional argument used to indicate that the update package resides on a remote ftp server. If this argument is given, the arguments for *server*, *user*, and *password* must also be given.

server - Optional argument that is the name or IP address of a remote ftp server containing the firmware update package.

user and password - Optional argument that is the username and password for the FTP server.

Note: The -v argument can be up to 64 characters long.

The command returns a 0 if the update request is successful, and non-zero if an error occurs.

25.12 Hooks for User Scripts

The update process provides hooks for user scripts in two places during the update process. User scripts should follow the convention of returning a "0" if successful and a non-zero for failure. The user scripts will be invoked as a forked sub-shell.

25.12.1 Update Mode User Scripts

Entering update mode requires that all JFFS2 drives be unmounted, which requires that all processes using these drives be stopped. Prior to entering update mode, the update process will call user scripts located in /home/update/scripts/ if they exist. The update process will execute any scripts that match the following pattern: /home/update/scripts/K*. No arguments are passed to these scripts. The update process will not fail if a script returns a failure, however, the update process will fail if the flash drives cannot be unmounted when necessary. A message indicating if the script was successful or failed will be output to the terminal. See Section 25.6, “Update Mode” on page 131 for more information on the Update Mode.

25.12.2 Data Restore User Scripts.

The update process will also execute user scripts following the update of the new /etc file system to flash. The update process will execute any user scripts that match the following pattern: /home/update/scripts/S*. Arguments will be passed that the scripts can use. These arguments are:

arg1 = Update Component Name: ["BlueCat", "etc_jffs", "RedBoot", "fpga"]

Note: For the user scripts, this argument will always be "etc_jffs".

arg2 = Update Direction: ["forward", "backward", "same"]

Note: "forward" means that the Update Package version is newer than the installed version. "same" means they are the same, and "backward" means the Update Package version is older then the installed version.

arg3 = Installed Package Version: i.e., "P00.105b". This is the version of the software that is currently installed on the cmm.

arg4 = Update Package Version: i.e., "P00.111a". This is the version of the update package software.

The update script will print a message to the terminal indicating whether the script was successful or failed. Currently, the failure of one of these scripts does not cause the overall update process to fail.

These scripts can be used to perform any user updates that need to coincide with the newly installed version.

Note: In prior versions of this document, the following examples showed how to use the user scripts to replace a script in the /etc/scripts directory. Going forward, all user scripts should be stored in /home/scripts. The directory /etc/scripts will be a link to /home/scripts. As a general rule, users should only put scripts and files under the /home directory.

25.12.3 Example Task - Replace /home/scripts/myScript with a newer version if updating forward or an older version if updating backward

Example update scenario:

- There are three CMM builds - call them A, B, and C. The CMM is currently installed with build B. Build A is older than B and C is newer.
- If the CMM is updated to build C, then the user script /home/scripts/myScript needs to be updated to a newer version.
- Similarly, if the CMM is reverted to build A, then the user script /home/scripts/myScript needs to be replaced with an older version.
- The old version of myScript is placed on the CMM as: /home/stagingarea/myScript.old
- The new version of myScript is placed on the CMM as: /home/stagingarea/myScript.new
- The Data Restore User Script written to manage /home/scripts/myScript is placed on the CMM as: /home/update/scripts/S10updateMyScript

A simple example of the S10updateMyScript could look like the following:

```
#!/bin/bash

direction=$2

if [ "$direction" = "forward" ] ; then
cp /home/stagingarea/myScript.new /home/scripts/myScript
elif [ "$direction" = "backward" ] ; then
cp /home/stagingarea/myScript.old /home/scripts/myScript
fi

exit $?
```

When the update process executes, and the Data Restore User Scripts are executed (during the update of the /etc partition), then the script S10updateMyScript will be executed and /home/script/myScript will be updated accordingly.

Variation on the above example:

- In this case, the different versions of myScript are put on the CMM (by the user) as:
  - /home/stagingarea/myScript.A
  - /home/stagingarea/myScript.B
  - /home/stagingarea/myScript.C

Note: If the CMM has build A installed, the "cmmget -d version" command will return the string "A" - and so on for builds B and C.

Another example of S10updateMyScript:

```
#!/bin/bash
```

```
newversion=$4
cp /home/stagingarea/myScript.$newversion /home/scripts/myScript
exit $?
```

25.13 Update Process

1. Client initiates an update request via CLI command
2. CMM validates the update request
   - CMM is not already doing an update
   - In a redundant configuration, the CMM must be standby
   - In a redundant configuration, redundancy must be "active"
   - CMM is not currently synchronizing
   - Enough space is present in ramdisk and flash for the update
3. If update request is valid then
   - Continue
4. Else
   - Exit
5. If FTP arguments supplied then
   - Retrieve package files (.zip, .md5) from FTP server
   - Exit if error occurs
6. Else
   - Copy package files (.zip, .md5) to package directory on ramdisk (/usr/local/cmm/temp/update/package)
   - Exit if error occurs
7. Validate that .zip file matches checksum in .md5 file
   - Exit if no match
8. Unzip the .zip file in the package directory on ramdisk
9. Validate that all files in the unzipped package match the md5 checksum in the validationFile.
   - Exit if any files fail
10. Validate that the package matches the CMM platform ("cpci" or "atca")
    - a.Exit if mismatch
11. Transition to Update Mode
    - Call any user supplied scripts in /home/update/scripts/K*
    - Shutdown CMM apps (/etc/rc.d/init.d/cmm stop)
    - Copy /etc to /usr/local/cmm/temp/update/etc
    - Unmount /etc and /home
    - Exit if any of these filesystems cannot be unmounted

12. Update Components (listed in UpdateMetadata file)
    - If a component update fails
      Stop updating components
      Attempt to remount /etc and /home
      Exit the update process, but do not reboot
    - If the component is "etc_jffs" then
      Remount /etc and /home
      Call any user supplied scripts in /home/update/scripts/S*
      Process saveList - copying files and directories from /usr/local/cmm/temp/update/etc to /etc
13. If the process has been successful so far then
    - Remount /etc and /home
    - Update the /etc/versions file - backing up the old one as "/etc/versions.<version>"
    - Copy the update.log file from ramdisk to /etc/cmm/update/update.log
    - Reboot CMM

25.14 Update Process Status and Logging

During the update process, status is sent to stdout as it executes. Output will also be appended to the file /usr/local/cmm/temp/update/update.log. This file will be copied to /etc/cmm/update.log at the completion of the update process.

Status output will be of the format:

MM/DD/YY HH:MM:SS [process[pid]]: [Message String]

25.15 DEBUG_UPDATE Variable

Setting the environment variable DEBUG_UPDATE prior to starting an update will print out additional debug level output to the user including all files that are copied during the update process. To set the DEBUG_UPDATE variable type the following in the CLI:

```
export DEBUG_UPDATE=1
```

25.16 Update Process Sensor and SEL Events

Once the CMM enters the update mode in which no CMM processes are active, events cannot be entered into the SEL directly. While in this mode, the CMM will enter events into the eeprom. Once the system reboots into the OS, the CMM will log the events from the eeprom into the SEL.

Note: Events that occur prior to entering the update mode are logged directly into the SEL.

25.17 RedBoot Update Process

The firmware can also be updated through Redboot. This update is done at a pre-OS level, meaning that the update is executed before the OS loads. This method requires updating over TFTP through the eth0 Ethernet port and must be done locally. A separate update package is needed if this method is used. The following instructions are included with the update package.

25.17.1 Required Setup

To update the flash on the CMM using the Redboot method, the following setup is required:

1. A host computer running Microsoft* Windows XP, 2000 or NT4.0 operating systems
2. The above computer should be configured with a static IP address of 192.168.100.20 and a subnet mask of 255.255.255.0
3. The above computer should be on the same subnet as the CMM or connected directly.
4. It is highly recommended that the subnet be isolated. The only two network entities on the subnet should be the CMM and the Windows computer.

25.17.2 Update Procedure

Read through the following steps completely before starting the update.

1. Extract the contents of the zip file into a folder (e.g. "C:\CMM\") on the host PC.
2. Open a DOS Command Line Prompt on the Windows PC and change directory to the above folder (e.g. C:\CMM)
3. To execute the update, the CMM needs to be reset using the reset button on the faceplate while simultaneously executing update_cmm.bat on the host with the following options:
   a. The MAC address of the connected interface on the CMM is the first command line parameter. To find this out, use the command *ifconfig eth0* on the CMM CLI to find out the MAC address of eth0.
   b. A second argument ,"FFS", must be used when upgrading/downgrading to a different version. Note that this will reset the passwords, IP addresses and other custom Linux environment settings to factory defaults and clear the event log entries.So far the command line on the host should look like this:

      ```
      update_cmm 00-a0-b7-d8-d8-5d FFS
      ```

   c. By default, the update script assumes that the host computer and the CMM are on the network 192.168.100.x. If your host computer is on a different network, provide another command line argument to the update_cmm batch file. This argument should be an unused IP address on the same network the host computer is on. For example:

      ```
      update_cmm 00-a0-b7-d8-d8-5d FFS 192.168.100.50
      ```

After executing the batch file, follow the instructions as prompted by the batch file for a successful update.

Appendix D Example CLI Commands

The following table shows examples of most CLI operations.

Note: The variable "*N*" (as in blade*N* or fantray*N*) represents the chassis slot number of the device being acted on (such as Blade5 or fantray1). Please refer to chassis documentation for slot, fan, fan tray, and power supply location and information.

Table 55. Example CLI Commands (Sheet 1 of 2)

Use Case	CLI Command	Return
Get current chassis temperature for "Temp Snsr 1".	cmmget –l chassis –t *"Temp Snsr 1"* –d current	Current Temperature for "Temp Sensr 1" on chassis.
Get fan tray 1 presence.	cmmget –l fantray1 –d presence	Fantray1 is present. Fantray1 is not present.
Get the baseboard temperature of blade 5 when the sensor is named "ambient temp".	cmmget –l blade5 –t *"ambient temp"*–d current	Current Temperature of "ambient temp" on blade 15.
Find out if blade 15 is present.	cmmget –l blade15 –d presence	Blade15 is present. Blade15 is not present.
Get all thresholds for the +3.3 V sensor on blade 2.	cmmget –l blade2 –t "+3.3V" –d thresholdsall	Returns thresholds for upper non-recoverable, upper critical, upper non-critical, lower non-critical, lower critical, and lower non-recoverable.
Get the overall system health.	cmmget –l system –d health	Indicates if the entire system is healthy, or has health events.
Get a list of blades with problems	cmmget –l system –d unhealthylocations	List of locations with active health events.
Get the CMM's overall health.	cmmget –l cmm –d health	Indicates if the CMM is healthy or if there are active health events on the CMM.
Display the SEL of the CMM	cmmget –l cmm –d sel	Displays a list of SEL entries (if any) for the CMM.
Display the SEL of blade19.	cmmget –l blade*19* –d sel	Displays a list of SEL entries (if any) for blade 19.
Get the version of firmware on the CMM	cmmget –l cmm –d version	Displays the version of firmware currently installed on the CMM.
Power off blade 19.	cmmset -l blade*19* –d powerstate –v poweroff	Displays a success or failure message.
Power on blade 20.	cmmset –l blade*20* –d powerstate –v poweron	Displays a success or failure message.
Reset blade 3.	cmmset –l blade*3*–d powerstate -v reset	Displays a success or failure message.
Get all of the chassis FRU information.	cmmget –l chassis -t fru -d all	Displays a list of all of the FRU information from the chassis.
Get all fantray2 FRU information	cmmget –l fantray2 –t fru –d all	Displays a list of all of the FRU information from the CMM.

Table 55. Example CLI Commands (Sheet 2 of 2)

Use Case	CLI Command	Return
Get a blade's FRU information	cmmget –l blade*[1-19]* –d fru	A list of FRU information.
Find out what sensors are on blade 3.	cmmget –l blade3 –d listtargets	A list of sensor names as defined in their SDRs on blade 3.
List the data items that can be queried or set on blade 4.	cmmget –l blade4 –d listdataitems	A list of dataitems for blade 4.
Find out what can be queried on blade4's +3. 3V sensor.	cmmget –l blade4 –t "+3.3V" –d listdataitems	A list of commands to be used as dataitems for the +3.3V.
Enable SNMP Traps	cmmset –l cmm –d snmpenable –v enable	Success or failure
Set SNMP Trap Address1	cmmset –l cmm –d snmptrapaddress1 –v 134.134.100.34	Success or failure
Set SNMP Community	cmmset –l cmm –d snmpcommunity –v public	Success or failure

Appendix E Datasheet Reference

This appendix provides links to datasheets, standards, and specifications for the technology designed into the CMM.

E.1 Intel® CompactPCI* Product Information

Information, collateral, and software updates for all CompactPCI-compatible Intel products can be found at:

http://developer.intel.com/design/network/products/cbp/linecard.htm

E.2 CompactPCI

Current CompactPCI Specifications can be purchased from PICMG* for a nominal fee. Short form specifications in Adobe Acrobat format (PDF) are also available on PICMG's website at:

http://www.PICMG.org/gcompactPCI.htm

E.3 IPMI

The Intelligent Platform Management Initiative is described and current specifications can be found at:

http://developer.intel.com/design/servers/ipmi/spec.htm

E.4 Intel® IOP310 Processor Chipset

For more information about the Intel® IOP310 Processor Chipset, see the following Web site:

http://developer.intel.com/design/iio/docs/iop310.htm

Appendix F Warranty Information

F.1 Intel® NetStructure™ Compute Boards & Platform Products Limited Warranty

Intel warrants to the original owner that the product delivered in this package will be free from defects in material and workmanship for two (2) year(s) following the latter of: (i) the date of purchase only if you register by returning the registration card as indicated thereon with proof of purchase; or (ii) the date of manufacture; or (iii) the registration date if by electronic means provided such registration occurs within 30 days from purchase. This warranty does not cover the product if it is damaged in the process of being installed. Intel recommends that you have the company from whom you purchased this product install the product.

THE ABOVE WARRANTY IS IN LIEU OF ANY OTHER WARRANTY, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ANY WARRANTY OF INFRINGEMENT OF ANY OTHER PARTY'S INTELLECTUAL PROPERTY RIGHTS, OR ANY WARRANTY ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

This warranty does not cover replacement of products damaged by abuse, accident, misuse, neglect, alteration, repair, disaster, improper installation or improper testing. If the product is found to be otherwise defective, Intel, at its option, will replace or repair the product at no charge except as set forth below, provided that you deliver the product along with a return material authorization (RMA) number (see below) either to the company from whom you purchased it or to Intel. If you ship the product, you must assume the risk of damage or loss in transit. You must use the original container (or the equivalent) and pay the shipping charge. Intel may replace or repair the product with either a new or reconditioned product, and the returned product becomes Intel's property. Intel warrants the repaired or replaced product to be free from defects in material and workmanship for a period of the greater of: (i) ninety (90) days from the return shipping date; or (ii) the period of time remaining on the original two (2) year warranty.

This warranty gives you specific legal rights and you may have other rights which vary from state to state. All parts or components contained in this product are covered by Intel's limited warranty for this product. The product may contain fully tested, recycled parts, warranted as if new.

F.1.1 Returning a Defective Product (RMA)

Before returning any product, contact an Intel Customer Support Group to obtain either a Direct

Return Authorization (DRA) or Return Material Authorization (RMA). Return Material

Authorizations are only available for products purchased within 30 days.

Return contact information by geography:

F.1.2 For the Americas

Return Material Authorization (RMA) credit requests e-mail address: requests.rma@intel.com

Direct Return Authorization (DRA) repair requests e-mail address: uspss.repair@intel.com

DRA on-line form: http://support.intel.com/support/motherboards/draform.htm

Intel Business Link (IBL): http://www.intel.com/ibl

Telephone No.: 1-800-INTEL4U or 480-554-4904

Office Hours: Monday - Friday 0700-1700 MST Winter / PST Summer

F.1.3 For EMEA

Return Material Authorization (RMA) e-mail address - emea.fs@intel.com

Direct Return Authorization (DRA) for repair requests e-mail address: emea.fs@intel.com

Intel Business Link (IBL): http://www.intel.com/ibl

Telephone No.: 00 44 1793 403063

Fax No.: 00 44 1793 403109

Office Hours: Monday - Friday 0900-1700 UK time

F.1.4 For APAC

RMA/DRA requests email address: apac.rma.front-end@intel.com

Telephone No.: 604-859-3111 or 604-859-3325

Fax No.: 604-859-3324

Office Hours: Monday - Friday 0800-1700 Malaysia time

Return Material Authorization (RMA) requests e-mail address: rma.center.jpss@intel.com

Telephone No.: 81-298-47-0993 or 81-298-47-5417

Fax No.: 81-298-47-4264

Direct Return Authorization (DRA) for repair requests, contact the JPSS Repair center.

E-mail address: sugiyamakx@intel.co.jp

Telephone No.: 81-298-47-8920

Fax No.: 81-298-47-5468

Office Hours: Monday - Friday 0830-1730 Japan time

If the Customer Support Group verifies that the product is defective, they will have the Direct Return Authorization/Return Material Authorization Department issue you a DRA/RMA number to place on the outer package of the product. Intel cannot accept any product without a DRA/RMA number on the package. Limitation of Liability and Remedies

INTEL SHALL HAVE NO LIABILITY FOR ANY INDIRECT OR SPECULATIVE DAMAGES (INCLUDING, WITHOUT LIMITING THE FOREGOING, CONSEQUENTIAL, INCIDENTAL AND SPECIAL DAMAGES) ARISING FROM THE USE OF OR INABILITY TO USE THIS

PRODUCT, WHETHER ARISING OUT OF CONTRACT, NEGLIGENCE, TORT, OR UNDER ANY WARRANTY, OR FOR INFRINGEMENT OF ANY OTHER PARTY'S INTELLECTUAL PROPERTY RIGHTS, IRRESPECTIVE OF WHETHER INTEL HAS ADVANCE NOTICE OF THE POSSIBILITY OF ANY SUCH DAMAGES, INCLUDING, BUT NOT LIMITED TO LOSS OF USE, BUSINESS INTERRUPTIONS, AND LOSS OF PROFITS. NOTWITHSTANDING THE FOREGOING, INTEL'S TOTAL LIABILITY FOR ALL CLAIMS UNDER THIS AGREEMENT SHALL NOT EXCEED THE PRICE PAID FOR THE PRODUCT. THESE LIMITATIONS ON POTENTIAL LIABILITIES WERE AN ESSENTIAL ELEMENT IN SETTING THE PRODUCT PRICE. INTEL NEITHER ASSUMES NOR AUTHORIZES ANYONE TO ASSUME FOR IT ANY OTHER LIABILITIES.

Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitations or exclusions may not apply to you.

Appendix G Customer Support

This appendix offers technical and sales assistance information for this product, and information on returning an Intel NetStructure product for service.

G.2 Technical Support and Return for Service Assistance

For all product returns and support issues, please contact your Intel product distributor or Intel Sales Representative for specific information.

G.3 Sales Assistance

If you have a sales question, please contact your local Intel ® NetStructure™ Sales Representative or the Regional Sales Office for your area. Address, telephone and FAX numbers, and additional information is available at Intel's website, located at:

http://www.intel.com/network/csp/sales/

Intel Corporation
Telephone (in U.S.) 1-800-755-4444
Telephone (Outside U.S.) 1-973-993-3030

Appendix H Agency Approvals

H.1 CE Certification

The ZT 7102 meets the intent of Directive 89/336/EEC for Electromagnetic Compatibility and Low-Voltage Directive 73/23/EEC for Product Safety. The ZT 7102 has been designed for NEBS/ETSI compliance.

H.2 Safety

UL/cUL 60950	Safety for Information Technology Equipment (UL File # E179737)
EN/IEC 60950	Safety for Information Technology Equipment
CB Report Scheme	CB certificate and Report

H.3 Emissions Test Regulations

FCC Part 15, Subpart B

EN 55022

CISPR 22

Bellcore GR-1089

H.3.5 EN 50081-1 Emissions

GR-1089-CORE	Sections 2 and 3
EN 55022	Class A Radiated
EN 55022	Power Line Conducted Emissions
EN 61000-3-2	Power Line Harmonic Emissions
EN 61000-3-3	Power Line Fluctuation and Flicker

H.3.6 EN 55024 Immunity

GR-1089-CORE	Sections 2 and 3
EN 61000 4-2	Electro-Static Discharge (ESD)
EN 61000 4-3	Radiated Susceptibility
EN 61000 4-4	Electrical Fast Transient Burst
EN 61000 4-5	Power Line Surge

EN 61000 4-6 Frequency Magnetic Fields

EN 61000 4-11 Voltage Dips, Variations, and Short Interruptions

H.4 Regulatory Information

H.4.7 FCC (USA)

This product has been tested and found to comply with the limits for a Class A digital device pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment.

This product generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Note: This device complies with Part 15 of the FCC Rules.

Caution: If you make any modification to the equipment not expressly approved by Intel, you could void your authority to operate the equipment.

H.4.8 Industry Canada (Canada)

Cet appareil numérique respecte les limites bruits radioélectriques applicables aux appareils numériques de Classe A prescrites dans la norme sur le matériel brouilleur: “Appareils Numériques”, NMB-003 édictée par le Ministre Canadien des Communications.

This digital apparatus does not exceed the Class A limits for radio noise emissions from digital apparatus set out in the interference-causing equipment standard entitled: “Digital Apparatus,” ICES-003 of the Canadian Department of Communications.

H.5 Product Safety Information

H.5.8.1 Safety Precautions

Review the following precautions to avoid personal injury and prevent damage to this product or products to which it is connected. To avoid potential hazards, use the product only as specified. Read all safety information provided in the component product user manuals and understand the precautions associated with safety symbols, written warnings, and cautions before accessing parts or locations within the unit. Save this document for future reference.

Warning: To **Avoid Electric Overload:** To avoid electrical hazards (heat shock and/or fire hazard), do not make connections to terminals outside the range specified for that terminal. See the product user manual for correct connections.

Warning: **To Avoid the Risk of Electric Shock:** When supplying power to the system, always make connections to a grounded main. Always use a power cable with a grounded plug (third grounding pin). Do not operate in wet, damp, or condensing conditions.

Caution: **System environmental requirements**: Components such as Processor Boards, Ethernet Switches, etc., are designed to operate with external airflow. Components can be destroyed if they are operated without external airflow. External airflow is normally provided by chassis fans when components are installed in compatible chassis. Never restrict the airflow through the unit's fan or vents. Filler panels or air management boards must be installed in unused chassis slots. Environmental specifications for specific products may differ. Refer to product user manuals for airflow requirements and other environmental specifications.

Warning: **Device heatsinks may be hot during normal operation**: To avoid burns, do not allow anything to touch heatsinks.

Warning: **Do Not Operate Without Covers:** To avoid electric shock or fire hazard, do not operate this product with any removed enclosure covers or panels.

Warning: **To Avoid the Risk of Electric Shock:** Do not operate in wet, damp, or condensing conditions. To avoid electric shock or fire hazard, do not operate this product with enclosure covers or panels removed.

Warning: **Avoid injury, fire hazard, or explosion**: Do not operate this product in an explosive atmosphere.

Warning: **If Your System Has Multiple Power Supply Sources:** disconnect all external power connections before servicing.

Warning: Power supplies must be replaced by qualified service personnel only.

Caution: **Lithium batteries are not field-replaceable units**. There is a danger of explosion if a battery is incorrectly replaced or handled. Do not disassemble or recharge the battery. Do not dispose of the battery in fire. When the battery is replaced, the same type or an equivalent type recommended by the manufacturer must be used. Used batteries must be disposed of according to the manufacturer's instructions. Return the unit to Intel for battery service.

Warning: **Avoid injury**: This product may contain one or more laser devices that are visually accessible depending on the plug-in modules installed. Products equipped with a laser device must comply with International Electrotechnical Commission (IEC) 60825.