SSI Compute Blade Specification

November 2010

Revision 1.0.1

Important Information and Disclaimers:

1. THE SERVER SYSTEM INFRASTRUCTURE PROMOTERS (“SSI PROMOTERS”) MAKE NO WARRANTIES WITH REGARD TOTHIS SSI SPECIFICATION (“SPECIFICATION”), AND IN PARTICULAR DOES NOT WARRANT OR REPRESENT THAT THIS SPECIFICATION OR ANY PRODUCTS MADE IN CONFORMANCE WITH IT WILL WORK IN THE INTENDED MANNER. NOR WILL SSI PROMOTERS ASSUME ANY RESPONSIBILITY FOR ANY ERRORS THAT THE SPECIFICATION MAY CONTAIN OR HAVE ANY LIABILITIES OR OBLIGATIONS FOR DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, INDIRECT, PUNITIVE, OR CONSEQUENTIAL DAMAGES WHETHER ARISING FROM OR IN CONNECTION WITH THE USE OF THIS SPECIFICATION IN ANY WAY.

2. NO REPRESENTATIONS OR WARRANTIES ARE MADE THAT ANY PRODUCT BASED INWHOLE OR PART ON THE ABOVE SPECIFICATION WILL BE FREE FROM DEFECTS OR SAFE FOR USE FOR ITS INTENDED PURPOSE. ANY PERSON MAKING, USING OR SELLING SUCH PRODUCT DOES SO AT HIS OR HER OWN RISK.

3. THE USER OF THIS SPECIFICATION HEREBY EXPRESSLY ACKNOWLEDGES THAT THE SPECIFICATION IS PROVIDED AS IS, AND THAT THE SSI PROMOTERS MAKE NO REPRESENTATIONS, EXTENDS NO WARRANTIES OF ANY KIND EITHER EXPRESS OR IMPLIED ORAL OR WRITTEN, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTY OR REPRESENTATION THAT THE SPECIFICATION OR ANY PRODUCT OR TECHNOLOGY UTILIZING THE SPECIFICATION OR ANY SUBSET OF THE SPECIFICATION WILL BE FREE FROM ANY CLAIMS OF INFRINGEMENT OF INTELLECTUAL PROPERTY, INCLUDING PATENTS, COPYRIGHTS AND TRADE SECRETS NOR DO THE SSI PROMOTERS ASSUME ANY OTHER RESPONSIBILITIES WHATSOEVER WITH RESPECT TO THE SPECIFICATION OR SUCH PRODUCTS.

4. A NON-EXCLUSIVE COPYRIGHT LICENSE IS HEREBY GRANTED TO REPRODUCE THIS SPECIFICATION FOR ANY PURPOSE PROVIDED THIS “IMPORTANT INFORMATION AND DISCLAIMERS SECTION (PARAGRAPHS 1-6) IS PROVIDED IN WHOLE.

5. UPON REQUEST FROM AN ADOPTER, THE SSI PROMOTERS WILL GRANT A NON-EXCLUSIVE, WORLD-WIDE LICENSE UNDER ANY NECESSARY CLAIMS, DEFINED IN THE ADOPTERS AGREEMENT, TO MAKE, HAVE MADE, USE, IMPORT, SELL, OFFER TO SELL, AND OTHERWISE DISTRIBUTE AND DISPOSE OF COMPLIANT PORTIONS, DEFINED IN THE ADOPTERS AGREEMENT, PROVIDED THAT SUCH LICENSE NEED NOT EXTEND TO ANY PART OR FUNCTION OF A PRODUCT IN WHICH A COMPLIANT PORTION IS INCORPORATED THAT IS NOT ITSELF PART OF THE COMPLIANT PORTION. SUCH LICENSE WILL BE GRANTED ON REASONABLE AND NONDISCRIMINATORY (“RAND”) TERMS AND MAY BE CONDITIONED UPON ADOPTER’S GRANT OF A RECIPROCAL LICENSE TO THE SSI PROMOTERS.

6. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY OTHER INTELLECTUAL PROPERTY RIGHTS IS GRANTED.

Revision 1.0.1

*Product names are trademarks, registered trademarks, or service marks of their respective owners.

Contents

Figures

Tables

Revision History

The following table lists the revision schedule based on revision number and development stage of the product.

Revision	Project Document State	Date
1.0.0	Initial public release.	9/18/2009
1.0.1	Bug fix: 143, 155, 270	11/10/2009

Note: Not all revisions may be published.

1 Introduction

1.1 Scope

This *SSI Compute Blade Specification* defines a set of standard hardware and management interfaces for the creation of blade server platforms. This document defines the requirements for an SSI Compute Blade.

1.2 Purpose

The primary purpose of the SSI Compute Blade Specification is to:

- Promote interoperability across building blocks from multiple vendors.
- Maximize the use of common off-the-shelf components from the server industry, thereby reducing customer development costs and speeding time to market.
- Enable innovation and differentiation, by allowing OEMs/ODMs to implement value added differentiation, rather than the standard server blade designs.

1.3 Audience

The primary audience for this specification is:

- Platform Architects
- System Architects
- Hardware Design Engineers
- Product Line Managers
- System Technologists and Planners
- Test and validation Engineers
- Marketing Engineers and Planners

1.4 Specification Compliance

Products making the claim of compliance with this specification **shall** provide, at a minimum, all features defined as mandatory by the use of the keyword "**shall**". Such products may also provide recommended features associated with the keyword "should" and permitted features associated with the keyword "may".

1.5 Reference Documents

- IPMI – Intelligent Platform Management Interface Specification, v2.0 rev 1.0E3, February 16, 2006, Copyright © 2004, 2005, 2006 Intel Corporation, Hewlett-Packard Company, NEC Corporation, Dell Inc., All rights reserved.
- IPMI – Platform Management FRU Information Storage Definition, V1.0, Document revision 1.1, September 27, 1999 Copyright © 1998, 1999 Intel Corporation, Hewlett-Packard Company, NEC Corporation, Dell Inc., All rights reserved.
- IPMI – Intelligent Platform Management Bus Communications Protocol Specification, V1.0, Document revision 1.0, November 15, 1999 Copyright © 1998, 1999 Intel Corporation, Hewlett-Packard Company, NEC Corporation, Dell Inc., All rights reserved.
- SCSI Primary Commands – 2 (SPC-2) Revision 3 or later, ANSI/INCITS 351-2001
- SCSI Reduced Block Commands (RBC), ANSI/INCITS 330-2000 [2006]
- Universal Serial Bus Mass Storage Class, Bulk-only Transport, Revision 1.0, Sept 31, 1999
- PICMG 3.0 Revision 2.0 AdvancedTCA Base Specification, March 18, 2005.
- *SSI Compute Blade Mezzanine Specification*
- *SSI System Management Guide*
- IEEE Std 802.3ap-2007 "Backplane Ethernet" standard
- Advanced Configuration and Power Interface (ACPI) specification 2.0b
- Extensible Firmware Interface (EFI) version 1.1
- Unified Extensible Firmware Interface (UEFI) version 2.0
- Trusted Computing Group (TCG) Trusted Platform Module (TPM) PC Client specifications version 1.2
- Windows Hardware Error Architecture (WHEA) Platform Design Guide version 1.0
- OEM Activation version 2.0 For Microsoft Windows Vista™ Operating systems
- DMTF - System Management BIOS (SMBIOS) Reference specification 2.3.4

1.6 Terms and Abbreviations

The following terms and acronyms are used in specific ways throughout this specification:

Table 1-1: Terms and Abbreviations

Term	Definition
ASHRAE	American Society of Heating, Refrigerating, and Air Conditioning Engineers
Base Management Interface (BMI)	This is the IPMB-based management interface used by the Chassis Manager to communicate with the blade management controllers.
blade	This is resource module that plugs into the blade chassis. A blade can provide many different types of resources to the chassis, including compute functions, storage capabilities, additional I/O interfaces and switching capabilities and special purpose capabilities. A blade can be a single-wide module (assumed) or a double-wide module, occupying two adjacent slots in the chassis.
Blade server	A system comprised of a chassis, chassis resources (power, cooling, Chassis Manager), compute blades, and communication (switch) blades. The chassis may contain additional modules, such as storage.
bottom	When used in reference to a board, the end that would be on the bottom in a vertically oriented chassis.
CFM	Cubic Feet per Minute. A measure of volumetric airflow. One CFM is equivalent to 472 cubic centimeters per second.
chassis	The mechanical enclosure that consists of the midplane, front boards, cooling devices, power supplies, etc. The chassis provides the interface to boards and consists of the guide rails, alignment, handle interface, face plate mounting hardware, and midplane interface.
chassis ground	A safety ground and earth return that is connected to the chassis metal and available to all PBAs.
Chassis Management Module (CMM)	Dedicated intelligent chassis module that hosts the Chassis Manager functionality.
Chassis Manager (CM)	Set of logical functions for hardware management of the chassis. This may be implemented by one or more dedicated Chassis Management Modules or by one or more blade management controllers and/or payload processors.
Chassis Management Controller (CMC)	Set of logical functions for hardware management of the chassis, implemented by one or more blade management controllers and/or payload processors.
cold start	Cold start is the time when blades receive the payload power for the first time.
component side 1	When used in reference to a PBA, the side on which the tallest electronic components would be mounted. Identical with the Right side as defined below. AKA Primary Side
component side 2	When used in reference to a PBA, the side normally reserved for making solder connections with through-hole components on Component Side 1 but on which low height electronic components may also be mounted. Identical with the Left side as defined below. AKA Secondary Side
creepage	Surface distance required between two electrical components.

Term	Definition
face plate	The front-most element of a PBA, perpendicular to the PBA, that serves to mount connectors, indicators, controls and mezzanines.
guide rail	Provides for the front board guidance feature in a slot.
handle	An item or part used to insert or extract blades into and out of chassis.
Intelligent Platform Management Bus (IPMB)	IPMB is an I^2C-based bus that provides a standardized interconnection between managed modules within a chassis. ftp://download.intel.com/design/servers/ipmi/ipmb1010ltd.pdf
Intelligent Platform Management Interface (IPMI)	IPMI v2.0 R1.0 specification defines a standardized, abstracted interface to the platform management subsystem of a computer system. ftp://download.intel.com/design/servers/ipmi/IPMIv2_0rev1_0.pdf
interconnect channel	An interconnect channel is comprised of two pairs of differential signals. One pair of differential signals for transmit and another pair of differential signals for receive.
Link	Network link representing either a single channel or aggregation of channels (example: KX4 using 4 channels would be a single Link). The term "Link" does not represent virtualized aggregation such as "teaming".
LFM	Linear Feet per Minute. A measure of air velocity. One LFM is equivalent to 0.508 centimeters per second.
logic ground	Chassis-wide electrical net used on blades and midplanes as a reference and return path for logic-level signals that are carried between boards.
managed module	Any component of the system that is addressable for management purposes via the specified management interconnect and protocol. A managed module is interfaced directly to the chassis BMI.
Management Controller	An intelligent embedded microcontroller that provides management functionality for a blade or other chassis module.
may	Indicates flexibility of choice with no implied preference.
mezzanine	The mezzanine is a PBA that installs on a blade PBA horizontally. It provides additional functionality on the blade PBA and provides electrical interface between the blade PBA and the midplane PBA. Both the blade PBA and mezzanine PBA are contained inside the blade module.
midplane	Equivalent to a system backplane. This is a PBA that provides the common electrical interface for each blade in the chassis and on both the front and back of the PBA.
module	A physically separate chassis component that may be independently replaceable (e.g., a blade or cooling module) or attached to some other component (e.g., a mezzanine board).
Compute Blade	A blade that conforms to the requirements defined by the SSI standard set of specifications.
out-of-band (OOB)	Communication between blades that does not need the host or payload to be powered on
payload	The hardware on a blade that implements the main mission function of the blade. On a compute blade, this includes the main processors, memory, and I/O interfaces. The payload is powered separately from the blade management subsystem. Payload power is controlled by the blade management controller.
PBA	Printed board assembly. A printed board assembly is a printed circuit board that has all electronic components attached to it.

Term	Definition
PCB	Printed circuit board without components attached.
Peak power	The peak power drawn by a blade is defined as the maximum power that a blade can draw for a very short period of time during a hot insertion, hot removal, or during a cold start.
pitch line	Horizontal pitch line between slots.
SAP	Service Applet Package
shall	Indicates a mandatory requirement. Designers must implement such mandatory requirements to ensure interchangeability and to claim conformance with this specification. The use of **shall not** indicates an action or implementation that is prohibited.
should	Indicates flexibility of choice with a strongly preferred implementation. The use of **should not** indicates flexibility of choice with a strong preference that the choice or implementation be avoided.
slot	A slot defines the position of one blade in a chassis
top	When used in reference to a blade the end which would be on top in a vertically oriented chassis.
U	Unit of vertical height defined in IEC 60297-1 rack, shelf, and chassis height increments. 1U=44.45 mm.
WDT	Watchdog timer

2 SSI Compute Blade Architecture

2.1 Blade Server Goals

Blade servers are gaining in popularity, but no standardization exists, making it difficult for ODMs and LOEMs to build products that interoperate and are easy and inexpensive to design.

2.1.1 High Level Strategic Goals

The following are some of the major high level strategic goals of the current blade server specifications:

- Interoperability.
- Reduced R&D cost and efforts through design re-use from 1U rack-mount server designs.
- Faster TTM.
- Ease of use through standards based management functionality.
- OEM differentiation through configuration, system power management, performance, and cost optimizations.
- Investment protection by creating technology roadmap and feature headroom support (e.g., support of future interconnection technologies like IEEE Std 802.3ap-2007 "Backplane Ethernet" standard defined serial 10 Gbps Ethernet, PCI Express* gen2, future multi-core processor support, headroom for more DIMM memories etc).
- System thermal and mechanicals designs to fit in the existing data center infrastructures without requiring massive data center re-design through expensive cooling systems and/or onsite support technicians.

2.1.2 High Level Technical Goals

The following are some of the high level technical goals of the current blade server specifications:

- Best in class (perf/W/$) dual socket processor designs.
- Re-use of high volume server enabled components (e.g. industry standard vertical DIMMs, heat sinks, hard disk drives, etc).
- Hot insertion and extraction of blades.
- Single 12V bulk DC power input power to blades.
- Support for up to 16 DIMMs.

- Up to two Small Form Factor (SFF) hot swappable hard drives.
- Flexible secondary fabric support through mezzanine cards.
- Front to rear cooling.
- Operating Ambient temperature of 10 – 35°C
- Single mezzanine card in a single-wide blade and two mezzanines in a double-wide blade.
- User accessible KVM and serial console on the front of each chassis.

2.2 SSI Blade Server System Functional Blocks

The SSI Blade Server Specifications define a modular decomposition of a blade server into the following components:

Chassis Midplane Providing common, shared power, cooling, interconnection, and management resources

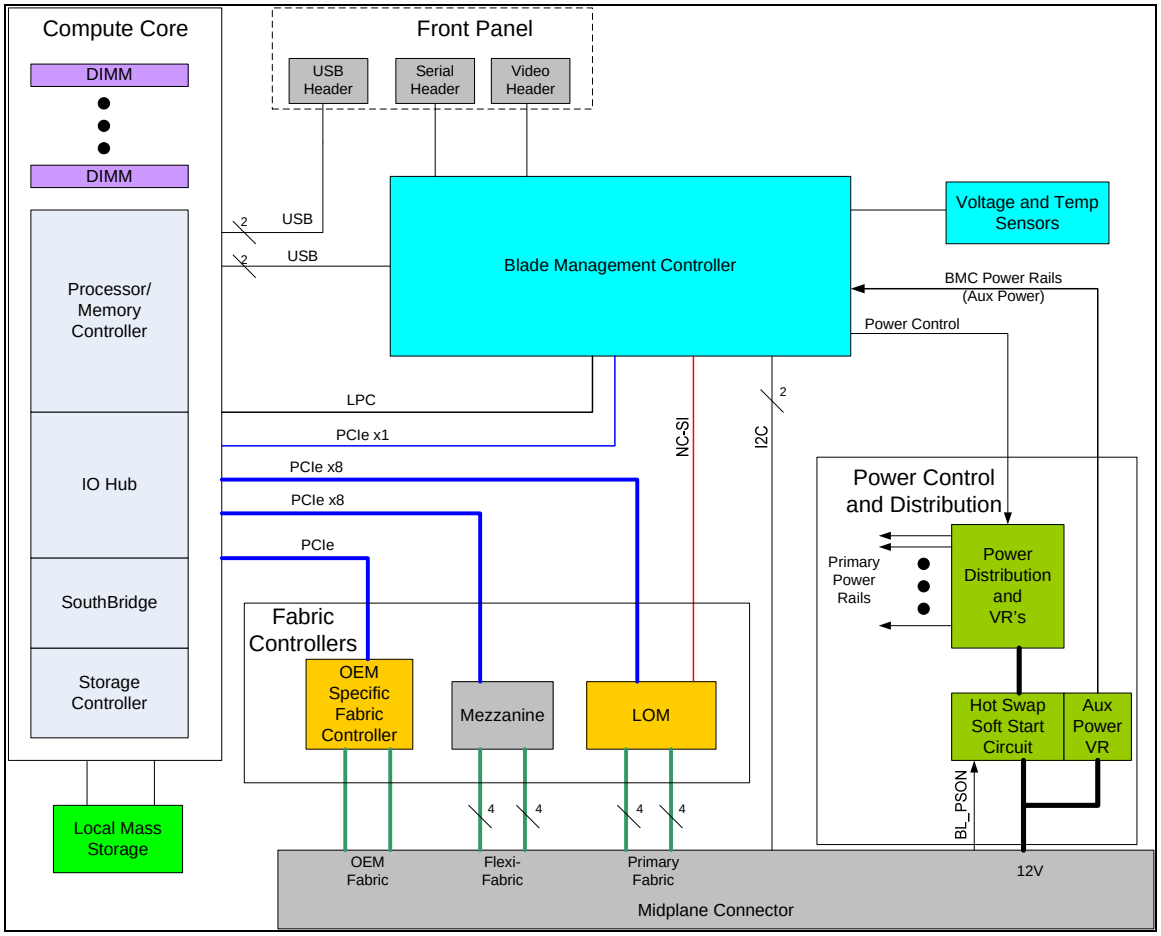
Compute Blades Providing the mission execution portion of a blade server. These may be purely computational engines, e.g., 2 or 4 processor server equivalents, or more task specific modules comprised of specialized hardware.

Switches Providing communications and data transport interconnections both intra-chassis, between compute blades and other modules, and inter-chassis (to other chassis) or networks. Switch modules may support different technologies, such as Ethernet, Fibre Channel, or PCI Express.

CMMs Providing the hardware and software chassis management module interfaces for controlling the power, thermal, and module identification tasks.

An SSI Compute Blade server consists minimally, of a chassis that accepts compute blades and at least one SSI switch. The chassis provides adequate power and cooling for its blade population.

Figure 2-1: Typical Compute Blade Functional Block Diagram



The block diagram shows common compute blade subsystems: the compute core, fabric controllers, power control and distribution, and the blade management.

Compute core: The compute core consists of the processor, memory controller, IO hub, South Bridge, and Storage controller, as well as local memory or DIMMs. Many of these elements with the exception of DIMMs can be implemented with various levels of integration, from the traditional discrete processor(s), memory hub controller, south bridge, etc., to a completely integrated part.

Local mass storage is an option to improve performance in certain applications (e.g., OS and Swap space). An alternative implementation is to locate the mass storage off the blade accessed through one of the high speed fabrics. One option is to implement a SAS connection over the OEM fabric with the SAS lanes going to a central shared disk subsystem.

If the blade is implemented with local mass storage, there are two primary technologies to consider, traditional rotating magnetic media (Hard Disk Drive) or solid state disks (SSD) implemented with flash technology. In the past, SSD technology has been avoided due to capacity limitations but now 32GB are common. SSDs also suffered from wear out problems associated with flash technology which in turn limits

the SSD's useful life. With appropriate wear leveling algorithms, the useful life of SSD's are now over 5 years and increasing, making them viable options for some applications.

Fabric controllers: To interface with the midplane there is a LAN On Motherboard (LOM) controller connecting to the primary fabric. The mezzanine may connect to the secondary fabric. An OEM specific controller is used to connect to the OEM fabric. See chapter 7 for more details.

Power control and distribution: This sub-system receives 12V from the midplane, controls the hot plug/soft start capability and generates all voltages for the entire blade. Note the BMC receives a 3.3V aux voltage generated from the same 12V rail on the midplane as the rest of the blade. See chapter 4 for more details.

Blade management: See chapter 6 for details about the Blade Management Controller.

3 Compute Blade PBA Mechanical Specifications

This section provides the mechanical specifications for the compute blade PBA.

3.1 Compute Blade PBA Dimensions

The compute blade PBA form factor **shall** be as defined in Figure 3-1. Datums have been defined to explain the relationship between the compute blade PBA, the blade enclosure, and the server midplane.

Datum 'A' is established as the top surface or 'Side A' surface of the compute blade PBA. The Blade Power Connector, Blade Signal Connector, Optional Blade Signal Connector, and Guide Module Connector are mounted to this surface. Datum 'B' and Datum 'C' **shall** be used to define the location of the compute blade PBA I/O connectors using Cartesian coordinates.

As shown, the overall compute blade PBA dimensions **shall** be 269.24 mm [10.600"] by 406.40 mm [16.000"].

Figure 3-2 details the compute blade I/O connectors, including the Hi-Speed Mezzanine Signal I/O signal connector on the High Speed Mezzanine PBA. The mounting holes shown are to be supported on the compute blade PBA for use in securing the High Speed Mezzanine in place. For direction in determining compute blade maximum component heights for components beneath the mezzanine PBA, see the *SSI Compute Blade Mezzanine Specification* reference document.

Figure 3-3 defines the maximum component height for all compute blade PBA components, including heatsinks. There should be no compute blade PBA components located directly underneath the location of the High Speed Mezzanine Signal Connector. All dimensional references and tolerance analyses assume that the High Speed Mezzanine PBA is 2.36 mm [0.093"] ± 0.20 [0.008"] in thickness.

The mezzanine card interfaces to the compute blade via an x8 PCI Express* connection through a 120-pin connector, Tyco* 0.8mm pitch Free Height connector or equivalent. An optional, second 80-pin connector (Tyco* 0.8mm pitch Free Height connector) may be used for an additional x8 PCIe interface. See the *SSI Compute Blade Mezzanine Specification* reference document for specific part numbers.

Figure 3-1: Compute Blade PBA Dimensions

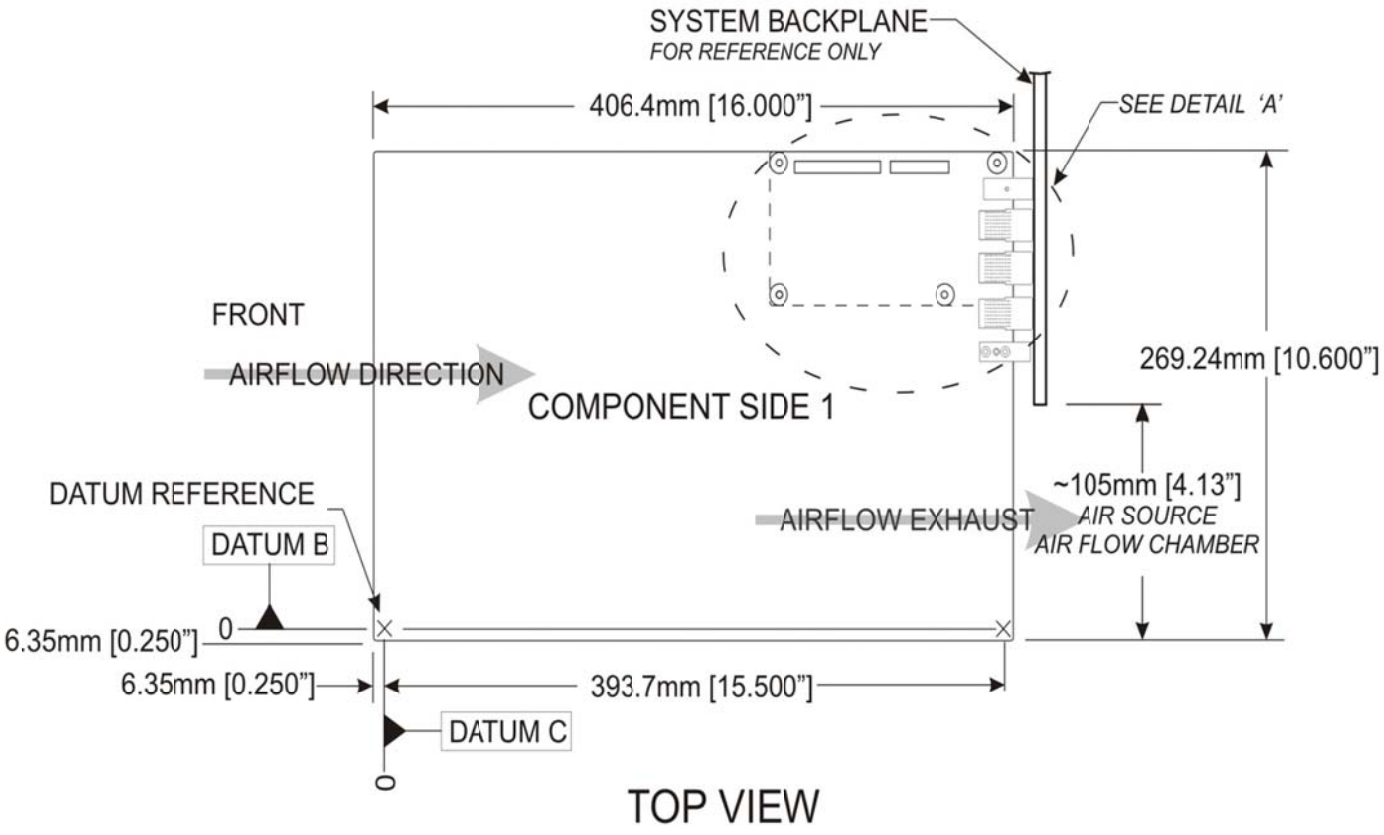


Figure 3-2: Compute Blade IO Connectors and High Speed Mezzanine PBA, Detail A

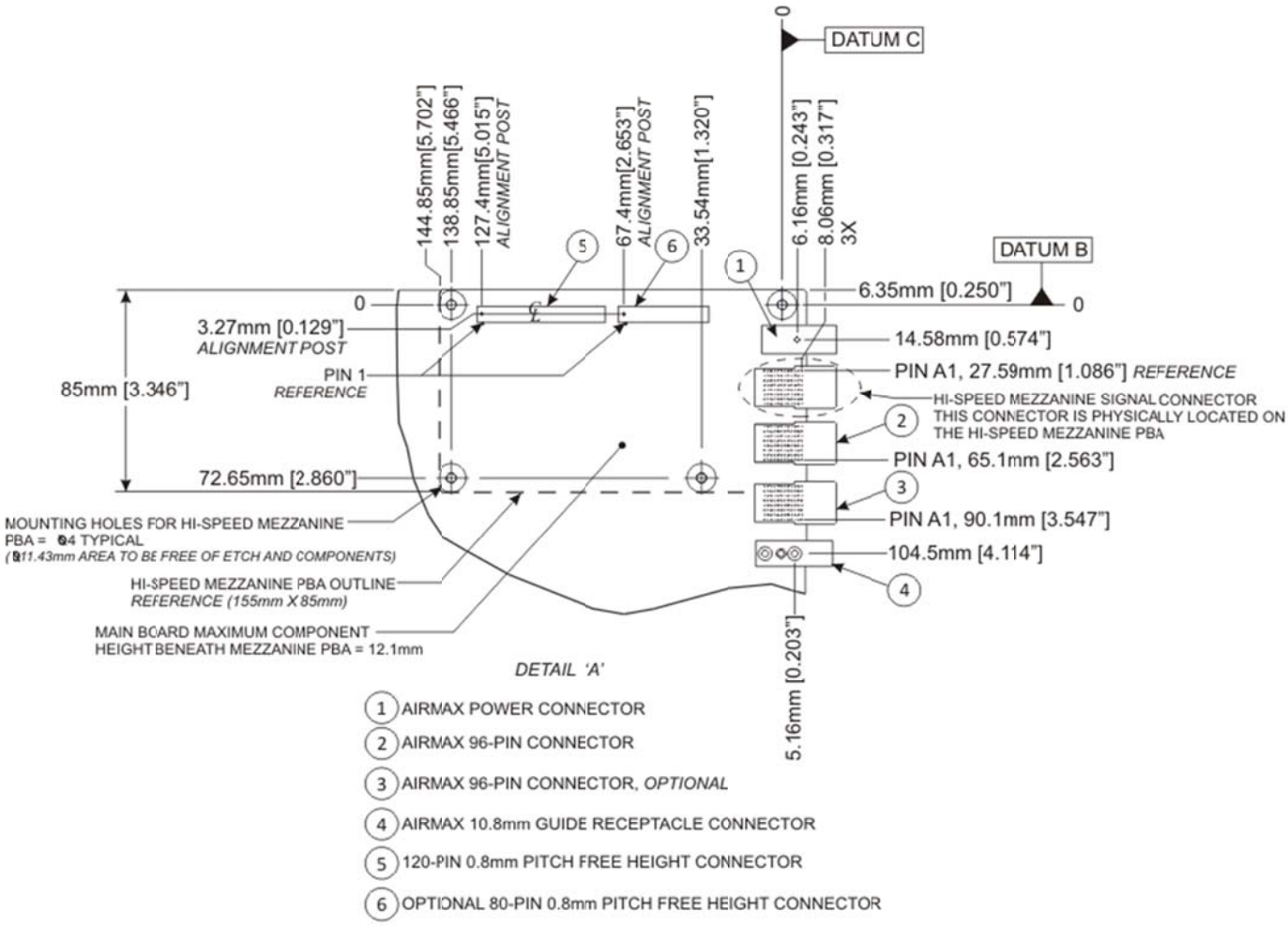
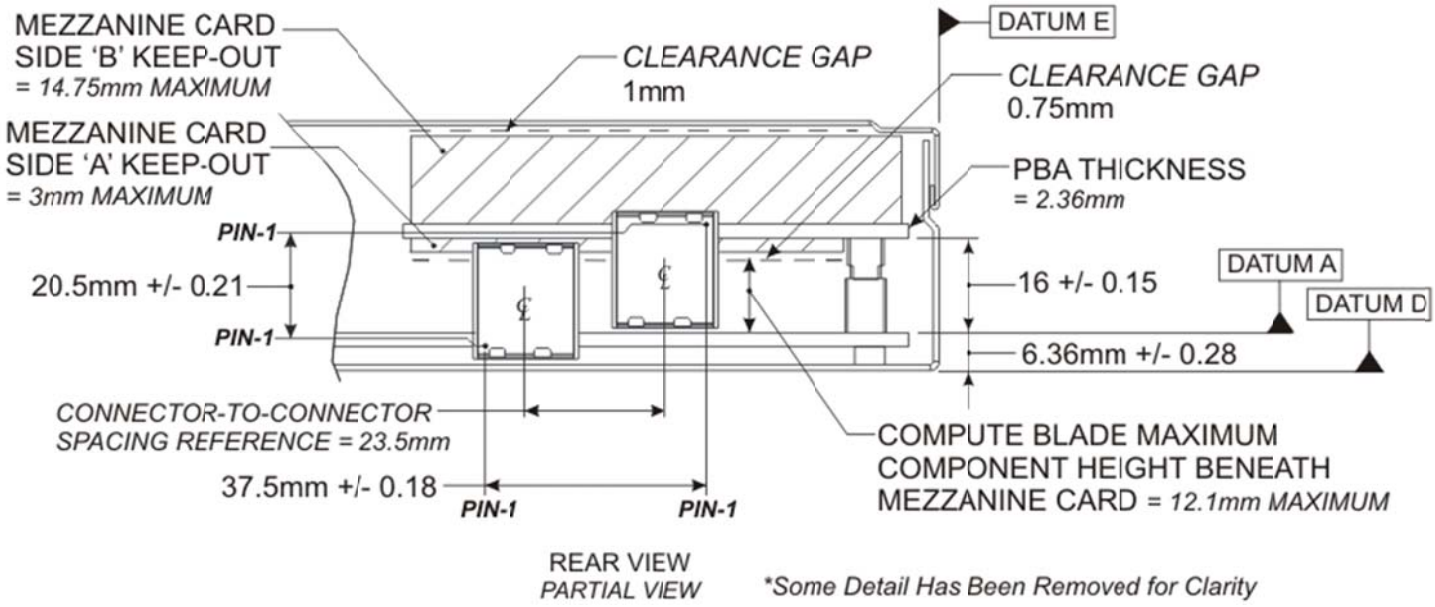


Figure 3-3: Clearance Considerations for the High Speed Mezzanine PBA and the Compute Blade PBA



Note: The Hi-Speed mezzanine PBA Connector provides the optional Flexi-Interconnect.

The addition of the optional High Speed Mezzanine PBA will restrict the height of components that can reside on the compute blade PBA beneath the mezzanine PBA in that area. The recommended clearance dimensions are depicted in Figure 3-3. These dimensions are based on conservative tolerance analysis deviations. The 16.00mm dimension is based on the Tyco* 0.8mm pitch Free Height connector or equivalent. Compute blade maximum component heights for components beneath the mezzanine PBA, including components requiring heatsinks, must not exceed the 12.10mm maximum height requirement.

The dimensions between the Compute Blade Signal Connector and the High Speed Signal Mezzanine Connector have been defined, with appropriate tolerances. These dimensions and tolerances must be closely obeyed in the development of the blade server midplane PBA design, and how the midplane PBA interfaces with the compute blade enclosure mounting surfaces. Additional information is provided in Chapter 9.

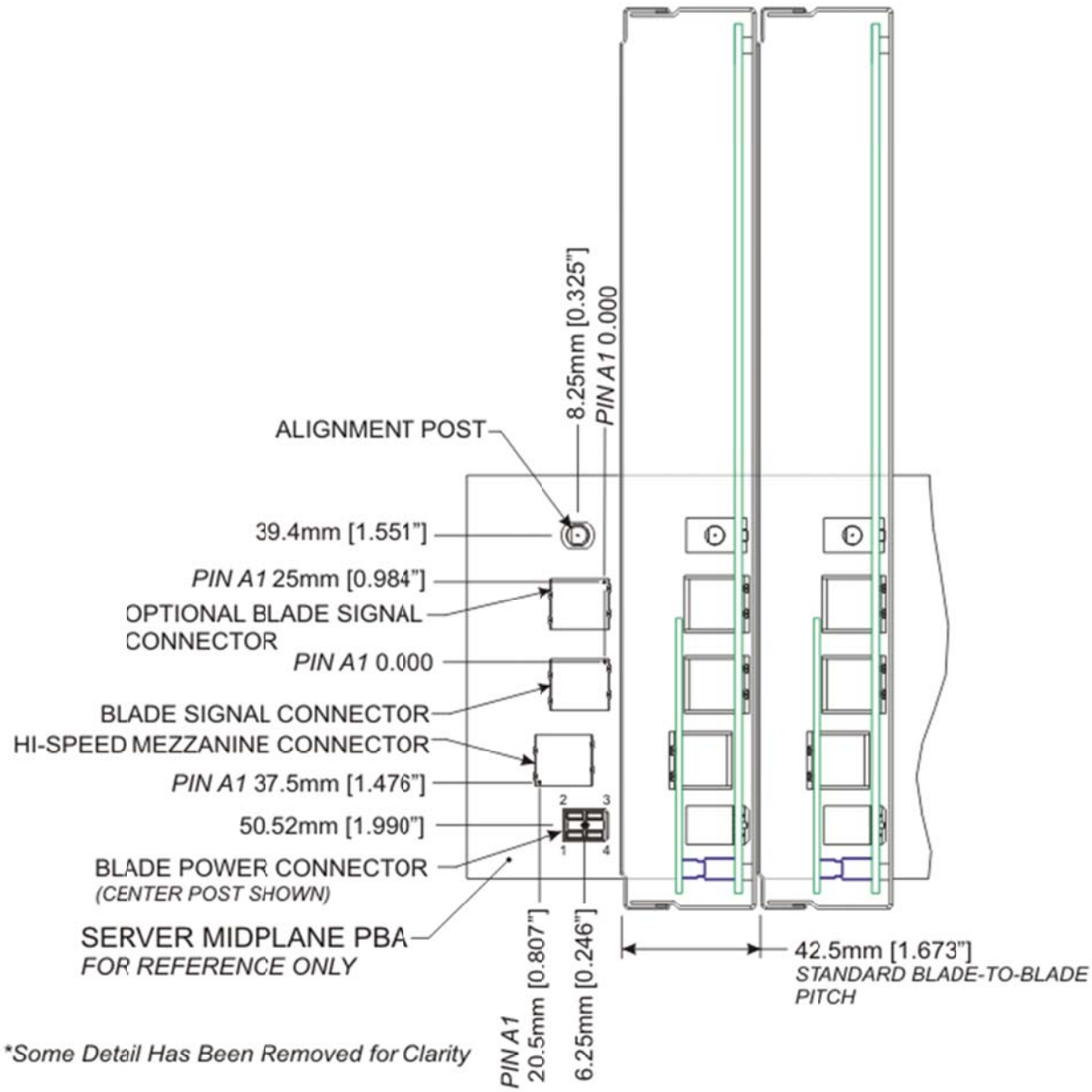
3.1.1 Compute Blade PBA Thickness

The compute blade PBA thickness **shall not** exceed 2.36 mm [0.093"] ± 0.20 [0.008"].

3.1.2 Compute Blade I/O Connector Reference to the Chassis Midplane

Figure 3-4 depicts a representation of the server midplane. The midplane mating connector type and locations have been defined.

Figure 3-4: Server Midplane PBA Reference



SERVER MIDPLANE PBA EXAMPLE
VIEW FROM THE FRONT OF THE COMPUTE BLADE
PARTIAL VIEW

3.2 Compute Blade Mechanical, Thermal, and PBA Offset

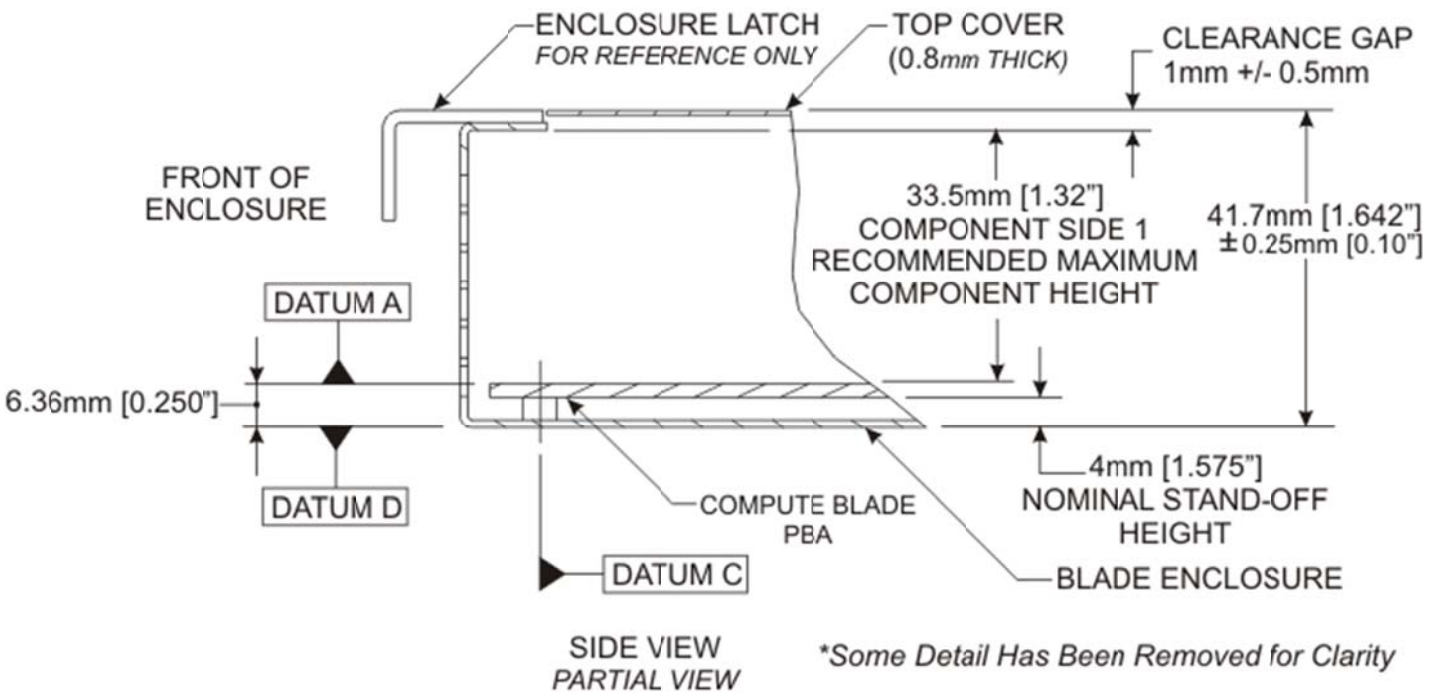
The blade mechanical and thermal limits are determined by various factors:

- Compute blades total power.
- Compute blade usage and heat sink volumetrics (length x width x height).
- Server density, total power, and cooling capabilities of the chassis and the capability of the facility where the chassis is deployed.

Based on analysis of these factors, the following key specification parameters have been established:

- The compute blade-to-blade pitch **shall** be 42.50 mm [1.673”].
- The compute blade enclosure height **shall** be 41.70 mm [1.642”].
- The compute blade PBA offset **shall** be 6.36 mm [0.250”].

Figure 3-5: Compute Blade Dimensional Requirements



In Figure 3-5, Datum ‘A’ defines the top surface of the compute blade PBA component side 1. Datum ‘C’ establishes a means to locate the blade PBA connectors. Datum ‘D’ defines the mounting surface of the blade enclosure. Datum ‘D’ is the interface surface between the blade enclosure and the server chassis. Datum ‘D’ establishes the relationship between the blade enclosure’s mounting surface and Datum ‘A’, the compute blade PBA, and thus, the midplane connectors, which are mounted to Datum ‘A’.

3.3 PBA Component Height

The specified component heights assume the material thickness of the module enclosure is 1.00 mm [0.039”]. For the purposes of this document, the enclosure security cover is assumed to be 0.08 mm [0.032”] thick. Bow and warp of the enclosure and the compute blade PBA should be carefully considered when selecting components. Stiffeners, insulators, and/or other preventative measures may be necessary to reduce the risk of component interference, shorting, and/or other issues. The component heights for the primary and secondary sides assume a maximum PBA thickness of 2.36 mm [0.093”] ± 0.20 [0.007”]. Any deviation from the recommended material thicknesses will directly impact the PBA component heights.

3.3.1 Primary Side Component Height

When using a 0.80 mm [0.032”] thick blade enclosure top cover, the component height on the primary side of the compute blade PBA **shall** be no greater than 33.5 mm [1.32”]. This allows for a 1.0 mm clearance gap between the compute blade PBA components and the blades top cover. Bow and warp of the top cover must be addressed by the blade enclosure manufacturer. See Figure 3-5 for details.

3.3.2 Secondary Side Component Height

When using a 1.0 mm [0.039”] thick blade enclosure base, the maximum component height on the secondary side of the compute blade PBA should be 2.8 mm [0.11”]. See Figure 3-5 for details.

3.3.3 PBA Component Keep-out Zones

Unless explicitly stated otherwise, keep-out zones defined in this specification apply to both component side 1 and component side 2 surfaces of a PBA, including the outer layers of the PBA. Traces may be routed on interior layers if proper creepage and clearance distances are maintained to all holes within the keep-out zones; holes in keep-out zones without explicit signal definitions **shall** be considered to be chassis ground for purposes of determining minimum creepage and clearance distances.

3.4 Compute Blade PBA Connectors

Connector specifications that are described in this section are applicable to single- and double-wide blade modules.

This section does not discuss the interface connector between the Compute Blade PBA and the mezzanine PBA. For that information, see the mezzanine PBA reference document.

The Compute Blade PBA **shall** have at least three connectors:

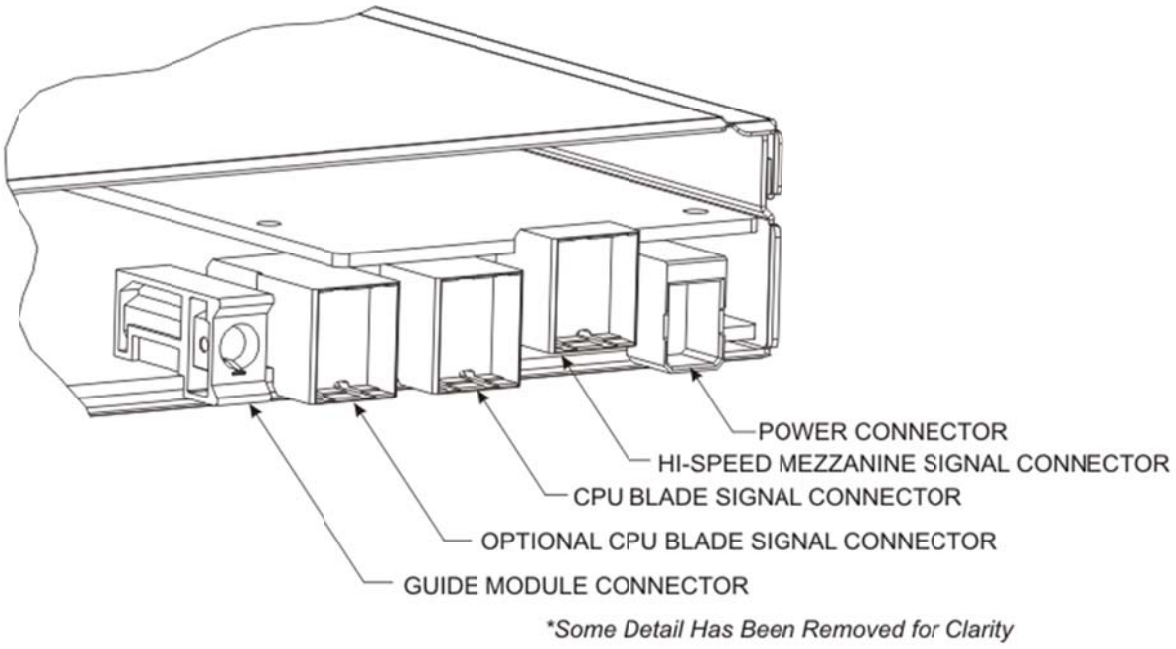
- A Compute Blade signal connector
- A 12-volt DC power connector

- A guide module connector [143]. This connector carries no signals, but is used to guide the insertion of the blade module into the midplane.

Figure 3-6 shows these connectors on the compute blade module and the connector on the mezzanine PBA. The connector on the mezzanine PBA provides the capabilities for the Flexi-Interconnect. For details of the Flexi-Interconnect and mezzanine PBA specification, see the SSI Compute Blade Mezzanine Specification.

See Figure 3-2 and Figure 3-4 for the precise locations of these connectors.

Figure 3-6: Compute Blade Power and Signal Connectors



3.4.1 Compute Blade Signal Connector

Each blade **shall** use an Airmax* 96-pin connector (see Table 3-4 for part number) or equivalent for the high-speed data transport interface and management interface. This connector provides the electrical interface for up to ten I/O channels, forming the connection between the blade and the midplane PBA. Supported interfaces are described in Table 3-1.

Table 3-1: Compute Blade Signal Connector

	A	B	C	D	E	F	G	H	I	J	K	L
8	Ground	P2_CH_C_TXP	P2_CH_C_TXN	Ground	P1_CH_B_TXP	P1_CH_B_TXN	Ground	Resvd	BL_SLT_ID5	Ground	Resvd	Resvd
7	P2_CH_C_RXP	P2_CH_C_RXN	Ground	P1_CH_B_RXP	P1_CH_B_RXN	Ground	Resvd	Resvd	Ground	Resvd	Resvd	Ground
6	Ground	P1_CH_C_TXP	P1_CH_C_TXN	Ground	P2_CH_B_TXP	P2_CH_B_TXN	Ground	OEM_CH_B_TXP	OEM_CH_B_TXN	Ground	Resvd	Resvd
5	P1_CH_C_RXP	P1_CH_C_RXN	Ground	P2_CH_B_RXP	P2_CH_B_RXN	Ground	OEM_CH_B_RXP	OEM_CH_B_RXN	Ground	Resvd	Resvd	Ground
4	Ground	P2_CH_A_TXP	P2_CH_A_TXN	Ground	CMM_Spare	BL_PSON (shrtpin)	Ground	P1_CH_D_TXP	P1_CH_D_TXN	Ground	BL_SLT_ID4	BL_PRES_N
3	P2_CH_A_RXP	P2_CH_A_RXN	Ground	SMB_SCL_B	SMB_SDA_B	Ground	P1_CH_D_RXP	P1_CH_D_RXN	Ground	BL_SLT_ID2	BL_SLT_ID3	Ground
2	Ground	P1_CH_A_TXP	P1_CH_A_TXN	Ground	P2_CH_D_TXP	P2_CH_D_TXN	Ground	OEM_CH_A_TXP	OEM_CH_A_TXN	Ground	BL_SLT_ID0	BL_SLT_ID1
1	P1_CH_A_RXP	P1_CH_A_RXN	Ground	P2_CH_D_RXP	P2_CH_D_RXN	Ground	OEM_CH_A_RXP	OEM_CH_A_RXN	Ground	SMB_SCL_A	SMB_SDA_A	Ground
	A	B	C	D	E	F	G	H	I	J	K	L

Note: In the above table, P1 = Primary Midplane interconnect and P2 = Second Primary Midplane interconnect. The green cells indicate GND connections

Table 3-2: Compute Blade Signal Names, Interconnects, and Connector Locations

Signal Name	Signal Description	Purpose	Connector Location
P1_CH_A_RXP	RX+ of Primary CH 1	Mid-plane Interconnect	A1
P1_CH_A_RXN	RX- of Primary CH 1	Mid-plane Interconnect	B1
P1_CH_B_RXP	RX+ of Primary CH 2	Mid-plane Interconnect	D7
P1_CH_B_RXN	RX- of Primary CH 2	Mid-plane Interconnect	E7
P1_CH_C_RXP	RX+ of Primary CH 3	Mid-plane Interconnect	A5
P1_CH_C_RXN	RX- of Primary CH 3	Mid-plane Interconnect	B5
P1_CH_D_RXP	RX+ of Primary CH 4	Mid-plane Interconnect	G3
P1_CH_D_RXN	RX- of Primary CH 4	Mid-plane Interconnect	H3

Signal Name	Signal Description	Purpose	Connector Location
P1_CH_A_TXP	TX+ of Primary CH 1	Mid-plane Interconnect	B2
P1_CH_A_TXN	TX- of Primary CH 1	Mid-plane Interconnect	C2
P1_CH_B_TXP	TX+ of Primary CH 2	Mid-plane Interconnect	E8
P1_CH_B_TXN	TX- of Primary CH 2	Mid-plane Interconnect	F8
P1_CH_C_TXP	TX+ of Primary CH 3	Mid-plane Interconnect	B6
P1_CH_C_TXN	TX- of Primary CH 3	Mid-plane Interconnect	C6
P1_CH_D_TXP	TX+ of Primary CH 4	Mid-plane Interconnect	H4
P1_CH_D_TXN	TX- of Primary CH 4	Mid-plane Interconnect	I4
P2_CH_A_RXP	RX+ of Second Primary CH 7	Mid-plane Interconnect	A3
P2_CH_A_RXN	RX- of Second Primary CH 7	Mid-plane Interconnect	B3
P2_CH_B_RXP	RX+ of Second Primary CH 8	Mid-plane Interconnect	D5
P2_CH_B_RXN	RX- of Second Primary CH 8	Mid-plane Interconnect	E5
P2_CH_C_RXP	RX+ of Second Primary CH 9	Mid-plane Interconnect	A7
P2_CH_C_RXN	RX- of Second Primary CH 9	Mid-plane Interconnect	B7
P2_CH_D_RXP	RX+of Second Primary CH 10	Mid-plane Interconnect	D1
P2_CH_D_RXN	RX- of Second Primary CH 10	Mid-plane Interconnect	E1
P2_CH_A_TXP	TX+ of Second Primary CH 7	Mid-plane Interconnect	B4
P2_CH_A_TXN	TX- of Second Primary CH 7	Mid-plane Interconnect	C4
P2_CH_B_TXP	TX+ of Second Primary CH 8	Mid-plane Interconnect	E6
P2_CH_B_TXN	TX- of Second Primary CH 8	Mid-plane Interconnect	F6
P2_CH_C_TXP	TX+ of Second Primary CH 9	Mid-plane Interconnect	B8
P2_CH_C_TXN	TX- of Second Primary CH 9	Mid-plane Interconnect	C8

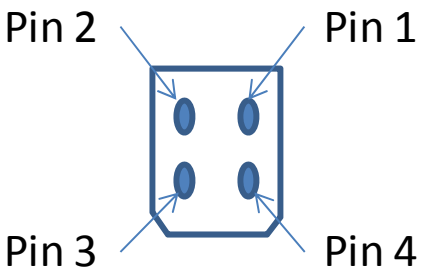
Signal Name	Signal Description	Purpose	Connector Location
P2_CH_D_TXP	TX+ of Second Primary CH 10	Mid-plane Interconnect	E2
P2_CH_D_TXN	TX- of Second Primary CH 10	Mid-plane Interconnect	F2
OEM_CH_A_RXP	RX+ of OEM Channel CH5	Mid-plane Interconnect	G1
OEM_CH_A_RXN	RX- of OEM Channel CH5	Mid-plane Interconnect	H1
OEM_CH_A_TXP	TX+ of OEM CH5	Mid-plane Interconnect	H2
OEM_CH_A_TXN	TX- of OEM CH5	Mid-plane Interconnect	I2
OEM_CH_B_RXP	RX+ of OEM CH6	Mid-plane Interconnect	G5
OEM_CH_B_RXN	RX- of OEM CH6	Mid-plane Interconnect	H5
OEM_CH_B_TXP	TX+ of OEM CH6	Mid-plane Interconnect	H6
OEM_CH_B_TXN	TX- of OEM CH6	Mid-plane Interconnect	I6
BL_SLT_ID0	Blade ID Address 0	Blade Management	K2
BL_SLT_ID1	Blade ID Address 1	Blade Management	L2
BL_SLT_ID2	Blade ID Address 2	Blade Management	J3
BL_SLT_ID3	Blade ID Address 3	Blade Management	K3
BL_SLT_ID4	Blade ID Address 4	Blade Management	K4
BL_SLT_ID5	Blade ID Address 5	Blade Management	I8
SMB_SCL_A	SM Bus A Serial CLK	Blade Management	J1
SMB_SDA A	SM Bus A Serial Data	Blade Management	K1
SMB_SCL_B	SM Bus B Serial CLK	Blade Management	D3
SMB_SDA_B	SM Bus B Serial Data	Blade Management	E3
BL_PSON	Blade Power On (short pin)	PWR_ON Control	F4
BL_PRES_N	Blade Present (Active Low)	Blade Management	L4

Signal Name	Signal Description	Purpose	Connector Location
GND	Ground Pins	Ground	A2,A4,A6,A8, C1,C3,C5,C7, D2,D4,D6,D8, F1,F3,F5,F7, G2,G4,G6,G8 I1,I3,I5,I7 J2,J4,J6,J8 L1,L3,L5,L7
Reserved	Reserved for future use. Do not use	NC	J5, K5, K6, L6, G7, H7, J7, K7, H8, K8, L8
CMM_Spare	Spare signal routed on the midplane to the CMMs for future use.	Spare (leave unconnected on blade)	E4

3.4.2 Compute Blade Power Connector

The power connector supplies +12V DC to the blade module. Each blade **shall** use right angle Airmax* VS power connector (see Table 3-4 for part number) or equivalent. The connector has four power contacts (two for +12VDC and two for Ground contacts). A single connector is capable of supplying power to single-wide compute blades. Pins 1 and 4 are ground. Pins 2 and 3 are +12 VDC.

Figure 3-7: Blade Power Connector Pinout (view from midplane)



The compute blade connector **shall** contain the male contacts and **shall** contain right-angle PBA-mounted terminations. The midplane connector **shall** contain female contacts and has straight PBA-mounted terminations. Each blade **shall** support sequential mating to provide the proper functionality during live insertion or extraction of the compute blade. Blade ground pins **shall** make the first contact. These are the longest pins. The +12V supply pins, the shortest pins, **shall** be mated next. The signal pins **shall** be used for the power on control signal BL_PSON to ensure the last contact. See section 4.7.1.1 for information about the BL_PSON.

The connector provides for a 2-degree, 2 mm misalignment in transverse and longitudinal directions. Press-fit versions of the midplane and compute blade connectors **shall** have retention screws.

3.5 Mechanical Module Guide Connector

To ensure a proper connection of the blade to the midplane, each blade **shall** incorporate the mechanical guide receptacle module connector, as shown by Figure 3-2 and Figure 3-3, and **shall** use an Airmax* VS guide connector (See Table 3-4 for part number) or equivalent.

3.6 Optional CPU Compute Blade Signal Connector

An optional Airmax* 96-pin connector (See Table 3-4 for part number) is for OEM specific usage. Pins are defined either for signals or grounds. Ground pins below must be tied to ground. Signal pins **shall not** be used for power. See Table 3-3.

Table 3-3: Optional CPU Compute Blade Signal Connector Pinout

	A	B	C	D	E	F	G	H	I	J	K	L
8	Ground	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal
7	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal	Ground
6	Ground	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal
5	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal	Ground
4	Ground	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal
3	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal	Ground
2	Ground	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal
1	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal	Ground	Signal	Signal	Ground
	A	B	C	D	E	F	G	H	I	J	K	L

3.7 Compute Blade to Midplane Connector Part Numbers

Table 3-4: Compute Blade Connectors

Connector	Description	Manufacturer	Part Number
CPU Compute Blade Signal Connector	4-pair vert, 2mm, 96-pin plug	FCI*	10084604-111LF
Optional CPU Compute Blade Signal Connector	4-pair vert, 2mm, 96-pin plug	FCI	10084604-111LF
Power Module Connector	2x2 Power module plug	FCI	10084603-001LF
Guide Module Connector	Mechanical guide receptacle	FCI	10084609-111LF

4 Compute Blade Power Distribution

The blade uses a right angle Airmax* VS power connector to provide an input voltage of 12V from the midplane. This connector is in a 2x2 configuration with two 12V and two ground pins assigned. The maximum power consumed by a blade **shall not** exceed 450W.

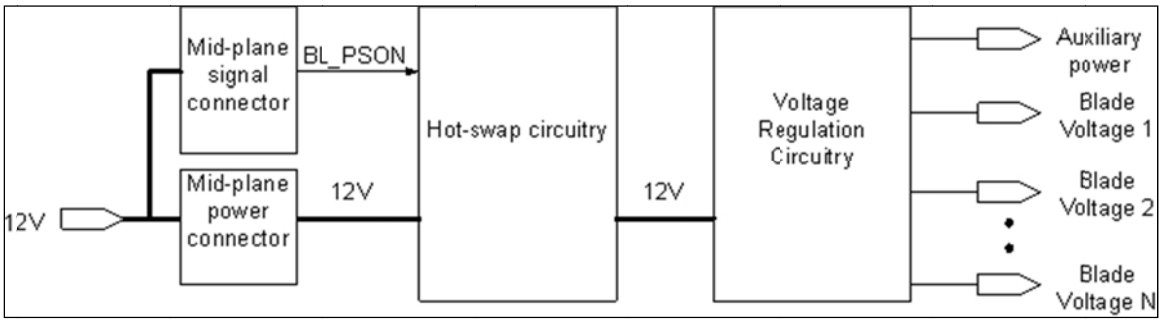
The blades **shall** support being hot plugged into a chassis. This requires that the blade implement a soft start circuit that minimizes the impact to the system power supplies when a blade is added to the system. A short circuit protection circuit **shall** prevent a fault on the blade from adversely affecting the rest of the system.

There is no auxiliary power provided to the blade through the midplane power or signal connectors, so the blade **shall** contain circuitry to generate an auxiliary power rail.

A right angle Airmax VS connector is used for the signals between the blade and midplane. This connector contains a short pin that is used to generate the BL_PSON signal to enable the hot-swap circuitry.

The following figure shows a block diagram of the power distribution model required by the blade.

Figure 4-1: Power Distribution Block Diagram



4.1 Power Sequencing

Once the compute blade is inserted and connected to the midplane connector and the power is reliably available, the 12V supply **shall** power up only the auxiliary management power supply to power the management sub-system on the blade. The blade management sub-system **shall** draw no more than 20W during this time. Once the blade discovery and activation negotiations are completed, the entire blade module should be powered on.

See Chapter 6 for details about power up negotiation and discovery requirements.

Power to the mezzanine **shall** be controlled by the blade such that the mezzanine does not power up before the blade.

4.2 Blade PBA Input Voltage Requirements

The blade PBA **shall** use the following input voltage:

Operating voltage limits:	+12V +5% / -3%

The input voltage is defined at the input connector to the blade. The system **shall** regulate the input voltage within the specified range over dynamic and static operating conditions.

Voltage ripple:	120mV peak to peak

The voltage ripple is defined at the input connector and **shall** be maintained by the system under static loading conditions. Under dynamic conditions the system only needs to meet the input voltage limits.

4.3 Input Voltage Disturbance

The following momentary disturbances **shall not** cause any interruption in the blade operation or any warning indicators on the blade.

Voltage surges:	14.5V for 100msec
Voltage sags:	9V for 2µsec
Under voltage shutdown delay:	MIN/10usec MAX/1msec

Warning and shutdown thresholds recommended. In the event of an under-voltage shutdown condition, the blade system will delay shutting down for the specified shutdown delay duration. Even in this case, the management controller **shall** log an under voltage event.

4.4 Input Current / Power

The current and power consumed by the blade **shall not** exceed the following values.

Input current (Vin = 11.64V):	37.5A continuous / 43.0A peak
Input current slew rating:	< 0.5A/usec
Maximum input power:	450W continuous / 500W peak (Includes 25W for a Mezzanine)

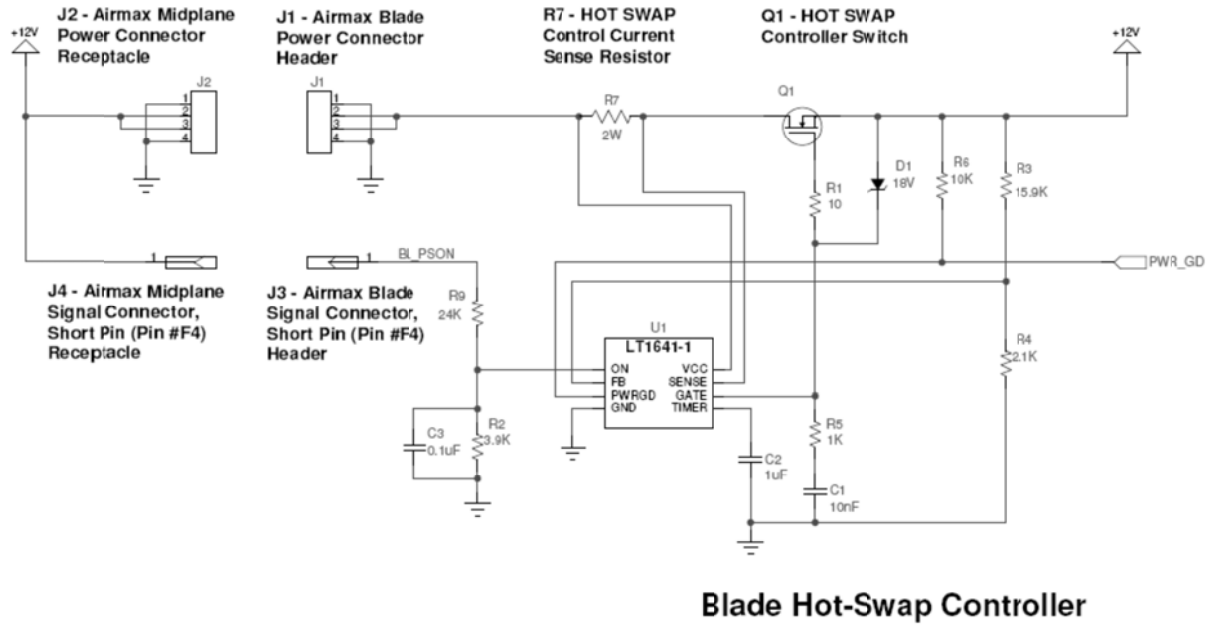
Peak power **shall** be supported for a minimum of 500 msec. This is for supporting peak power caused by hard drives, processors, and memory.

4.5 Hot swap Control

Figure 4-2 shows the blade hot-swap circuit. This circuit is required because the 12V supplied by the midplane is always active. The main purpose of the hot swap circuit is to make sure that the blade 12V rail is energized in a controlled way that protects the blade components as well as other components in the system. This protection circuit consists of several key components, as follows:

- The midplane power connector has ground pins that are slightly longer than the power pins. This ensures that blade circuitry is properly grounded before being energized by 12V.
- The midplane signal connector contains a short pin labeled BL_PSON signal. This signal is the last to connect upon blade insertion and the first to disconnect upon blade removal. The timing of this signal ensures that the blade 12V rail is stable before any blade circuitry is energized and it ensures that the blade circuitry is de-energized before the blade 12V rail is disconnected upon blade removal.
- A current limiting circuit ensures that the blade 12V rail is brought up at a slow enough rate to minimize the impact to the power supplies, or any other system component, when a blade is inserted.

Figure 4-2: Blade Hot-swap Circuit



4.6 Inrush Current

Inrush current **shall** be measured at power on for the blade and during hot insertion of the blade into the 12V power source. Input current slew rates **shall** be measured at power on of the blade, during hot swap, and during system operations under maximum dynamic loading conditions. The inrush current and current slew rates **shall not** exceed the following values.

Peak inrush current:	< 40A
Current slew rate (di/dt):	< 0.5A/usec
Inrush duration	< 100 msec

4.7 Over Current Protection

The blade **shall** provide the following over-current protection for the main 12V rail of the blade.

Over-current: 130% of max rated blade current draw at Vin = 12V (43A*1.3=55.9A)

Over current protection **shall** latch off the blade and disconnect it from the 12V source. The only way to recover from this over-current condition is to remove the blade from the system and re-insert it, presumably after a repair.

The individual voltage regulators that use the main 12V rail as input to generate the voltage rails used by the blade components **shall** also have over-current protection. In the event of a VR over-current condition the blade management controller **shall** log a voltage lost event. The Chassis Manager can request reapplication of blade power if it is believed that the fault has been cleared and the management controller can then enable the blade voltage regulators just as in any normal blade power on sequence. If the fault is still present, then the VR will once again fail with an over current condition and the process will be repeated. The Chassis Manager may choose to end this cycle at any time by leaving the failed blade in a powered down state and setting an alarm about the failure.

Adding a fuse to the +12V supply is allowed, but no specific requirement for fusing the +12V input to the blade is defined.

4.7.1 Power Signals

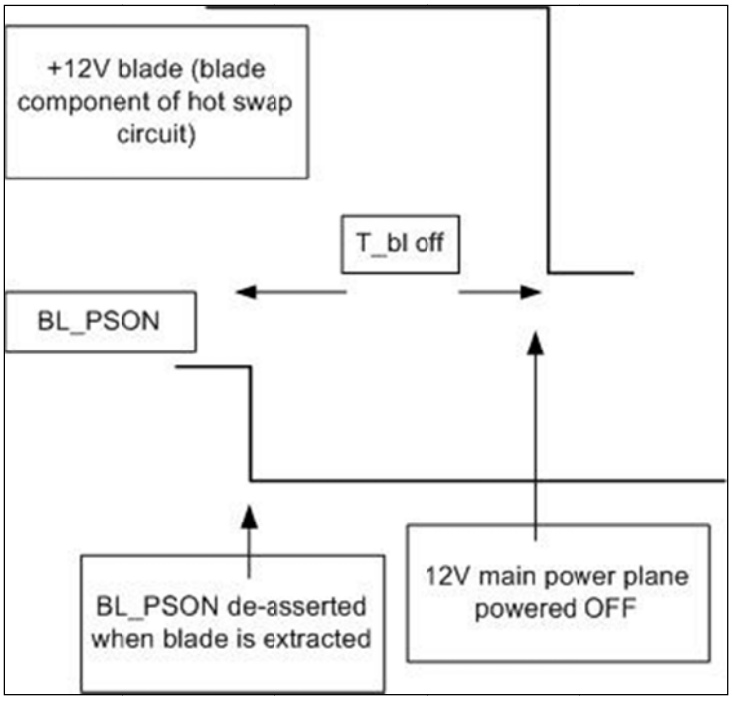
4.7.1.1 BL_PSON

The BL_PSON signal is an input that is used to indicate that the blade has seated properly in the slot. It is connected to a short pin (mate last / break first) for the purpose of supporting hot swap. Once the +12V pin is connected, the auxiliary power **shall** be turned on. The management controller **shall** power up and the BMC **shall** detect this signal ON. Once this signal is detected, the BMC **shall** start the power-up discovery negotiation with the Chassis Manager and turn on the 12V supply to the entire PBA if permitted by the Chassis Manager. Until then all voltage planes, except for the auxiliary voltage, on the blade should remain off. This short pin is connected to the main 12V power rail on the midplane.

Table 4-1: BL_PSON Signal Parameters

Signal Type	Input signal to compute blade (connected to +12V on midplane)	
BL_PSON = Low or Open	Compute Blade removed from midplane	
BL_PSON = High	Compute Blade fully inserted into midplane	
	Min	Max
BL_PSON absolute limits	-0.3V	14.5V
BL_PSON rising edge threshold; blade inserted		10.8V
BL_PSON falling edge threshold; blade removed		8.4V
BL_PSON hysteresis	300mV	
BL_PSON timing T bl off (BL_PSON = Low or Open to blade OFF)		5.0msec
BL_PSON input impedance	10kΩ	

Figure 4-3: BL_PSON Timing



5 Compute Blade Midplane Interface Overview

This section defines the requirements for the midplane interconnection for the high-speed signal interconnects, management signal interconnects and OEM-specific signal interconnects.

5.1 High-Speed Interconnect Requirements

For the purpose of this specification, any electrical signal that runs at a serial data rate of one gigabit per second (1Gbps) or higher over a differential pair of copper traces will be considered as a high-speed interconnect. Also, a pair of transmit and receive signals together (a total of four copper traces) will be defined as a high-speed channel in this specification. A link is the logical aggregation of one or more channels.

Three distinct types of high-speed channels are defined in this specification: Primary, optional Flexi-Channel , and OEM channels. These are shown in Table 5-1. These three different channels provide different platform cost / performance optimization options and different functional capabilities.

Channels 1 through 4 are defined as the primary channels and are reserved for IEEE Std 802.3ap-2007 "Backplane Ethernet" standard compliant Ethernet interconnection. Channels 5 and 6 are reserved for OEM-specific usages and differentiation. These channels are intentionally left as technology-agnostic. Channels 7 through 10 are defined as second primary channels. The second primary offers a redundant path to the primary channels and **shall** be the same technology as the primary channels.

Table 5-1: Interconnect Channels and Classes

Primary Midplane Interconnect Channels (1 through 4) Link 0				OEM-defined Midplane Interconnect Channels (5 and 6)		Second Primary Midplane Interconnect Channels (7 through 10) Link 1			
CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8	CH9	CH10
1X Enet	1X Enet	1X Enet	1X Enet	Rsvd 1X	Rsvd 1X	1x Enet	1x Enet	1x Enet	1x Enet
IEEE Std 802.3ap-2007 "Backplane Ethernet" standard compliant (1000BASE-KX, 10GBASE-KX4 and 10GBASE-KR)				OEM defined Link 2 and 3 as separate, or Link 2 if aggregated		IEEE Std 802.3ap-2007 "Backplane Ethernet" standard compliant (1000BASE-KX, 10GBASE-KX4 and 10GBASE-KR)			

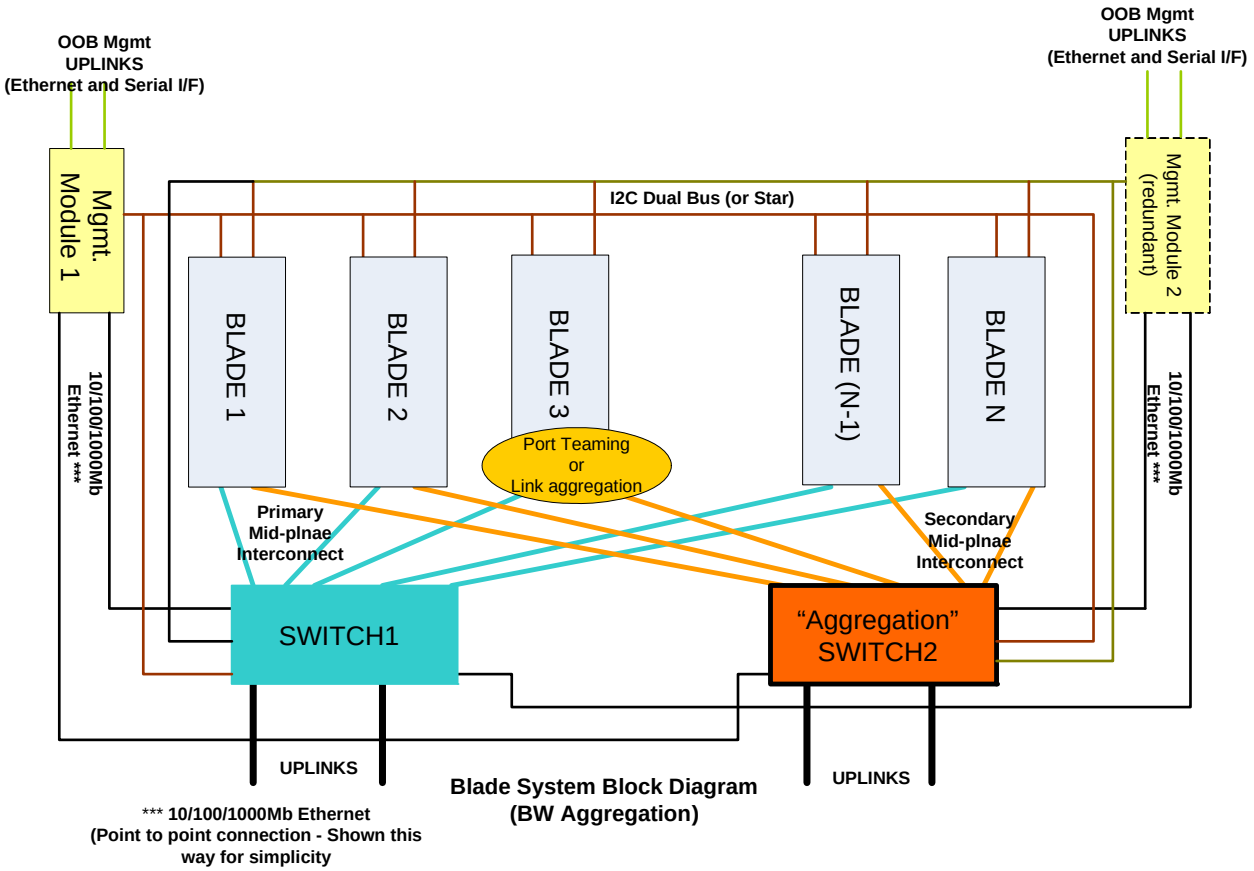
Each SSI Compute Blade **shall** implement IEEE Std 802.3ap-2007 "Backplane Ethernet" standard compliant (1000BASE-KX or 10GBASE-KR) on Primary Ch1 at a minimum. For blades implementing IEEE Std 802.3ap-2007 "Backplane Ethernet" standard compliant 10GBASE-KX4 **shall** use all the Primary channels CH1 – CH4.

Several system interconnection and configuration options are possible using these interconnects.

An aggregated 8x channel configuration is primarily used for high-bandwidth applications. Both the primary and second primary channels implement identical transmission speed across all channels (e.g., Ethernet IEEE10GBASE-KX4). The bandwidth aggregation can be accomplished through the use of "teaming" software or through "link aggregation" (e.g., IEEE802.3ad).

In case of a failure of one or more channels, the system will run at a theoretically lower available bandwidth. Each system designer **shall** ensure that the total sustained bandwidth used in this configuration is less than 100% of the total channel capacity available. Figure 5-1 shows this configuration.

Figure 5-1: Aggregated 8x Channel Configuration



A fully redundant 4x channel configuration is used in enterprise and high-availability systems. The Primary channels and the primary switch may be configured as the active switch, and the Second Primary channels and the corresponding second switch can be used either as "hot" standby or as "active" standby. It is left up to the system designer to choose the specific implementation. When a Primary channel or the primary switch fails, the standby switch becomes active. The inter-switch link provides a fast hardware mechanism to check the status of the two switches. Other failover mechanisms are also deployed.

5.2 Primary Channel Interconnect Requirement

Each blade **shall** reserve the primary channels for use of Ethernet technology and **shall** be compliant with 1000BASE-KX, 10GBASE-KX4 and 10GBASE-KR signaling protocols as defined by IEEE Std 802.3ap-2007 "Backplane Ethernet" standard specifications. This channel **shall** be able to negotiate down to 1Gbps if KX4 or KR is used.

Each SSI compute blade **shall** reserve and allocate the signal pins for all of the channels listed in Table 5-1 on the blade connector that are defined in Table 3-2.

At a minimum, each server blade **shall** implement IEEE 802.3ap defined Ethernet compatible interconnection on primary channel 1 (Ch1). Each blade should provide IEEE Std 802.3ap-2007 "Backplane Ethernet" standard compliant technology on the remaining three primary channels for expandability.

The primary interconnect channels **shall** meet the channel requirements as defined in section 8 of this specification. The primary interconnect design should incorporate test and debug features that will allow a system compliance test to be run. Primary channels **shall** implement an electronic interconnect detection protocol as defined in section 6.7 of this specification.

Several usage cases are possible using the Primary and the second Primary channels (CH1-4 and CH7-10), e.g. in a high availability configuration, two sets of identical channels (IEEE 10BASE-KX4) can be used for both primary and second primary channels, to provide a "failover" capability in case of an unrecoverable error in any of the primary channels.

In this mode, the second primary channels need to be used in similar order. For instance, a 1x redundant mode **shall** use CH1 and CH7.

Other applications can be a higher bandwidth (e.g. 80Gbps per Blade) application in which case all the channels (CH1-4 and CH7-10) **shall** implement IEEE 10BASE-KR technology. Link aggregation or "software teaming" **shall** be implemented to aggregate the links for additional bandwidth.

See the SSI Switch Specifications for details regarding the high-speed signaling and technology support for switches.

5.3 Optional Flexi-Channel Interconnect Requirement

For maximum system design flexibility the optional Flexi-channels are provided through the usage of a mezzanine card.

The primary usage of the optional Flexi-channels is to support a different interconnect technology in addition to the primary Ethernet interconnects. The optional Flexi-channels can use industry-standard interconnect technology such as Ethernet, Infiniband®, Fibre Channel, or for use in the emerging PCI Express* interconnect technology to provide "shared I/O" capabilities in the system. A chassis may provide a redundant or non-redundant implementation of the Flexi-Channels interconnect.

See the SSI Compute Blade Mezzanine Specification for the details of designing SSI-compliant Mezzanine boards.

5.4 OEM Defined Interconnect Channel Requirements

Channels 5 and 6 are reserved for OEM-specific usage.

Each server blade **shall** reserve the pins on the signal connector as defined in section 3.4. OEM-reserved channels should be designed to meet the channel requirements as defined in section 8 of this specification. The OEM interconnect design should incorporate test and debug features that will allow a system compliance test to be run. It is also recommended that OEM channels should implement an electronic interconnect detection protocol as defined in section 6.7 of this specification. If redundancy is required, channel 6 may be implemented as a redundant channel.

5.5 Management Interface Configurations

This specification defines two different models of management connectivity for the management controller.

Each blade in the chassis (server, switch and special purpose blade) **shall** implement a dual (one for redundancy) IPMB, using an industry-standard I2C serial interface to the midplane via the blade signal connector and **shall** allocate the pins as defined in Table 3-1. The I2C interface **shall** provide chassis discovery and configuration on power up or hot plug-in and plug-outs as described in section 6.

Each blade in the system **shall** support a redundant Ethernet interconnection to the midplane to support Ethernet-based management.

5.6 Interconnect Channel Configuration

Except for the OEM Channels 5-6, channels are grouped into four links of x4 channels (A,B,C and D): 1-4, 7-10, 11-14, and 15-18. A fixed set of valid channel set configurations for these groups are defined in Table 5 2. Note that the Primary and Second Primary Channels 1-4 and 7-10 are Ethernet only.

During chassis power on or after blade insertion, a blade goes through a "discovery" sequence to find out the capabilities of the interfaces (e.g. number of channels, data transfer speed for those channels, technology supported on the channels etc). This discovery sequence is described in section 6.7.

For systems that do not use all of the channels of each of the interconnect groups, the management system **shall** configure both ends to the lowest common denominator, as long as compatible interconnect technologies are used on each end.

6 Compute Blade Management Specification

This chapter defines the SSI blade management requirements for compute blades to a level that will allow multi-vendor hardware to interoperate and work simultaneously in the same SSI-compliant chassis.

Management interfaces, protocols, and processes are mandated which place requirements on both SSI Compute Blades and the Chassis Manager. This is not a Chassis Manager specification, and so does not make requirements on the Chassis Manager. However, Chassis Manager behavior and responsibilities are described to provide context for blade management controller behaviors and requirements.

When discussing various management features and behavior, the term "blade" and "blade management controller" may be used interchangeably.

Various aspects of SSI Compute Blade management are either derived from or inspired by the PICMG 3.0 AdvancedTCA (ATCA) specification. This includes various commands and the Operational State machine and hot-swap model implemented by the blades.

6.1 Compute Blade Functional Requirements

The management related functional responsibilities of a compute blade include participation in the following processes:

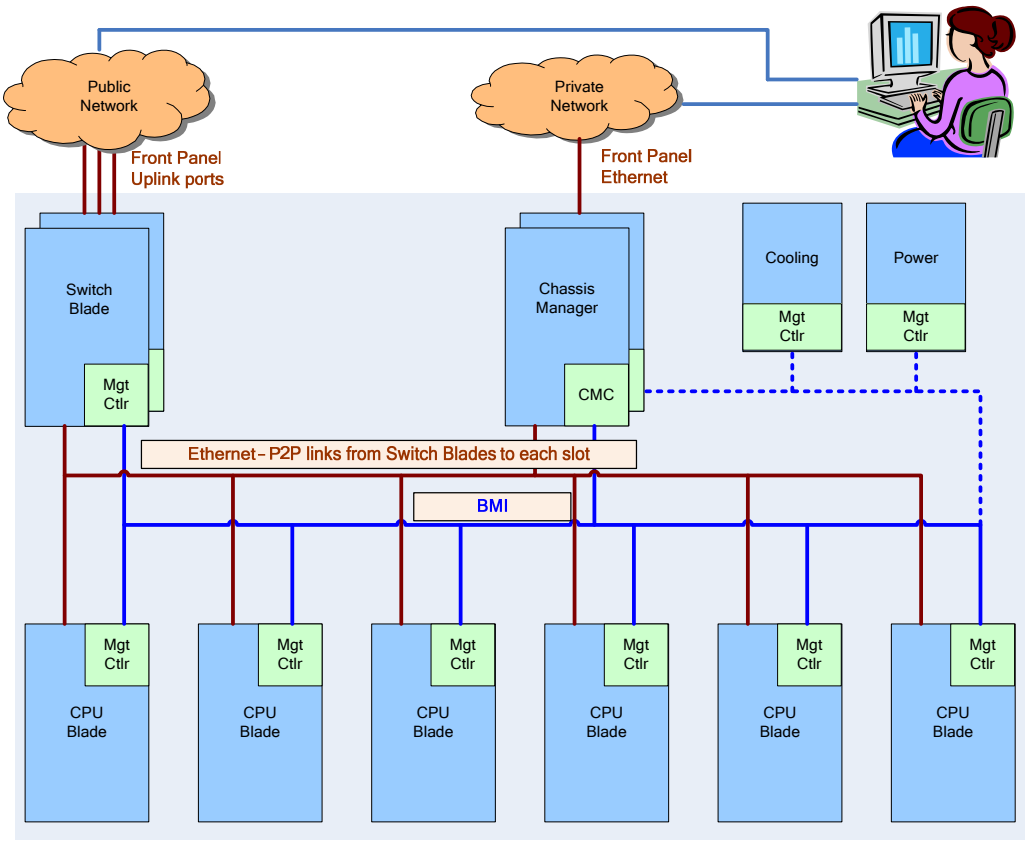
Table 6-1: Compute Blade Management Controller Functional Requirements

No.	Capability	Description
1.	Discovery	A blade **shall** provide support for a presence-detection pin defined in the midplane connector, thus participating in the discovery process. A blade is expected to assert presence (drive the presence detection pin LOW) when seated in the chassis.
2.	Hot insertion/extraction	SSI compute blades are hot swappable and **shall** implement a *Managed Module Operational State Machine* that manages the blade's activation state. The Chassis Manager implements the policies controlling a blade's activation or deactivation state.
3.	Power budgeting	All blades **shall** participate in power budgeting by responding to power negotiation requests from the Chassis Manager and **shall not** power up the blade payload until and unless it is allowed to do so by the Chassis Manager. A blade may or may not receive power permission from the Chassis Manager depending on the prevailing power budget constraints. Blades **shall** operate at or below the Chassis Manager specified power level.

No.	Capability	Description
4.	Power renegotiation	A blade may initiate power renegotiation at any time after activation. Typically, this is done when the power requirement changes due to another event. A blade **shall not** operate at a higher power level than it is permitted to by the Chassis Manager until the renegotiation succeeds. A blade may receive a power command instructing it to operate at a lower level. In such cases, a blade **shall** comply.
5.	Interconnect compatibility verification	Blades implement the functionality required to determine compatible interconnect between the blades and switches. A blade **shall** respond to the Chassis Manager with it's interconnect capabilities and **shall not** enable the midplane Interconnect interconnection until permitted by the Chassis Manager.
6.	Payload power and reset control	A blade **shall** support payload power up and graceful power down as part of the activation/deactivation process. It **shall** also support power cycle, cold/warm reset, and diagnostic interrupt generation on command.
7.	IPMI support	IPMI FRU information, sensors, event generation and logging.
8.	Security	If a payload system interface is implemented, a management controller **shall** allow only certain commands to be issued from the payload that will create traffic on the management interconnect IPMB interface (BMI).
9.	Detection of version support	A blade **shall** report the version of the SSI specification it supports. This information may be used by the Chassis Manager before it issues power up permission.
10.	Management communications support	A blade **shall** implement a redundant chassis management IPMB interface (BMI) for communication to the Chassis Manager as well as Ethernet based communication.
11.	Slot identifier	A blade **shall** configure its BMI IPMB address based on its slot unique identifier.
12.	Cooling management	A blade **shall** forward thermal conditions events to the Chassis Manager to support dynamic cooling in the chassis.

Figure 6-1: Typical Blade System Management Interface Diagram represents a typical blade-based system showing the compute blades, the chassis management controller, and hardware interfaces.

Figure 6-1: Typical Blade System Management Interface Diagram



6.2 Intelligent Platform Management Interface (IPMI) Support

The SSI Compute Blade management infrastructure is based on the *IPMI 2.0, v1.0* specification, including IPMI defined communication interfaces and protocols, sensor/event model, and inventory information description and access model.

A blade's management controller **shall** implement the sub-set of IPMI features and commands as specified by this document. This document has priority with respect to extensions or other variances from the *IPMI 2.0* specification requirements.

While the *IPMI 2.0* specification defines a variety of capabilities for platform management, this specification does not require blade management controllers to implement all of the capabilities defined in the IPMI specification.

6.2.1 IPMI Message Formats

All IPMI command requests and responses defined in this specification **shall** follow the message format defined in the *IPMI v2.0* specification. To uniquely identify requests and responses as SSI defined, the following requirements are placed on the IPMI commands defined in this specification.

- All IPMI requests that are defined in this specification **shall** use the NetFunction value of *Group Extension Request* (2Ch).
- All IPMI responses that are defined in this specification **shall** use the NetFunction value of *Group Extension Response* (2Dh).
- The first byte of the data byte area of both request and response of all IPMI commands defined in this specification **shall** be set to the value of 02h, indicating the command is SSI defined.

6.2.2 Completion Codes

Unless otherwise specified in this specification or in the IPMI specification, completion codes to IPMI commands **shall** be chosen from the standard set defined in section 5.2 of the IPMI specification. OEM completion codes should be avoided.

6.2.3 Inventory Information

The *IPMI v2.0* specification defines Field Replaceable Unit (FRU) Information commands and formats for querying and retrieving Inventory information from an IPMI capable entity. The *Intelligent Platform Management FRU Information Storage Definition v1.0 r1.1* specification defines the common format and use of the FRU Information using IPMI.

To provide for minimum level of inventory information from blades in an interoperable fashion, blades **shall** implement the IPMI defined logical FRU Inventory Device and the FRU Inventory Device commands, as described in section 34 of the *IPMI v2.0* specification.

Inventory information in the *Intelligent Platform Management FRU Information Storage* specification is categorized into different types.

As part of the blade FRU Information, blade management controllers **shall** implement the following FRU areas:

- Common Header
- Board Information Area
- Product Information Area

Implementations may provide additional levels of inventory information beyond what is required by this specification.

The IPMI FRU Inventory Information commands allow for both reads and writes of FRU Inventory information. It is legal for blades to disallow writes to the FRU Inventory information for integrity reasons. When a management controller does allow writes to the FRU Inventory information and external entities do support such capabilities, the entity initiating the writes to the FRU Inventory Information **shall** ensure that the offsets and checksums be updated appropriately.

A blade **shall** implement the following IPMI 2.0 FRU Inventory Device commands:

- *Get FRU Inventory Area Info*
- *Read FRU Data*

A blade may implement the *Write FRU Data* IPMI command.

See the *IPMI 2.0* and *Intelligent Platform Management FRU Information Storage* specifications for additional details on the command formats and FRU Information area formats.

The Inventory information implemented by a blade **shall** be available in any operational state, so that this information can be queried even before the blade's payload power is turned on.

The logical FRU Inventory Device ID for the blade as a whole is 0.

A blade should allow FRU Inventory Area writes for the purpose of supporting customer information such as asset numbers.

6.2.4 Sensors

The *IPMI 2.0* specification defines a sensor model that is implemented by SSI Compute Blades. This specification defines a minimal set of sensors required by an SSI Compute Blade, such as the *Managed Module Operational State* and *Blade Aggregated Thermal* sensors. Blade implementers are encouraged to provide other sensors to describe the state of various blade attributes such as voltages, currents, temperatures, and payload processor status.

In addition, a blade is required to implement a minimum subset of IPMI sensor commands to allow management software get current sensor state. These are:

- *Get Sensor Reading*
- *Get Sensor Hysteresis*
- *Get Sensor Threshold*

IPMI allows for sensors that don't provide readings (event only sensors). Those sensors do not support the above commands. All sensors defined in this specification explicitly state whether they do or do not support readings.

As defined by IPMI, each blade management controller implements a set of generator specific independent sensor namespaces (each with their own sensor number range). A generator can be one of the blade management controller's

IPMI logical units, or payload software (e.g., BIOS). A Generator ID consists of an IPMB address (the blade management controller's) plus LUN (0, 1, or 3), or a Software ID as defined in IPMI 2.0 specification section 5.5.

A chassis unique sensor is defined by the following 3-tuple:

- (BMI address, Generator ID, Sensor #)

6.2.4.1 LUN Addressing

Most implementations will only use blade LUN 0 for non-payload software sensors, however it is possible to run out of sensor number space and so the use of the other 2 LUNS (OEM1 = 1, OEM2 = 3) are allowed. LUNs will show up in 2 contexts. Sensor events specify an event generator which includes the LUN and IPMI Sensor Device command access to sensor state requires that commands be directed to the correct LUN for the specific sensor. Each sensor's SDR also indicates the sensor's associated LUN.

This has the greatest impact on the Chassis Manager implementation, since although a blade management controller need not implement support for LUNs other than 0, the Chassis Manager **shall**.

6.2.5 Sensor Data Records (SDRs)

IPMI Sensor Data Records (SDRs) provide management software with a discovery mechanism for a blade's sensors, management devices, and other objects.

For sensors, SDRs define sensor type and how to interpret sensor readings and events. They also associate sensors with the entities that the sensors are monitoring.

A blade **shall** provide SDRs for all sensors implemented by one of the blade's logical units (LUNs 0, 1, or 3), whether they provide readings or not. SDRs are not required for payload software defined sensors. Sensors that do not provide readings may have *Event Only Sensor Data Records* (type 03) defined for them.

Note that there are additional SDR related requirements for attached modules. See section 6.2.9 for details.

6.2.5.1 Sensor Device commands

A blade **shall** implement the IPMI Sensor Device commands to provide access to its SDR Repository:

- *Get Device SDR Info*
- *Get Device SDR*
- *Reserve Device SDR Repository*

These commands are used instead of the IPMI SDR Storage commands because they support the dynamic behavior associated with a hot plug environment.

The Chassis Manager **shall** use these commands when reading all blade SDRs for management purposes as defined in this specification.

6.2.5.2 Non-Sensor SDRs

A blade **shall** provide a *Management Controller Device Locator Record* (type 12) describing its management controller's capabilities.

A blade **shall** provide a *FRU Device Locator Record* (type 11) for each mezzanine and attached module slot whether it is occupied or not. This record notifies management software of the existence of the slot, provides a user readable name, and indicates the IPMI FRU Id to use to read inventory (FRU Information) describing the attached module.

A blade **shall** provide *Device-Relative Entity Association Records* to provide management software with an understanding of the relationship of sensors to the things they are monitoring.

6.2.5.3 Entity IDs and Instances

SDRs contain Entity ID and Entity Instance fields that associate the sensor or FRU with a specific entity being monitored – e.g., processor, disk, memory module, power module. These fields **shall** be populated using IPMI defined Entity ID values or values defined in this specification.

The Entity ID for an SSI Compute blade is A0h.

For the purposes of this specification, the chassis is considered the equivalent of the IPMI abstraction of *System.* Because of this, all Entity Instances used for blade Entities **shall** be specified as *device-relative*, since they describe entities that are specific to the blade (*device*), not the chassis (*System*). This means that all Entity Instance values **shall** be between 60h and 7Fh inclusive.

6.2.6 Sensor Events

IPMI Sensors can be configured to generate Sensor Events dependent on the state of what is being monitored and the sensor configuration information in the sensor's SDR. Typical events that a blade may generate include over/under voltage, over temperature, critical interrupts, change of operational state, etc. These events include information on the source of the event (sensor generator ID, sensor number), as well as specifics on severity and direction (getting better or worse).

6.2.6.1 Sensor Event Log

Sensor Events are logged into the blade Sensor Event Log (SEL) as Sensor Event Records, providing a blade local history of failures and other occurrences.

A blade's SEL **shall** hold at least 1024 entries.

To support access to the Sensor Events in a blade's SEL, a blade **shall** implement the following IPMI Sensor Event Log commands:

- *Get SEL Info*
- *Get SEL Allocation Info*
- *Reserve SEL*
- *Add SEL Entry*
- *Get SEL Entry*
- *Get SEL Time*
- *Set SEL Time*

6.2.6.1.1 Event Logging Disabled Sensor

A blade management controller **shall** support an instance of the IPMI *Event Logging Disabled* (10h) sensor in order to notify the Chassis Manager of significant SEL conditions such as being cleared and becoming full. The following IPMI defined sensor offsets **shall** be supported:

- 02h - Log Area Reset/Cleared
- 04h - SEL Full

A blade management controller should support the *SEL Almost Full* (05h) offset.

6.2.6.1.2 SEL Timestamps

Sensor Event Records contain a timestamp value that helps management software and administrators understand the relative sequencing of events.

A blade **shall** maintain an internal SEL timestamp clock to use when creating Sensor Event records. A blade may implement a battery backed up real-time clock or other reliable time source in order to initialize its SEL timestamp clock.

If a blade does not implement a battery backed real-time clock, then on insertion, a blade **shall**, at its earliest convenience during the M1 state, request the time from the Chassis Manager by sending it a *Get SEL Time* command, and setting its own SEL timestamp clock from the result.

A blade **shall** implement a *System Event* (12h) sensor with support for offset 05h (Timestamp Clock Synch). This sensor may be event only. When the SEL timestamp clock is set, a blade **shall** generate a pair of events (pre and post clock setting) from this sensor correlating the timestamps for events occurring before and after the new clock value.

If a blade loses the value of its SEL timestamp clock (e.g., due to a management controller reset), the blade **shall** request the time from the Chassis Manager as above.

6.2.6.1.3 System Event Log Policy

The standard IPMI SEL behavior is for the SEL to stop logging event messages once it has filled. This behavior makes it more likely that, in a cascade of faults, the events representing the genesis of the cascade can be found. However, a full or nearly full SEL can cause important events to be not logged

Note: ALL events are sent to the Chassis Manager whether they are locally logged on the blade or not.

Another approach is to implement the SEL as a circular buffer. In this case, when the SEL is full, new events overwrite the oldest events.

This specification allows a blade to implement both algorithms and a Chassis Manager to choose which it prefers. The *Set System Event Log Policy* command provides a mechanism for a Chassis Manager to determine the current policy and to set the policy.

A blade **shall** implement the *Set System Event Log Policy* command. The *Stop Logging on Full* (01h) policy is required. The *Circular Buffer* (02h) policy is optional. If the *Circular Buffer* policy is not implemented, an attempt to set the SEL Policy to that policy will cause a blade to return the completion code *Cannot Execute Command* (D6h).

Table 6-2: Set System Event Log Policy Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h must be used.
	2	Policy control 00h = No change 01h = Stop Logging on Full Policy If this policy is selected, then the SEL will log events until all available space is exhausted and then stop logging. 02h = Circular Buffer Policy If this policy is selected, then if the SEL is full (available space exhausted), then a new event will replace the oldest event (first SEL entry – record ID 0) and the second oldest event becomes the oldest event (now the first SEL entry – record ID 0). All other values reserved
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h must be used.
	3	Current Policy (after request policy control executed) 1h = Stop Logging on Full Policy 2h = Circular Buffer Policy

6.2.6.2 Event Forwarding

In addition to maintaining a Sensor Event Log, a blade **shall** by default forward all sensor events it generates to the Chassis Manager over the BMI for consolidation into chassis state. This **shall** be done whether or not events are able to be logged into the SEL (say, due to a SEL full condition or if SEL logging is disabled).

A blade is an IPMI Event Generator, and as such, it **shall** implement the IPMI *Set Event Receiver* command which identifies the BMI target to which to forward events. A blade **shall** set its initial Event Receiver BMI address to 20h.

The *Platform Event Message* command that IPMI defines as the command used to forward sensor events does not fully support identifying the generator of an event other than the blade management controller. Because of this, this specification defines an *SSI Event Message* command.

A blade **shall** use the *SSI Event Message* command when delivering sensor events generated by blade software (BIOS and/or applications). A blade may use the *Platform Event Message* or *SSI Event Message* commands for events generated by a blade management controller logical unit.

A Chassis Manager **shall** accept both *Platform Event Message* and *SSI Event Message* commands.

The following table defines the *SSI Event Message* command. It is almost exactly like the IPMI *Platform Event Message* command over the System Interface (see the *IPMI 2.0* specification table 29-4), but with the addition of the SSI Identifiers and extended Generator ID field.

Table 6-3: SSI Event Message Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	2-3	Generator ID Byte 1 [7:1] - Blade 7 bit BMI I^2C address or System Software ID [0] - Generator type 0 = Blade management controller 1 = System software Byte 2 [7:2] - Reserved [1:0] - LUN if Generator type = Blade management controller, else 0
	4	EvMRev
	5	Sensor Type
	6	Sensor #
	7	Event Dir | Event Type
	8	Event Data 1
	9	Event Data 2
	10	Event Data 3
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.

6.2.7 Watchdog Timer

If a blade management controller implements a Payload management communications interface (system interface), then the following requirements **shall** hold:

A blade **shall** implement the IPMI Watchdog related commands and functionality. These are:

- *Get Watchdog Timer*
- *Set Watchdog Timer*
- *Reset Watchdog Timer*

A blade **shall** implement an IPMI *Watchdog2* sensor and provide its associated SDR. Assertion events **shall** be enabled for all supported action offsets. The sensor **shall** provide readings.

The Watchdog functionality is only operational in the *Active* (M4) or *Deactivation Requested* (M5) operational states. The Watchdog timer **shall** be reset when the blade transitions to a state other than M4 and M5.

A blade **shall** implement the following timer expiration actions with the indicated behaviors:

- *Hard Reset* - Payload is reset, no change of operational state
- *Power Cycle* - Payload is power cycled, no change of Operational State
- *Power Down* - Payload power is turned off with a forced transition to the *Deactivation in Progress* (M6) state

As with *IPMI 2.0*, the Watchdog pre-timeout actions are optional.

6.2.8 GUIDs

A *Globally Unique Identifier* identifies its owner uniquely and provides management software with a way to track hardware inventory, potentially even across chassis.

A blade **shall** implement a Device GUID and **shall** support the IPMI *Get Device GUID* command as defined by IPMI.

A Chassis Manager should read a blade's Device GUID and use it to tag events from the blade. This allows it to properly associate sensor events with blades even if they change slots in the chassis.

A blade may also support a System GUID representing the blade as a whole. A System GUID is preferentially used by some IPMI functionality (e.g., Platform Event Traps).

6.2.9 Boot Option Support

The IPMI Boot Options feature allows the payload boot process to be controlled, including what device to boot from, what caused the boot, and loader/OS boot parameters.

If a compute blade supports a payload system management interface, then the following requirements **shall** be followed.

A compute blade management controller **shall** support the following IPMI Boot Options commands:

- Get System Restart Cause
- Set System Boot Options
- Get System Boot Options

A compute blade management controller **shall** support all IPMI-defined Boot Options.

In addition, the blade **shall** support an SSI OEM Boot Option parameter used to extend the IPMI boot selector (in IPMI Boot Option parameter *Boot Flags* (5), data 2).

Table 6-4: SSI Boot Option Parameters

Parameter Name	#	Description
OEM Parameter Block Size Table	120	See section 7.15.1.
Reserved	121	Reserved for future SSI definition.
Reserved	122	Reserved for future SSI definition.
Reserved	123	Reserved for future SSI definition.
Reserved	124	Reserved for future SSI definition.
Boot Order Table	125	See section 7.15.7.
SSI Boot Device Selector	126	data 1 – extensions to the IPMI boot device selector enumeration (parameter Boot Flags(5) data byte2). Read/write 0 - Use standard IPMI boot device selector in parameter 5 1 - Force configuration boot 2 - Force boot to local default hard-drive 3 - Force boot to external default hard-drive Other values reserved
Slot Configuration Table	127	See section 7.15.3.

6.2.10 Attached Module Support

SSI Compute Blades may implement support for application or function specific mezzanines or adapters. Differing amounts of management may be implemented for such modules. However, the common model is that a blade management controller proxies for the entity management information, presenting it through its own interfaces.

If a blade supports mezzanines or other attached modules, the blade **shall** provide IPMI FRU information for the modules via IPMI FRU Information commands. Each such module **shall** have a unique FRU Inventory Device ID associated with it and the blade **shall** implement the IPMI defined *FRU Controller Device Locator Record* (SDR type 11h) in order for the Chassis Manager to locate the FRU Information.

6.2.10.1 Non-Hot Swappable Modules

A non-hot swappable module is one that cannot be attached or removed while the blade is inserted into the chassis. Every such module attaches at a "site". A non-hot swappable module site **shall** be represented by a sensor of type *Slot/Connector (21h)*. Minimally, offset 2 (*Device Installed/Attached*) **shall** be implemented. This sensor serves as the Entity Presence sensor for the module. An SDR **shall** be provided for this sensor that associates it with module entity. See *IPMI 2.0, Chapter 40. Handling Sensor Associations* for background on IPMI Entities and Entity Presence.

If there are sensors implemented on the module, the blade **shall** represent those sensors in its own IPMI sensor name space, provide sensor access via IPMI Sensor Device commands, and provide event generation.

The blade **shall** provide SDRs for all module sensors and their entity information **shall** correctly identify the module as the source. As indicated in the *IPMI 2.0* Specification, if the module's Entity Presence or equivalent sensor indicates that the module is not present, these sensors should be ignored by the Chassis Manager and other management software.

A blade **shall** include any module power requirements as part of the values it advertises during the power negotiation process.

Note that these requirements are irrespective of whether the attached module hosts an intelligent management controller or not, although, obviously, their implementation will be very different in the two cases.

6.2.10.2 Hot Swappable Modules

Management support for hot swappable modules is not as yet covered by this specification. Blades may implement hot swappable storage (hard-drives), but management as defined by this specification is not involved.

6.3 Management Interfaces

An SSI Compute Blade supports management interfaces to the Chassis Manager. These interfaces go through chassis slot connector defined signals. An SSI Compute Blade supports the chassis base management interconnect (BMI) and redundant Ethernet management interfaces via the Primary Ethernet channels.

An SSI Compute Blade may support a management interface to its payload.

6.3.1 IPMI Messaging Support

A blade **shall** support the following IPMI Messaging Support commands:

- *Get Channel Info*
- *Get Channel Access*
- *Set Channel Access*

These commands allow the Chassis Manager to discover the capabilities of the blade's IPMI messaging implementation including what IPMI channels exist and what their types are.

6.3.2 Slot Identifier and Presence Lines

Each SSI Compute Blade chassis slot is assigned a chassis unique six-bit slot identifier. This identifier is encoded into a set of signals available via the slot's

chassis midplane connector. The identifier is used by a blade management controller to generate a specific address that allows the Chassis Manager and other entities to determine the blade's physical location.

A six-bit slot identifier provides a maximum of 64 blades that can be addressed on the BMI.

Slot Identifier (5:0)	Description
000000b to 111111b	Allowed for assignment to module management controllers

Because a blade is a hot-pluggable entity, it cannot have a statically defined address and so each blade needs to generate an IPMB address dynamically. A chassis unique BMI address is generated based on the blade's slot identifier. Each blade **shall** support the slot identifier to acknowledge/identify itself for overall chassis management.

The BMI IPMB address of the chassis management controller (CMC) (the part of the hardware that provides the Chassis Manager functionality) is pre-defined to be 20h so that blade management firmware need not implement any discovery mechanism for the Chassis Manager.

The first Slot Identifier shall (000000b) correspond with the first presence line (BL_PRES_1), the second (000001b) shall correspond with the second, and so on. The slot identifier, presence line, and physical slot labeling shall all correspond. The table below shows the sequence.

External Slot Label	Presence Line to CMM	Slot ID Pins
Blade 1	BL_PRES_1	000000b
Blade 2	BL_PRES_2	000001b
..Blade N	…BL_PRES_N	…N-1 b

6.3.3 Chassis Base Management Interconnect

The *Intelligent Platform Management Bus* (IPMB) specification defines a protocol for the transmission of IPMI messages over an I^2C bus. The SSI Compute Blade specification defines a pair of IPMB busses on a slot's chassis midplane connector. This pair of busses is called the base management interconnect (BMI). The individual busses, IPMB-A and IPMB-B, are named with letters to avoid confusion with management controller IPMI channel numbers. Both busses are active during normal management communication.

A blade's BMI interface forms the basis for out-of-band management between the blade's management controller and the Chassis Manager. Except where

explicitly noted, all Chassis Manager to blade management communication defined in this specification can be carried over this interface.

The BMI interface **shall** always be available with management or standby power. This way, communication with a blade's management controller **shall** be possible even when the payload power of a blade is turned off.

A blade's BMI is intended to be the interface over which critical chassis functionality such as discovery (e.g., insertion/extraction), power negotiation, and recovery operations such as payload power cycle, reset, and reboot are performed. This requires higher levels of reliability and availability under all conditions. To meet these functionalities, the following requirements are placed on blades that connect to the BMI.

- A blade **shall** only connect its management controller to the BMI.
- A blade **shall** implement redundant IPMB interfaces (bus IPMB-A and bus IPMB-B) to the BMI, for communication with the Chassis Manager.
- A blade management controller's BMI bus interfaces **shall** be multi-master and master/slave capable.
- A blade management controller **shall** aggregate both BMI busses as a single IPMI IPMB channel (channel 0).
- A blade management controller **shall** be able to simultaneously transmit or receive messages over either of the BMI busses.
- Responses **shall** be sent on the same bus over which the request was received.
- One of the BMI busses **shall** be considered the "active" bus. This is configurable (see *Module BMI Control* command below).
- Request messages sent by the managed module management controller **shall** be sent via the "active" bus. The exception would be during a redundant "takeover" where the failover CMM would instruct the BMC to change the active bus.
- The "active" bus designation simply requires all asynchronous messages (such as events) sent by the blade to be sent out of the active bus. Both busses still need to be able to receive and process messages.
- A blade management controller **shall** implement the capability to disable itself from an individual IPMB bus if it believes that it is causing faults on that bus. It may periodically re-enable the interface to see if the failure persists.
- A blade management controller **shall** support a data rate of at least 100 kbps over the IPMB interface.
- A blade management controller may use higher data rates up to 400Kbps. Electrical design care should be taken to achieve those higher data rates.
- A blade's management controller BMI bus interfaces **shall** inter-operate correctly with other bus masters using data/clock rates up to 400kbps, potentially clock stretching as necessary.

It is recommended that additional functionality (e.g. text console or KVM redirection) requiring more bandwidth than that provided by the BMI be carried over IP-capable transport.

Note that a blade management controller may implement other IPMBs internally on the blade. Those IPMBs are not subject to any of the requirements above.

6.3.3.1 BMI IPMB Address

An SSI Compute Blade's BMI IPMB Address is derived from the blade's Slot Identifier. Each BMI bus interface, A and B, is set to the same IPMB address.

The following formula **shall** be by a blade to generate the IPMB address based off the Slot Identifier.

IPMB address generation from Slot Identifier
IPMB address = 30h + (slot Identifier X 2)

Slot Identifier	Resultant IPMB address range
000000b to 111111b	30h to AEh

6.3.3.2 Module BMI Control Command

The *Module BMI Control* command allows the Chassis Manager to query and control a blade's use of the BMI by allowing one or the other BMI IPMBs to be administratively disabled. A bus can be under blade control, either enabled or disabled because of detected errors, or the bus can be under administrative control and be disabled. It is not possible to administratively enable a bus that the blade has disabled.

An administratively disabled bus is only disabled for sending asynchronous messages (such as events) by the blade. However, the blade's management controller **shall** continue to receive and respond to messages. This is not intended to be a normal operating mode but rather a tool for debugging and handling I2C bus problems. The bus can be re-enabled by the CMM turning on the proper bit as indicated in Table 6-5.

The command also allows the current management and error state of each bus to be queried.

A blade **shall** implement the *Module BMI Control* command.

Table 6-5: Module BMI Control Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	2	IPMB control [7:6] – Reserved [5:4] - Active Bus Control 0h = No change 1h = IPMB-A is active 2h = IPMB-B is active [3:2] – BMI IPMB-B Control 0h = No change 1h = Blade controls 2h = Administrative disable [1:0] – BMI IPMB-A Control 0h = No change 1h = Blade controls 2h = Administrative disable
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	3	IPMB State [7:5] - Reserved [4] - Active Bus 0b = IPMB-A is active 1b = IPMB-B is active [3] – BMI IPMB-B State 0b = No blade detected errors 1b = Blade bus disable [2] – BMI IPMB-B Control State 0b = Blade controls 1b = Administrative disable [1] – BMI IPMB-A State 0b = No blade detected errors 1b = Blade bus disable [0] – BMI IPMB-A Control State 0b = Blade controls 1b = Administrative disable

6.3.3.3 BMI Security

All IPMI commands specified here are categorized into two groups: Informative and Administrative. Management controllers **shall** implement the following rules to prevent spurious messages on the BMI.

- Each management controller **shall** process administrative commands it receives only if the IPMI command has 20h as the source (requestor) IPMB address.

- Each management controller **shall** ensure that it uses its Slot Identifier-based IPMB address as the source (requestor) address when sending any message on the BMI.

Note: Only the CMC that is running the Chassis Manager (active CMC in case of redundant operation), is allowed to send IPMB messages with source (requestor) IPMB address of 20h.

6.3.3.4 Physical Isolation of Blades on the BMI

A blade management controller **shall** disable a BMI IPMB interface if it thinks that it has failed. It should periodically re-enable a failed interface and test it to determine if the failure has been repaired. For example, the failure could have been caused by another blade which has since been replaced.

6.3.3.5 Hot Swap Requirements

To prevent glitches on the BMI when a blade is being inserted or extracted from the chassis, the initial state of the IPMB clock and data signals on an management controller **shall** be disabled from connecting to the backplane until standby power is stable and the management controller is able to read the slot identifier and generate and set its BMI address.

A blade management controller **shall** ignore received transients of less than 50 ns duration. A management controller **shall** be able to detect and recover the I^2C from a Busy Timeout condition. Implementation of this feature is vendor-specific, but the mechanism used **shall** be compatible with the I^2C protocols.

A transmitting blade management controller should use the dummy message to restore the bus from a "Busy Timeout Condition". The busy timeout occurs if the SDA and SCL lines stay high for a period of time greater than the minimum "time-out waiting for bus free" period (T2, as specified in section 4 of the IPMB specification), before a stop condition occurs.

A dummy message with a known reserved address is shown in Table 6-6.

Table 6-6: Dummy Message Format

Start	Write Destination address = EEh								1=ACK	Stop
	CLK 1	CLK 2	CLK 3	CLK 4	CLK5	CLK 6	CLK 7	CLK 8		

As long as the blade management controller transmits to a known reserved address, no other device will try to respond. Regardless of which device transmitted the dummy message, all synchronous receiver/transmitter (of management controllers) on the bus **shall** clear their Busy states once the stop condition is received. At a minimum, a management controller **shall** meet all timing requirements in IPMB specifications.

A blade management controller should use separate I^2C buffers for the dual BMI I^2C busses, to help localize and isolate the busses in case of a fault on a

bus. PBAs and modules **shall** provide a 10k ± 5% pull-up resistor from each of the Hardware Address signals to the management controller's input voltage supply connection.

6.3.4 Ethernet Management

The SSI chassis supports blade management over Ethernet as well as over the BMI. Ethernet provides a higher performance interface and so allows for a richer set of management features to be implemented, such as IPMI Serial Over LAN (SOL), Keyboard/Video/Mouse (KVM) redirection, and virtualized storage media.

A blade may be used in an SSI chassis that implements a range of management models. The Ethernet management traffic may be intra-chassis and secure allowing authentication and privacy free operation, or the model may expose management traffic external to the chassis, requiring blade management to implement security. As a result, a blade **shall** support a range of features which are configured by the Chassis Manager as appropriate.

A blade **shall** support 2 management LAN channels to provide redundancy.

Implementation of management capability **shall** be over Primary Interconnect Ethernet channel 1 and Second Primary Interconnect channel 7. Blade management may share these channels with payload traffic.

Blades implement management over Ethernet supporting the following protocol features for each channel:

- IPMI 2.0 RMCP+ messaging
- 802.1q VLAN messaging
- Static IP address configuration
- ARP
- A management unique MAC address (not shared with a payload interface)
- IPMI Sessions – support for a minimum of 2 simultaneous sessions
- Authentication and encryption – IPMI RMCP+ standard

A blade may support DHCP.

Ethernet-based management **shall** be available irrespective of the blade operational state (MState) and the state of payload power.

The two redundant management Ethernet channels **shall** be represented on the blade management controller as a single IPMI channel (channel 1).

A blade **shall** support the *Get Channel Cipher Suites* command to provide information on LAN Channel Authentication, Integrity, and Privacy algorithm support.

6.3.4.1 Configuration

The default state for a blade's IPMI LAN channel(s) on blade insertion or chassis power-on **shall** be *disabled*. It is the Chassis Manager's responsibility to configure the blade's LAN channels via the BMI during the blade activation process. After configuration is complete, the Chassis Manager will use the *Set Channel Access* command to enable the channels.

A blade **shall** store its LAN Configuration Parameters persistently, but the default channel state requirement prevents the blade from acting on these values before the Chassis Manager has the opportunity of validating their values or configuring new ones. This allows the Chassis Manager to implement different addressing or configuration schemes that are dependent on blade Slot Identifier or other attributes.

A blade **shall** accept LAN configuration commands and allow LAN channel enabling in any operational state. A blade **shall** support IPMI LAN channel configuration via the IPMI *Get LAN Channel Configuration Parameters* and *Set LAN Channel Configuration Parameters* commands. When changing the values below, first the "Set In Progress" parameter **shall** be set to 01. Then, after the the other parameters are set, the "Set In Progress" parameter **shall** be set to 00.

The transition of the "Set In Progress" parameter from 01 (set in progress) to 00 (set complete) **shall** trigger the actual configuration of the network based on the LAN Configuration Parameters. Additionally, any time the "Set In Progress" parameter is set to "00" (even if it was already at "00"), this **shall** also trigger the same configuration. The following parameters **shall** be supported:

Table 6-7: LAN Configuration Parameters

Parameter Name	Number
Set In Progress	0
IP Address	3
IP Address Source	4
MAC Address	5
Subnet Mask	6
IPv4 Header Parameters	7
Default Gateway Address	12
Number of Destinations	17
802.1q b ID	20
802.1q VLAN Priority	21
SSI Second Primary Ethernet MAC Address	200 (OEM)
SSI Link Control	201 (OEM)
CMM IP Address	202 (OEM)

The *MAC Address* (5) parameter provides the MAC address for the Ethernet link associated with Primary Interconnect channel 1. All other IPMI defined LAN parameters apply to the IPMI LAN channel as a whole.

The SSI-defined LAN parameters are specified by the following table.

Table 6-8: SSI Compute Blade LAN Parameters

Parameter Name	#	Description
SSI Second Primary Ethernet MAC Address	200	data 1:6 - MAC address for the Ethernet link associated with Second Primary Interconnect channel 7. MS-byte first Can be Read Only or Read/Write as per IPMI spec on other MAC addresses.
SSI Link Control	201	data 1 – Current state of the underlying Ethernet links. Read Only [7:2] - reserved [1] - Second Primary Interconnect Channel 7 state 0 = disabled, 1 = enabled [0] - Primary Interconnect Channel 1 state 0 = disabled, 1 = enabled
CMM IP Address	202	data 1:4 – IP address of CMM. CMM shall write the IP address that the Blade shall use to communicate with it. The blade may use this for detecting LAN communications failures to the CMM

6.3.4.2 Event Forwarding over LAN (PET)

A blade may support sending sensor events to the Chassis Manager over IPMI LAN using the standard IPMI Platform Event Filtering (PEF) and Platform Event Trap (PET) features. If this is supported, the following requirements **shall** be implemented.

A blade **shall** support at least one LAN Alert Destination. This implies support for the following additional LAN Configuration Parameters:

Table 6-9: LAN Configuration Parameters

Parameter Name	Number
Community String	16
Destination Type	18
Destination Addresses	19
Destination Address VLAN Tags	25

PET packets are sent as UDP datagrams and so are effectively unreliable unless the LAN Alert Destination Type is configured as *Acknowledged*. In that case, the blade management controller will keep resending the PET until the Chassis Manager acknowledges it by sending the blade a *PET Acknowledge* command.

A blade **shall** support *Acknowledged* LAN Alert Destination functionality. The Chassis Manager may configure the destination as such to guarantee event delivery.

An alternate strategy is for the Chassis Manager to periodically check the blade System Event Log to see if events have occurred for which it has not seen PET packets.

6.3.4.3 Sessions and Authentication

Some SSI chassis management configurations allow for management traffic that is visible outside the chassis environment or to blade payload software. To support these deployment models, blades **shall** implement LAN channel security. This means support for IPMI RMCP+ LAN sessions, authentication, and encryption.

Implementation of IPMI LAN-based sessions requires IPMI support for users and passwords and their associated commands.

A blade **shall** support at least 2 users.

A blade **shall** support at least 2 simultaneous IPMI sessions. Session-based features may have additional session requirements.

A blade **shall** support the following IPMI commands:

- *Get Session Challenge*
- *Get Channel Authentication Capabilities*
- *Activate Session*
- *Set Session Privilege Level*
- *Close Session*
- *Get Session Info*
- *Set User Access*
- *Get User Access*
- *Get User name*
- *Set User Name*
- *Set User Password*
- *Get Channel Cipher Suites*
- *Set Channel Security Keys*

Minimally, a blade **shall** support the IPMI cipher suite algorithms:

Table 6-10: Cipher Suite Algorithms

Type	Algorithm
Authentication	RAKP-none
	RAKP-HMAC-SHA1
	RAKP-HMAC-MD5
Integrity	None
	HMAC-SHA1-96
	HMAC-MD5-128
Confidentiality	None
	AES-CBC-128
	xRC4-128

6.3.5 Payload Interface

This specification does not require a management controller implementation to provide a management messaging interface to its payload (or even have an intelligent payload).

If a payload messaging interface is provided, it **shall** be an IPMI-compliant system interface. In addition, the IPMI Watchdog Timer and Boot Option related commands **shall** be implemented to support payload boot and operational software failure recovery. See sections 6.2.7 and 6.2.9.

Additionally, if a payload system interface is provided, the blade management controller **shall** support the *SendMessage* and *GetMessage* IPMI commands and bridging between the payload interface and the BMI to allow the payload to communicate with the Chassis Manager. See section 6.3.3.3 for security requirements relative to payload-generated commands.

Note that in some compute blade implementations, the command and response message payload data size may be as large as 131 bytes (not counting interface specific command overhead bytes). See section 7.15.1, Blade OEM Parameter Access, for an example. Payload System Interface clients **shall** be capable of reading and writing system interface messages of at least 131 bytes plus interface specific overhead bytes.

6.3.6 Latent Fault Detection

In order to avoid undetected faults (latent faults) of the Management interfaces, the Chassis Manager will periodically communicate to the blade over each active interface, e.g., BMI and Ethernet/RMCP+. This can be as part of normal management activities or a dedicated activity. The maximum period between communications on an interface is 10 seconds.

6.4 Compute Blade Discovery

The act of plugging in a compute blade causes the slot's *blade present* signal to be asserted. This is visible to the Chassis Manager which then queries the blade for information about itself, updating the Chassis Manager's current state about the slot.

The commands described in the following sections **shall** be supported to enable management subsystems to discover additional information about blades.

6.4.1.1 Get Address Information Command

All managed entities in a chassis have a set of names, attributes, and addresses. Those entities directly on the BMI have both a Slot ID and an IPMB address. In addition, they have a Physical Slot Number that relates to the markings on the chassis. Note that some entities may have a Physical Slot Number, but no Slot ID or IPMB address if they are managed by a different entity (consider a fan-pack managed by a cooling management module or the auxiliary blade of a double-wide blade).

The *Get Address Information* command provides a mechanism for discovery or enumeration of the managed entities in a chassis. A blade will only accept *Get Address Information* queries about itself and its managed entities. The Chassis Manager **shall** accept queries for any chassis slot.

This command takes a variable number of request parameters that allows querying of addressing information.

Table 6-11: Get Address Information Command Parameters

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI defined extension command. A value of 02h **shall** be used.
	(2)	FRU Device ID. This is an optional field that identifies the individual attached module on the module whose information is being requested. When this field is not present, the command returns the addressing information for the management controller at FRU Device ID 0. This field is ignored when *Address Key Type* is Physical slot number.
	(3)	Address Key Type. This field is optional and identifies the type of address that is being requested in Address Key field. When this field is not present, the command returns the addressing information for the management controller identified by FRU Device ID 0. 00h = Physical slot number 01h = BMI address All other values are reserved.
	(4)	Address Key.

Field	Byte	Content / Purpose
		This field is required if Address Key Type field is present. This holds the target BMI address or physical slot number, depending on the content of the Address Key Type field.
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI defined extension command. A value of 02h **shall** be used.
	3	FRU Device ID. This is the FRU Device ID for the module that matched the request.
	4	Physical Slots Occupied [7:6] - Slot count (0 = 1 slot) [5:0] - First physical slot number This is target's slot number. If unknown or not applicable, the value is 3Fh.
	5	BMI Address. Indicates the target's BMI address.
	6	Site Type. 01h = Compute Blade 02h = Ethernet switch 03h = Fibre channel switch 04h = PCI Express switch 05h = Infiniband® switch 06h = Dedicated Chassis Management Module 07h = Chassis configuration information 08h = Power unit module 09h = Power supply module 0Ah = Fan tray / cooling module 0Bh = Alarm board 0Ch = Mezzanine board 0Dh = Memory Module (e.g., DIMM) 0Eh = Reserved 0Fh = Generic module 10h to FFh are reserved 00h is reserved.

6.4.1.2 Get Compute Blade Properties Command

The *Get Compute Blade Properties* command enables the Chassis Manager and system software to query the maximum IPMI FRU Device ID number of modules (other chassis modules, mezzanines, etc) managed by a blade. This helps to limit the effort needed to enumerate the modules on a blade.

The command also provides the SSI Extension version number supported by the blade and the number of physical slots occupied by the blade.

Table 6-12: Get Compute Blade Properties Command Parameters

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI defined extension command. A value of 02h **shall** be used.
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	3	SSI Extension Version. Indicates the version of the SSI extensions implemented by the management controller. [7:4] = BCD encoded major version [3:0] = BCD encoded minor version This version of the specification defines version 10h of the SSI extensions. The value of 00h is reserved.
	4	Slot count This is the number of physical slots taken up by the module. It is assumed that these have contiguously increasing slot IDs.
	5	Max FRU Device ID. This is the largest valid FRU Device ID supported by the blade.

6.4.1.3 Blade Startup Sequencing

On chassis power-on, all of the blades' management controllers will be initializing simultaneously and attempting to talk to the Chassis Manager, sending events. This could overwhelm a Chassis Manager, causing command response timeouts or other issues.

A blade management controller should implement a startup delay that is tied to its Slot ID. After initialization, a blade should delay 1 times its Slot ID seconds before starting its conversation with the Chassis Manager. However, it should accept commands from the Chassis Manager as soon as it is able to.

6.5 Managed Module Operational State Management

The operational state of a blade reflects the blade's state with respect to being integrated into a chassis' operation. Operational state, in this context, does not cover whether the payload is operating correctly.

- An *inactive* or *de-activated* blade is only drawing power for its management subsystem and has no chassis resources (power, interconnect, cooling) allocated to it.
- An *activated* blade is in its normal operational state. It has its required chassis resources allocated and its payload is powered and running.

- A blade that is transitioning between *inactive* and *activated* is either *activating* or *deactivating*.

Operational state management is one of the most critical management functionalities in a blade environment. It forms the basis for managing blade insertion, extraction, power negotiation and sequencing, and interconnect compatibility verification in a chassis. Operational state management is achieved through communication of IPMI messages from the Chassis Manager to the blade and Managed Module Operational State sensor events from the blade to the Chassis Manager.

Blades and Chassis Managers **shall** implement Managed Module Operational State management as described in this section.

6.5.1 Managed Element States and Transitions

This specification defines 8 distinct blade operational states as well as the possible transitions between states. Each blade **shall** implement this state machine for itself. In addition, the Chassis Manager keeps track of each blade state transition, and takes actions based on them. Hence, it is necessary that both the Chassis Manager and the blade are always in sync with the blade's operational state. State transitions are communicated by the blade to the Chassis Manager over the BMI or potentially another management interface, using IPMI event messages.

Transitions from one state to another can be triggered by either IPMI commands sent by the Chassis Manager, or by some action or condition on the blade itself. Examples of such conditions include front panel activation request button presses and payload initiated power-down.

A blade **shall** implement the Operational State Machine as defined by Figure 6-2 and the state definitions in section 6.5.1.2.

6.5.1.1 Insertion and Extraction Criteria

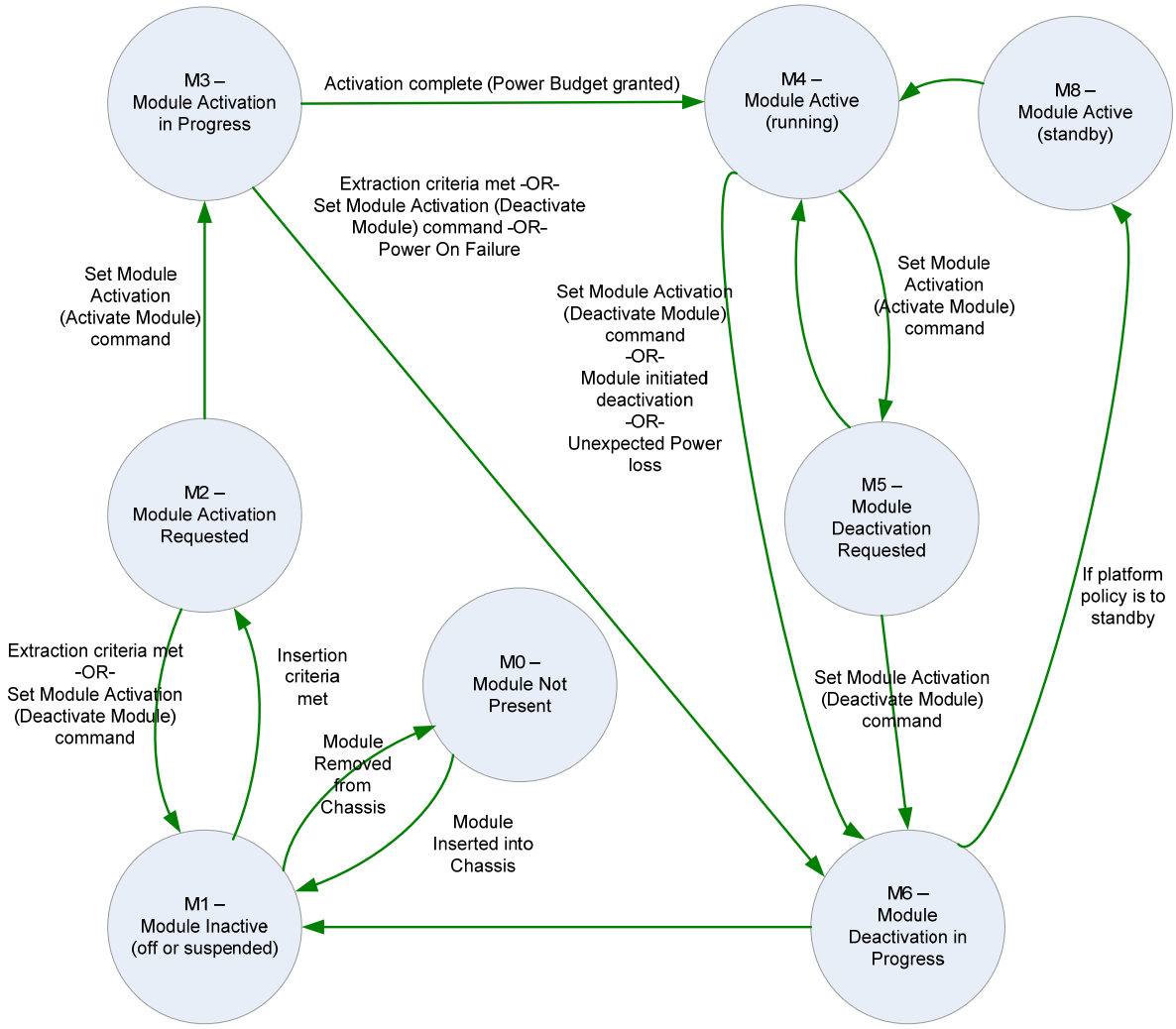
In addition to the Operational State Machine, there is a state variable associated with the Operational State Machine: *Activation Policy*. This variable is used to control transitions out of various operational states. The variable is affected by some SSI-defined IPMI commands (e.g., *Set Module Activation Policy*) and by front panel activation request button presses. The *Activation Policy* variable has two states: 0 = *Deactivate* and 1 = *Activate.*

- *Insertion Criteria Met*
  This condition is defined as the *Activation Policy* variable being equal to *Activation* (*Activation Policy == Activate*), plus any other implementation dependent activation requirements being met. *Insertion Criteria* gate the transition from the *Inactive* to the *Activation Requested* states.

• *Extraction Criteria Met*
This condition is defined as the *Activation Policy* variable being equal to *Deactivation* (*Activation Policy == Deactivate*), plus any implementation

dependent deactivation requirements being met. *Extraction Criteria* gate the transition from the *Activation in Progress* to the *Deactivation in Progress* states and *Active* to the *Deactivation Requested* states.

Figure 6-2: Module Operational State Machine with Valid Transitions



6.5.1.2 State Definitions

M0 – Not Present State

The M0 state represents the *Not Present* state of a blade, which is when it is physically not present in the chassis. From the Chassis Manager perspective, this state can be used to indicate that a particular slot is empty. Administrators can then determine if a blade can be inserted into the slot.

A blade cannot indicate that it is in this state (for obvious reasons); it is the Chassis Manager's responsibility to represent the state of an empty slot as the M0 state.

M1 – Inactive State

The M1 state represents the *Inactive* state of the blade. In this state, the blade is fully seated into its slot. The blade management controller has initialized itself, performed its self checks, determined its Slot ID, enabled its BMI interfaces, and can communicate with the Chassis Manager. The blade management controller can also detect the state of its hardware to determine if the criteria are met for transition towards activation. When a blade is deactivated it will remain in this state. In this state, the blade consumes standby power only and payload power is OFF. This is the only state in which a blade can be properly extracted from the chassis.

A blade in the M1 state will only transition to M2 when its *Insertion Criteria* have been met. If there are no implementation dependent *Insertion Criteria*, this means that on insertion, a blade will automatically transition to M2.

M2 – Activation Requested State

The M2 state represents the *Activation Requested* state of the blade. Transitioning to this state is an implicit request to the Chassis Manager to activate the blade and start normal operation. In this state, the Chassis Manager will read the blade's inventory information and SDRs to determine its identity and capabilities.

In this state, the blade consumes standby power only and payload power is OFF.

A return from this state to M1 will happen if the Activation Request is denied by the Chassis Manager (as communicated by the *Set Activation State (Deactivate Module)* command) or the *Extraction Criteria* are met, for example if a button press causes *Activation Policy == Deactivate*.

M3 – Activation in Progress State

The M3 state represents the *Activation in Progress* state. A blade enters the M3 state on receipt of the *Set ModuleActivation (Activate Module)* command from the Chassis Manager while in the M2 state.

In this state, the Chassis Manager performs a set of specification-defined actions on the blade to determine if the blade can be activated. Such actions include power negotiation, budgeting, and interconnect compatibility verification. Some implementations may also perform implementation-specific operations, such as configuration download, BIOS/EFI download, FPGA code update, VLAN or virtual link configuration of its network interfaces, etc.

After successful completion of power negotiation and a power budget having been assigned to the blade by the Chassis Manager, the blade may apply payload power and verify that power has successfully ramped up on the

different power rails of the payload hardware circuitry that the blade may implement. The specification does not define the exact steps the blade management controller takes to verify that payload power is applied successfully; this description is an example only.

Anytime after the blade has successfully applied payload power, it can autonomously transition to the M4 state. Surprise or ungraceful extraction of a blade in the M3 state could cause undesired results, such as corruption of data on a blade.

A transition to the M6 state **shall** occur under the following cases:

- If a blade determines that its payload power cannot be applied successfully.
- If the Chassis Manager determines that a blade should not be activated, due to either power budget or other issues, it will explicitly deactivate the blade by sending it a *Set Activation (Deactivate Module)* command.
- *Extraction Criteria* are met.

M4 – Active State (running)

The M4 state is a blade's normal operational state. Payload power is ON in this state and the blade performs its mission function.

In this state, a blade management controller monitors its hardware for warning and error conditions, such as thermal and voltage errors and reports them to the Chassis Manager.

A blade will stay in this state as long as *Activation Policy == Activate* and it is not forcibly or explicitly deactivated.

If the *Activation Policy* variable transitions to *Deactivate*, a blade will transition to the M5 state and wait for the Chassis Manager to either deny or approve the deactivation request.

A blade can be explicitly deactivated by the Chassis Manager via a *Set Module Activation (Deactivate Module)* command, resulting in a transition from the M4 state into the M6 state.

A forcible deactivation results from either a non-recoverable power or other failure on the blade or a payload initiated power off. Forcible deactivation causes a transition to the M6 state.

Surprise or ungraceful extraction of a blade in the M4 state could cause undesired results, such as corruption or loss of the operating system and data on the blade.

M5 – Deactivation Requested State

The M5 state represents the *Deactivation Requested* state of a blade. Payload power is ON in this state and the blade continues to perform the function it is typically intended to do. Transition into this state is an implicit request to the Chassis Manager to deactivate the blade and the blade remains in this state while the Chassis Manager evaluates whether the deactivation request of the blade can be granted.

In this state, the Chassis Manager may, based on implementation, communicate with remote management consoles, management applications, or administrators to determine whether it is ok for the blade to deactivate.

If the Chassis Manager determines that it is ok for the blade to deactivate, it sends a *Set Module Activation (Deactivate Module)* command to the blade. The blade then transitions to the M6 state.

Alternatively, if the Chassis Manager decides that it is not ok for the blade to deactivate, it sends a *Set ModuleActivation (Activate Module)* command to the blade. The blade then transitions back to the M4 state.

As in the M4 state, forcible deactivations may occur and cause a transition to the M6 state.

In the M5 state, any power budget granted to the blade is still valid and surprise or ungraceful extraction of a blade in this state could cause undesired results, such as corruption of the operating system and data on blade.

M6 – Deactivation in Progress State

The M6 state represents the *Deactivation in Progress* state of a blade. In this state, a blade begins and completes its process of deactivation. It is important to note that once a blade enters this state, the deactivation process cannot be rolled back or cancelled. Any such roll back or cancellation of deactivation is done in the M5 state only.

A blade should attempt a graceful power-down of the payload. However, if the payload is not responsive or takes an excessively long time to shut down, the blade should force payload power off. The deactivation process may also include other implementation dependent deactivation steps.

The graceful power-down mechanism is not specified, but for example, a compute blade in the M6 state may initiate deactivation of its OS through ACPI mechanisms. Depending on the applications executing on the blade, it may take time for the graceful unloading of applications and the operating system.

After deactivation, the blade power budget is considered revoked, the *Activation Policy* variable is set to *Deactivate*, and the blade **shall** transition from the M6 state to the M1 state. This transition is an indication to the Chassis Manager that deactivation has completed.

Alternatively, it is possible that the OS policy will cause the system to go to a "standby" (M8) state. In this case, the power budget is not revoked.

M7 – Communication Lost State

The M7 state is failure state that is not shown in Figure 6-2. It represents the *Communication Lost* state of a blade. The Chassis Manager's operational state for a blade enters this state when the Chassis Manager cannot communicate with the blade's management controller over any available management interface. Blades do not implement this state themselves.

In this state, all resources that were granted to the blade prior to its entering the Communication Lost state are retained, as it is possible that the blade is completely functional and the management controller on the blade may be going through a reset or a firmware update and may not be able to communicate with the Chassis Manager. The Chassis Manager will resynchronize with blade operational state when communication resumes.

M8 – Active State (standby)

The M8 state represents what is commonly known as "suspend to ram" or "standby" in a system that has an OS button push policy of suspending to RAM. In systems that support ACPI, this is also known as S3 "Standby". This state does not give up the power budget allocated to the running system. Therefore, it transitions directly back to M4 (running) from this state.

6.5.2 Managed Element Insertion Sequence

This section describes the typical insertion sequence for a blade in an SSI modular system.

M0->M1

A blade in the M0 state (Not Present) is inserted into a slot of the chassis. The blade's management controller executes on standby power and initializes itself. After reading its Slot Identifier from the midplane, it computes its BMI IPMB address, initializes its BMI interfaces, and automatically transitions from the M0 to M1 state. This causes an M0 to M1 transition event message to be sent to the Chassis Manager (this is actually optional). In the M1 state, the management controller determines whether the *Insertion Criteria* of the blade have been met such that it can transition out of the M1 state. Other implementation-specific *Insertion Criteria* may exist. The blade waits in the M1 state until all of these have been met.

M1->M2

When the *Insertion Criteria* have been met, the blade will transition to the M2 state, generating an M1 to M2 state transition event message. To the Chassis Manager, the M1 to M2 state transition event message from the blade is equivalent to an activation request. The blade will remain here until notified to activate.

M2->M3

If the administrative policy allows the blade to be activated, the Chassis Manager sends a *Set Module Activation (Activate Module)* IPMI command to the blade, causing the blade to transition from the M2 state to the M3 state. This, in turn, generates an M2 to M3 state transition event message.

If policy does not allow blade activation, the Chassis Manager will send a *Set Module Activation (Deactivate Module)* IPMI message to the blade, setting the *Activation Policy* variable to *Deactivate* and causing it to transition back to M1. *Activation Policy == Deactivate* ensures the blade will stay in the M1 state.

M3->M4

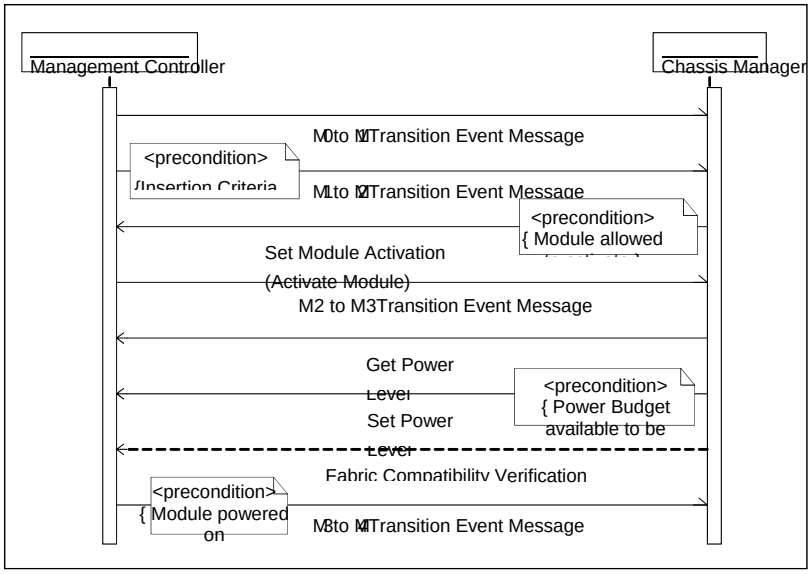
In the M3 state, the Chassis Manager performs a variety of operations to determine if it's truly appropriate to activate the blade. These include power budgeting, and interconnect compatibility verifications. The Chassis Manager queries the blade for its power budget requirements and verifies if there is enough power available in the chassis to be able to grant the highest level of power requested by the management controller. If the highest level of power is unavailable, the Chassis Manager will assign the best power level that can be assigned to the blade.

If there is not enough power budget available in the chassis to power up the blade in a power level that it can execute with, the blade may be left in the M3 state, so that power budget can be granted to it when some other blade in the chassis is deactivated such that the power previously allotted to the blade that was just deactivated can be applied to the blade in M3 state. Or, the Chassis Manager may transition the blade to M1 by deactivating it.

Once the Chassis Manager has granted the blade a power budget, the blade may apply payload power and then transition to M4, at its discretion. Since the *Activation Policy* variable was previously set to *Activate* in the M1 state, the blade will remain in M4 until the *Activation Policy* variable transitions to *Deactivate* or some other condition forces a deactivation.

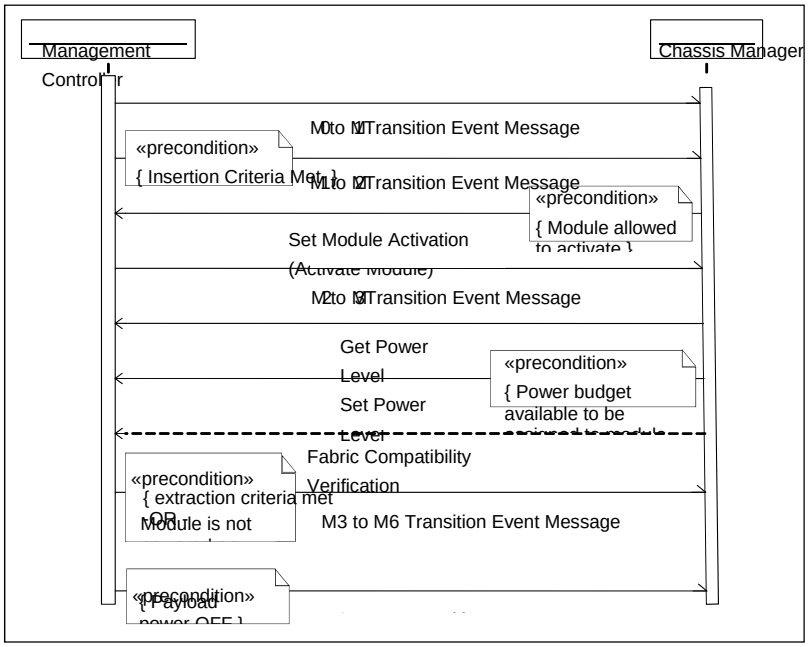
The M4 state is the normal operational state of the blade. In this state, the payload power is ON and the blade performs the functions that it is typically supposed to do.

Figure 6-3: Managed Module Transitions into the Active State



If the blade's management controller detects that the payload power could not be applied successfully due to some error, the management controller initiates deactivation process for the blade by transitioning it to the M6 state by sending the M3 to M6 state transition event message to the Chassis Manager, as shown in Figure 6-4.

Figure 6-4: *Module* Transition from "Activation in Progress" to "Inactive" State



6.5.3 Managed Element Extraction Sequence

This section describes the process of deactivating a currently operational (M4 state) Compute Blade.

Deactivation can be either requested or forced, depending on source and conditions.

- A deactivation request can be initiated through various sources, such as front panel activation request (power) button, or *Set Module Activation Policy (Activation Policy=Deactivate)* command from the Chassis Manager.
- Forced deactivation can be a result of a payload initiated power down, spontaneous loss of power (e.g., HW failure), thermal shutdown, or administrative decision by the Chassis Manager or remote management software.

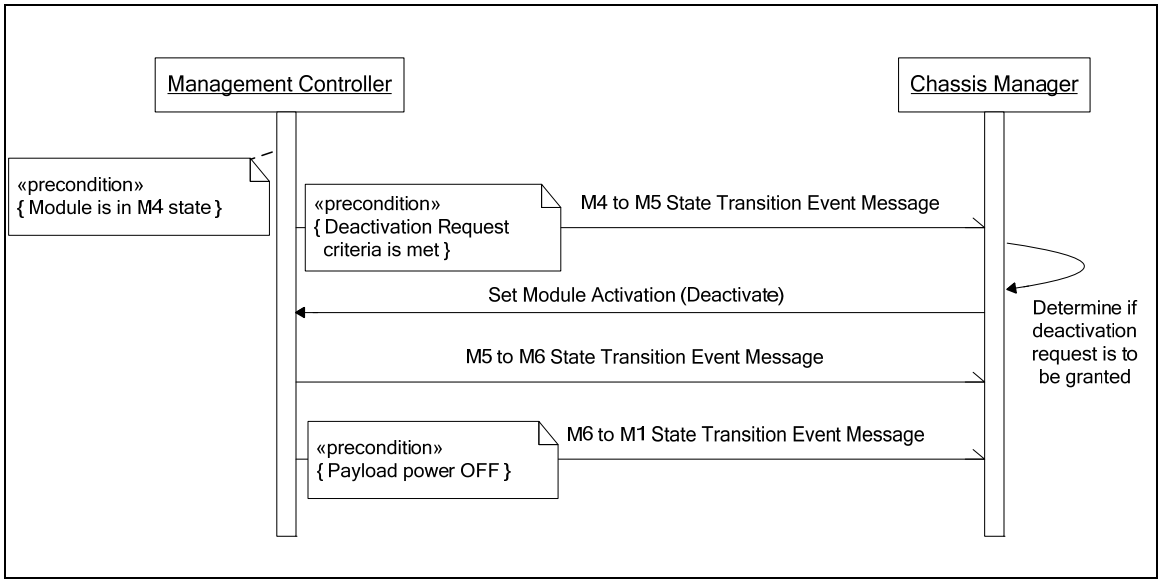
6.5.3.1 Requested Deactivation

During a requested deactivation, the blade transitions from the M4 (*Active*) state to the M5 (*Deactivation Requested*) state and sends an M4 to M5 state transition event message to the Chassis Manager. In the M5 state, the blade continues to perform its normal operations; power budget and other resources that are granted to it are still valid. From the payload's perspective, there is no change to its state. On receipt of the M4 to M5 state transition event message,

the Chassis Manager evaluates whether the blade can be deactivated. The Chassis Manager may communicate with external remote management consoles or system management application software in an implementation specific fashion to determine if the blade can be deactivated.

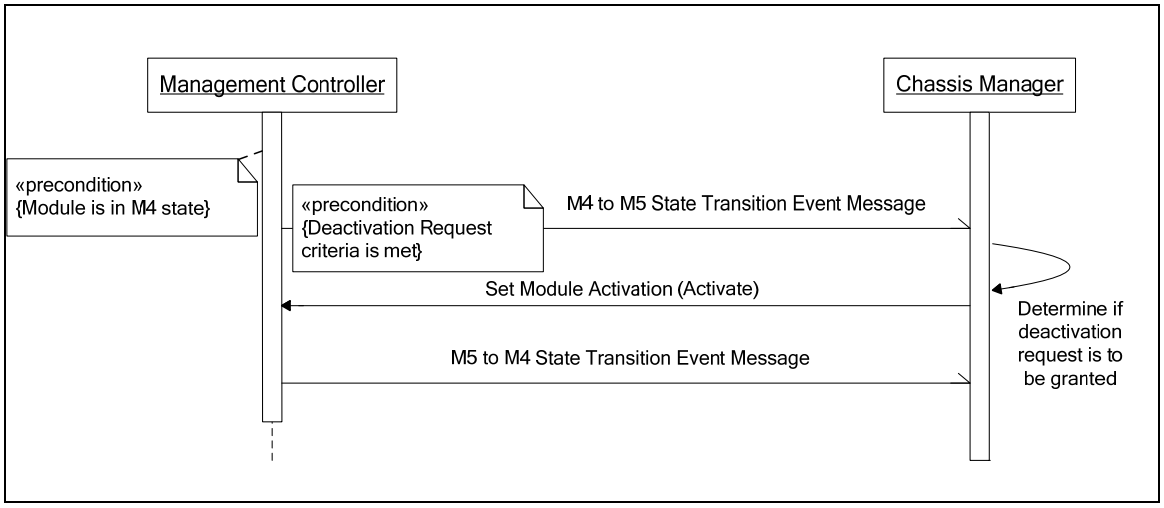
In the M5 state, the Chassis Manager can either allow or reject deactivation of the blade. If the deactivation request is allowed, the Chassis Manager sends a "*Set Module Activation (Deactivate Module)*" command to the blade. This causes the blade to transition from the M5 state to the M6 (*Deactivation in Progress*) state and to send an M5 to M6 state transition event message to the Chassis Manager, as shown in Figure 6-5. Since transitioning from the M6 state to the M1 state causes the *Activation Policy* variable to be set to *Deactivate*, the blade will remain in M1 until it is removed or reactivated.

Figure 6-5: Chassis Manager Granting Deactivation in "Deactivation in Progress" State



Alternatively, the Chassis Manager can reject the deactivation by sending a "*Set Module Activation (Activate Module)*" command to the blade. This causes the management controller on the blade to transition from the M5 state back to the M4 (*Active*) state and then to send an M5 to M4 state transition event message to the Chassis Manager. Since the above command also has the effect of setting the *Activation Policy* state variable to *Activate*, the blade will stay in the M4 state.

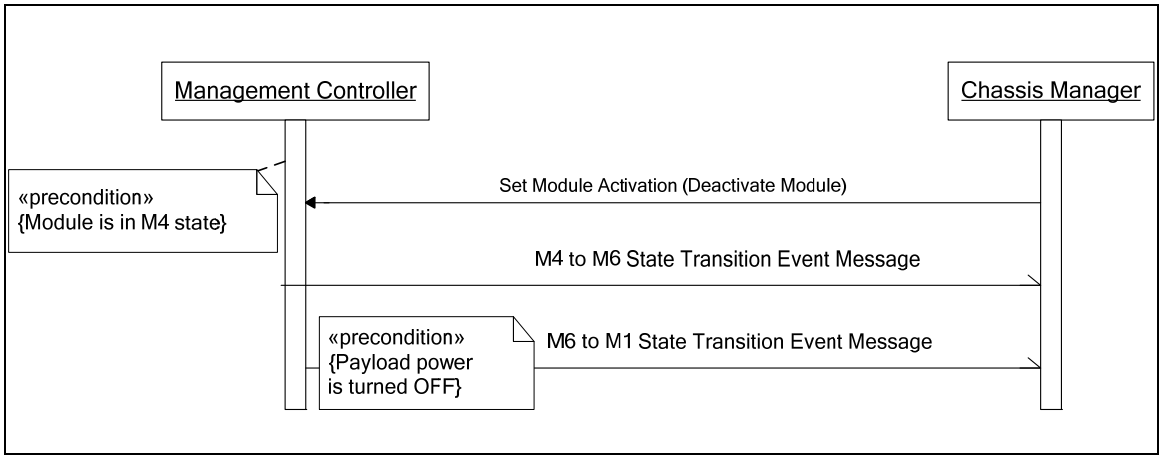
Figure 6-6: Chassis Manager Rejecting Deactivation Request



6.5.3.2 Forced Deactivation

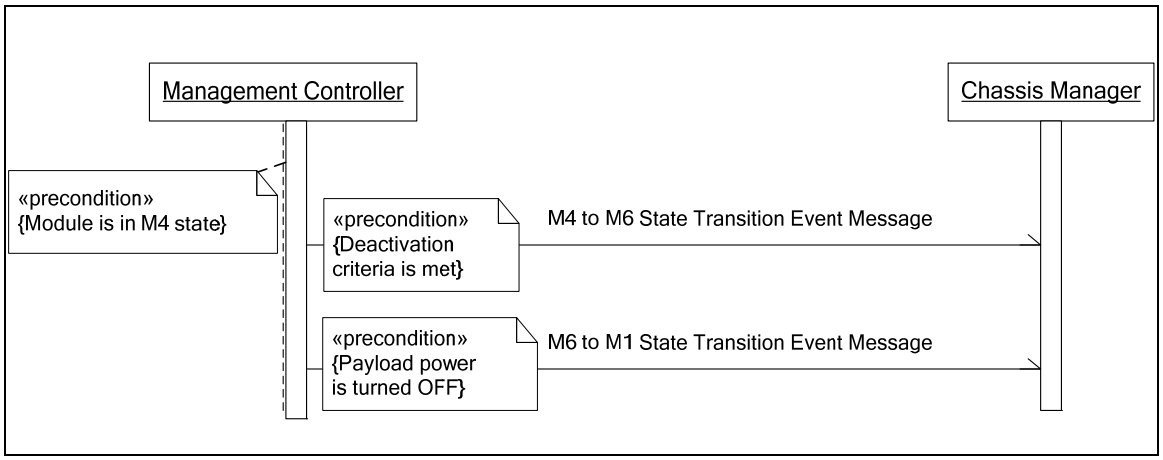
If the Chassis Manager determines (through some OEM mechanism or due to administrator action) that the blade needs to be deactivated. The Chassis Manager sends a "*Set ModuleActivation (Deactivate Module)*" command to the blade with the appropriate blade Device ID, causing the management controller on the blade to transition from the M4 state to the M6 (*Deactivation in Progress*) state and to send an M4 to M6 state transition event message to the Chassis Manager. The sequence diagram for such deactivation is shown in Figure 6-7. The blade will then transition to the M1 state and stay there until it is either removed or reactivated.

Figure 6-7: Chassis Manager-Initiated Deactivation



Alternatively, when the blade is in the M4 state, it may initiate its deactivation process without requiring permission from the Chassis Manager. In this case, the *module* may have determined through some OEM mechanism that it needs to be deactivated (e.g., emergency shutdown) or the payload forces payload power off. Is this case, the blade transitions to the M6 state and sends an M4 to M6 state transition event message to the Chassis Manager. After entering the M6 state, if payload power is not already off, a graceful shutdown will be attempted. A sequence diagram for such deactivation is shown in Figure 6-8

Figure 6-8: Managed Element-Initiated Deactivation



This specification does not define the exact mechanism of how the management controller on the blade communicates with the payload software or hardware to perform a shutdown; such mechanisms are implementation specific. However, this specification recommends that the deactivation should be done in a graceful fashion to prevent data corruption or loss. After the payload software is shut down, the management controller on the blade turns payload power OFF and then transitions from the M6 state to the M1 state and sends an M6 to M1 state transition event message to the Chassis Manager.

6.5.3.3 Reinsertion and Reactivation Sequence

A blade in the *Inactive* state may be extracted out of the chassis and reinserted into the chassis or may be reactivated, depending on usage models in the deployed environment.

The case of reinsertion of a blade into the chassis is identical to a fresh insertion of a blade in the chassis, and its handling is as described in the insertion sequence in section 6.5.1.1.

As for reactivation, when a blade enters the M1 state from the M2 or M6 state, its *Activation Policy* variable is set to *Deactivate*, preventing its transition out of the M1 state to the M2 state.

The *Activation Policy* variable can be set to *Activate*, enabling transition to M2, by either front panel power button presses or by the Chassis Manager's sending

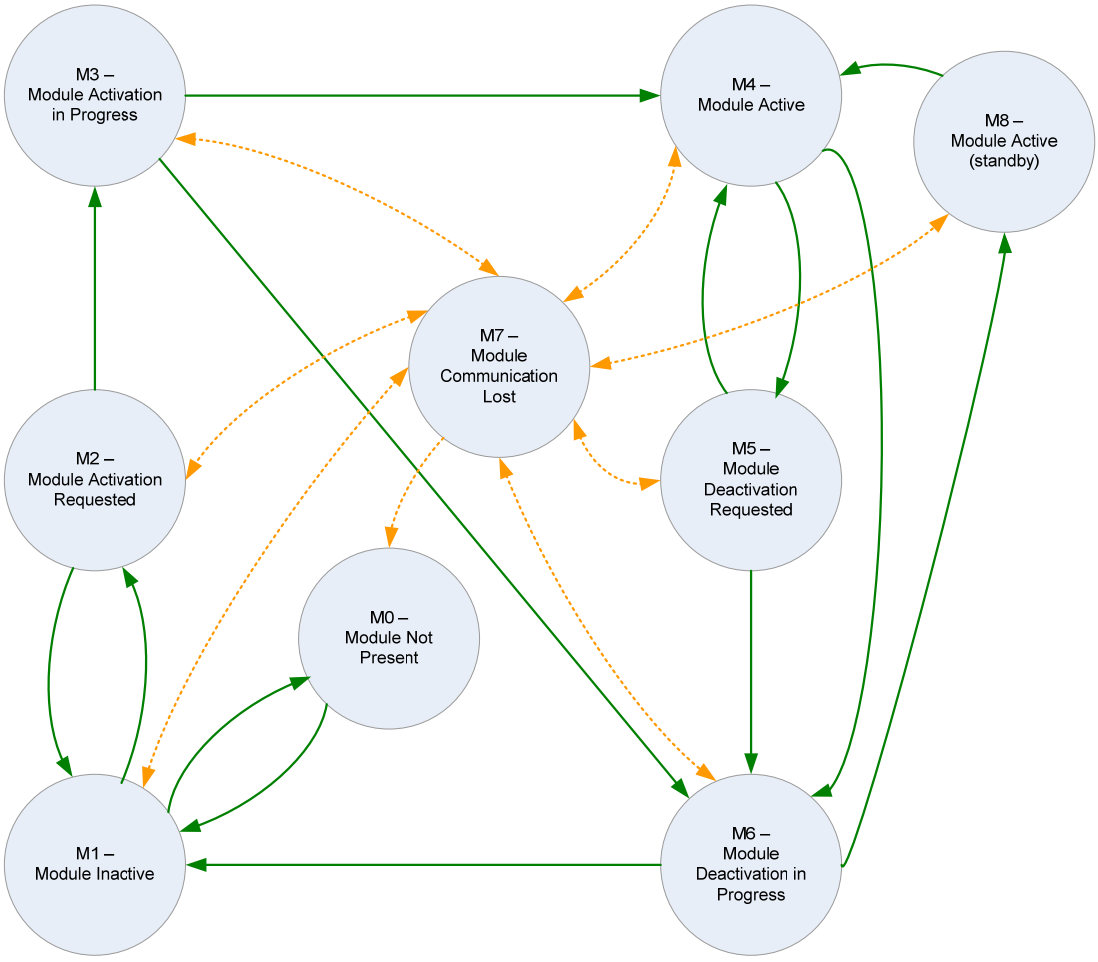
a *Set Module Activation Policy (Activation Policy = Activate)* command to the blade's management controller.

When the *Activation Policy* variable is set to *Activate*, the blade can transition out of the M1 state to the M2 state, assuming all other *Insertion Criteria* the blade may have are also satisfied.

6.5.4 Communication Lost

Apart from the valid state transitions shown in Figure 6-2, there exists another state representing a communication failure between the Chassis Manager and the blade. This is the M7 (*Communication Lost*) state, and is only implemented by the Chassis Manager as shown in Figure 6-9.

Figure 6-9: Module Transitions for "Communication Lost" State



After the blade enters the M1 state, the Chassis Manager periodically pings the blade to ensure that the Chassis Manager can communicate with the blade's management controller. It is possible that during normal operations, the

management controller may stop responding to the Chassis Manager over all available interfaces. This could happen for various reasons, such as management controller firmware hang, an management controller firmware update in progress, pins have shorted on the midplane connector, or a catastrophic power supply failure on the blade. It is critical that loss of communication between the Chassis Manager and the management controller(s) is detected and reported, so that administrators can determine whether any action needs to be taken to correct the failure.

When the Chassis Manager determines that it cannot communicate with the blade at all, it transitions its view of the blade operational state from the blade's last known state to the M7 state. In the M7, any resources assigned to the blade (power budget, chassis interconnect) are retained without change. The M7 state is only implemented by the Chassis Manager.

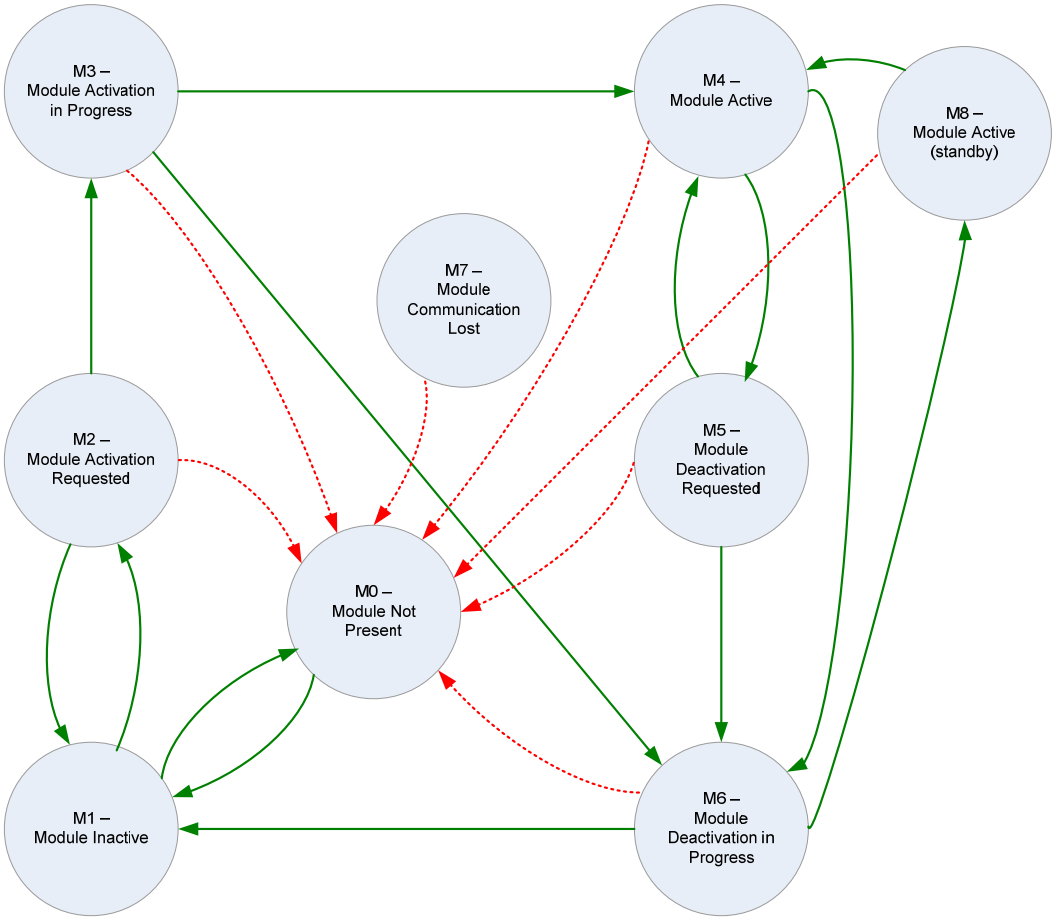
In the M7 state, the Chassis Manager will continue to periodically attempt to communicate with the blade. If and when communication is restored, the Chassis Manager queries the blade for current operational state to determine if it transitioned to some other state while communication was blocked. If extraction of the blade is detected via the slot presence signal change, the Chassis Manager will transition the blade to the M0 state.

6.5.5 Surprise Extractions

The remaining state transitions occur when the blade is extracted from the chassis ungracefully. This type of extraction can cause loss of data, corruption or other undesired consequences. While not expected as part of normal behavior, this specification expects surprise or ungraceful extractions of blades from the chassis. In Table 6-10 these transitions are shown in dashed red.

The Chassis Manager detects extraction of the blade from the chassis via the slot's Blade Present signal. When the Chassis Manager detects extraction of the blade from the chassis, it generates an Mx (current state) to M0 transition event on behalf of the blade that was extracted from the chassis and recovers any chassis resources allocated to the slot (e.g., power budget).

Figure 6-10: Transition for Surprise Extractions



6.5.6 Managed Module Operational State Sensor

In the SSI Compute Blade architecture, the IPMI defined *FRU State* sensor (sensor type 2Ch) is used to represent the operational state of a blade. The IPMI *FRU State* sensor is renamed as the *Managed Element Operational State* sensor in this specification.

The *Managed Module Operational State* sensor provides the blade with the ability to generate events for state transitions that the Chassis Manager can receive, using them to synchronize its internal state about the blade with the state that the blade is in. Secondly, the implementation of the *Managed Module Operational State* sensor also allows for a standardized mechanism to query the operational state of the blade at any point in time.

Every blade **shall** implement one *Managed Module Operational State* sensor representing the blade's operational state. This sensor **shall** provide readings.

In order to allow the Chassis Manager to receive state transition events, the sensor **shall** be initialized and capable of sending events before the blade's operational state machine transitions out of the M1 state.

A corresponding SDR for this sensor **shall** be present in the blade's SDR repository and **shall** be configured to support assertion event generation for all state transitions.

Table 6-13 shows the Event data generated for a *Managed Module Operational State* sensor.

A blade **shall** support the sensor specific event data 1 and 2 fields for this sensor type.

Table 6-13: Managed Module Operational State Sensor Event Data

Field	Content / Purpose
Event Type /Direction	[7] - Event Direction = *0b (Assertion)* [6:0] - Event Type = 6Fh (sensor specific)
Event Data 1	[7:4] - sensor specific event-extension code in Event Data 2, OEM code in Event Data 3 = Eh. [3:0] - Current State. See the IPMI specification for the FRU State sensor offsets.
Event Data 2	[7:4] - Cause of state change. See IPMI specification for values. [3:0] - Previous State. See the IPMI specification for the FRU State sensor offsets.
Event Data 3	[7:0] - FRU Device ID associated with the sensor = 0

6.5.7 Managed Module Operational State Management Commands

This section describes the SSI-defined IPMI commands used in the process of activating and deactivating blades. They are usually sent by the Chassis Manager to blades during the insertion and extraction processes.

6.5.7.1 Set Module Activation Command

The *Set Module Activation* command is used by the Chassis Manager to either activate or deactivate a blade.

Table 6-14: Set Module Activation Command Structure

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	2	FRU Device ID = 0
	3	Module Activation/Deactivation 00h = Deactivate Module 01h = Activate Module 02h – FFh = Reserved
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.

This command is only valid in certain blade operational states: M2, M3, M4, and M5. Some states add additional constraints. When this command is received in an invalid state, the blade **shall** return the IPMI defined completion code D5h (*Cannot execute command. Command, or request parameter(s), not supported in present state)*

Table 6-15 shows the effects of receiving this command when the blade is in each of the operational states.

Table 6-15: Example Effects of Set Module Activation Command

Managed Module Operational State	Module Activation / Deactivation in the request byte	Result
M0	00h = Deactivate Module	Not applicable in this state
	01h = Activate Module	
M1	00h = Deactivate Module	Not applicable in this state
	01h = Activate Module	
M2	00h = Deactivate Module	Module transitions from the M2 state to the M1 state. *Activation Policy* set to *Deactivate*
	01h = Activate Module	Module transitions from M2 state to the M3 state.
M3	00h = Deactivate Module	Module transitions from M3 state to the M6 state. *Activation Policy* set to *Deactivate*
	01h = Activate Module	Not applicable in this state
M4	00h = Deactivate Module	Module transitions from M4 state to the M6 state. *Activation Policy* set to *Deactivate*
	01h = Activate Module	Not applicable in this state
M5	00h = Deactivate Module	Module transitions from M5 state to the M6 state. *Activation Policy* set to *Deactivate*
	01h = Activate Module	Module transitions from M5 state to the M4 state. *Activation Policy* set to *Activate*
M6	00h = Deactivate Module	Not applicable in this state
	01h = Activate Module	
M7	00h = Deactivate Module	Not applicable in this state
	01h = Activate Module	
M8	00h = Deactivate Module	Not applicable in this state
	01h = Activate Module	Module Transistions from M8 Standby to M4 Active

6.5.7.2 Set Module Activation Policy Command

The *Set Module Activation Policy* command is used by the Chassis Manager to change the state of the *Activation Policy* state variable that is used to prevent a blade from proceeding through deactivation or activation respectively.

Table 6-16: Set Module Activation Policy Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	2	FRU Device ID (**shall** be 0)
	3	Module Activation Policy Value [7:1] - Reserved [0] - 0 = Set *Activation Policy* state variable to *Deactivate* 1 = Set *Activation Policy* state variable to *Activate*
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.

This command will normally be used by the Chassis Manager when re-activating a blade after it has been deactivated for some reason (and now stopped in M1), but it can be used to trigger state changes from other states as well.

Other commands (*Set Module Activation*), state transitions (e.g., M6->M1), and physical actions (e.g., Face Plate button press) can cause the *Activation Policy* state variable to change its value.

Table 6-17 shows the effects of receiving this command by the blade when it is in each of the operational states. When this command is received in an inappropriate state, the blade **shall** return the IPMI defined completion code D5h (*Cannot execute command. Command, or request parameter(s), not supported in present state)*

Table 6-17: Effects of Receiving the Set Module Activation Policy Command

Managed Module Operational State	Activation Policy Value	Result
M0	Not applicable	Not applicable in this state
M1	*Deactivate*	No change, blade stays in M1 state.
	Activate	Blade can transition to the M2 state, subject to other insertion criteria.
M2	*Deactivate*	Blade can transition to the M1 state, subject to other extraction criteria.

Managed Module Operational State	Activation Policy Value	Result
	Activate	No change, blade stays in M2 state.
M3	*Deactivate*	Blade can transition to the M6 state, subject to other extraction criteria.
	Activate	No change, blade stays in M3 state.
M4	*Deactivate*	Blade transitions to the M5 state.
	Activate	No change, blade stays in M4 state.
M5	Not applicable	Not applicable in this state
M6	Not applicable	Not applicable in this state
M7	Not applicable	Not applicable in this state
M8	Deactivate	Not applicable in this state
	Activate	Blade Transitions to M4

6.5.7.3 Get Module Activation Policy Command

The *Get Module Activation Policy* command is used to query the current value of the blade's *Activation Policy* state variable.

Table 6-18: Get Module Activation Policy Command

Field	Byte	Content/Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	2	FRU Device ID (**shall** be 0)
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	3	Module Activation Policy value [7:3] - Reserved [0] - 0 = *Deactivate* 1 = *Activate*

6.6 Power Management

One of the most important functionalities provided by system management architecture in an SSI environment is the power management of blades in the chassis. Power is a chassis shared resource and is managed by the Chassis Manager, which tracks the power budgets allocated to the various blades in the chassis and negotiates with the blades to allocate or recover their power budgets. The Chassis Manager guarantees that the sum of the power budgets allocated to the blades in the chassis is equal to or below the available chassis power capacity.

Power negotiation occurs when a blade is being activated (M3 state). Power renegotiation occurs during a blade's normal operating (M4) state and can be instigated by either the Chassis Manager or the blade. A blade's power budget is automatically reclaimed by the Chassis Manager when the blade transitions from the *Deactivation in Progress* (M6) state to the *Inactive* (M1) state.

A blade is not allowed to power on its payload if there is not enough chassis power available.

For each slot that it occupies, a managed module advertises the power levels that it implements and the power level it desires to operate at. These may be quantized into a small number of discrete power levels or expressed as a range of acceptable values.

If discrete, the blade advertises a minimum of 1 and a maximum of 21 power levels per slot. If range-based, the blade advertises 3 power levels – the minimum that it can accept, its desired power level, and the maximum level it can consume. The desired level may be equal to either the minimum or maximum values.

Each slot also has a current power level and a module desired power level. The Chassis Manager queries a module's supported and desired power levels, determines the available power based on chassis capacity and possible policy rules, and notifies the blade of the power it is allowed to use.

Different types of Compute Blades may have different power budget requirements, which are independent of the chassis. A blade may be able to execute with different power levels, depending on the time of the day, workload characteristics, or other externally determined policies. Blades capable of dynamically raising or lowering performance and/or functionality without going through a re-initialization process can consume different amounts of power, depending on the performance/functionality it provides.

Such a blade supports dynamic renegotiation of power to request power budget changes from the Chassis Manager, increasing or decreasing its current power level depending on performance/functionality requirements.

6.6.1 Multi-slot Compute Blades

A blade that spans more than one slot can draw power from each slot that it occupies. The power negotiation commands require a blade to negotiate for power for each slot that it occupies.

6.6.2 Power Negotiation

Power negotiation is the process of the Chassis Manager providing a power budget to a blade. It consists of

- Determining the power requirements for the blade (performed by the blade, but instigated by the Chassis Manager)
- Determining if the chassis has sufficient power available to allocate to the blade (performed by the Chassis Manager)
- Granting (or not) that power to the blade (also by the Chassis Manager).

The power negotiation process is performed after a blade enters the M3 (*Activation in Progress*) state. Power negotiation can also be performed for a blade already in the M4 (*Active*) state as part of the power renegotiation process, as described in section 6.6.3.

A blade **shall** be ready to negotiate for a power budget anytime after reaching the M3 state. This means that the blade **shall** have determined its supportable power levels by that time.

A blade **shall** aggregate the power requirements for its base hardware and any attached options, mezzanines, or associated hardware that derives power through the blade's slot connector(s).

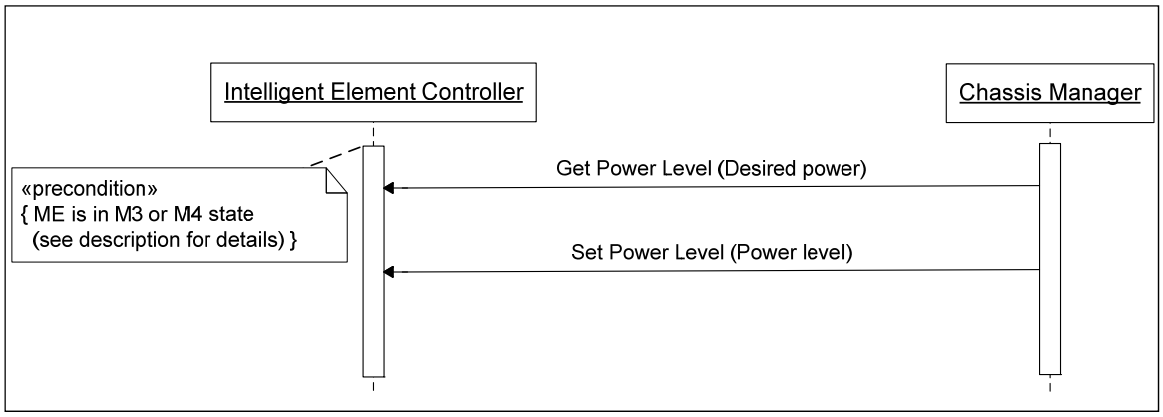
A blade **shall** provide separate supported power levels and desired power level for each slot that it occupies.

At the beginning part of the power negotiation process, the Chassis Manager sends the blade *Get Power Level* commands (one per occupied slot) that query for blade supported and desired power levels.

Once the Chassis Manager has retrieved the blade's desired power levels, it determines what power it can allocate, and sends the blade a *Set Power Level* command either granting a power budget or explicitly setting the power level to 0, indicating no power for the blade. Once the power levels have been set for all the slots of a blade, the negotiation process is considered complete.

A sequence diagram for power negotiation between the Chassis Manager and the management controller on blade is shown in Figure 6-11.

Figure 6-11: Sequence Diagram for Power Negotiation with Chassis Manager



6.6.2.1 Get Power Level Command

The Chassis Manager queries the blade for its power requirements by sending the *Get Power Level (desired power, slot offset)* command, as described in Table 6-19. The blade responds with the following information:

- Whether the blade supports dynamic reconfiguration of power consumption. This indicates that the blade can dynamically reconfigure power use without interrupting payload service.
- Whether the power levels are defined in a discrete or range based form.
- The blade's current or desired power level.
- The specific power at each implementation dependent supported power level.

A blade that does not support dynamic reconfiguration of power **shall** implement at least one power draw level.

After the blade enters the M3 state (or during power renegotiation) the Chassis Manager sends one instance of *Get Power Level (Desired power draw levels, slot offset)* command for each slot occupied by the blade.

Table 6-19: Get Power Level Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	2	FRU Device ID (**shall** be 0)
	3	Power Type and Slot [7:5] – Reserved

Field	Byte	Content / Purpose
		[4:3] – Power Type 00h = Current power draw level 01h = Desired power draw level All other values are reserved. [2:0] – Slot offset (0 based) Only valid for FRU Device ID of 0
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	3	Power properties of the managed module [7] - Dynamic power configuration. 1b = the module supports dynamic power reconfiguration. 0b = no dynamic power reconfiguration supported [6] - Power Draw format 1b = Range based – any power level value between Power Level[1] and Power Level[Max] may be used. 0b = Discrete power levels – only advertised power levels may be chosen. [5] - Reserved [4:0] - Requested Power Level. Represents the current power level or desired power level for the module is requesting for this slot, depending on the Power Type in the request data byte 3. This value is an index into the Power Level [] table starting from byte 5 up to byte 26. There is a maximum of 21 power level entries, and this index (1 based) represents which power level is desired.
	4	Power Multiplier. This byte defines the number of tenths of watt by which to multiply values in byte 5 and beyond.
	5	Power Level [1]. This entry represents the lowest level of power that the module can execute with. 00h is returned in case module does not consume power from the Slot offset.
	(…)	
	(N)	Power Level [Max]. The last entry represents the highest level of power that the module can execute with. The maximum value of N is 26 (corresponding to a maximum of 21 power levels) due to 32 byte IPMI message size limitations.

After the requested power levels are retrieved via this command, the Chassis Manager computes whether there is enough power budget available to satisfy the blade's desired levels. The Chassis Manager **shall** grant the highest power level, less than or equal to the blade's desired power level, consistent with available power.

After determining the power levels that the blade can execute with, the Chassis Manager allocates the power to the blade and deducts it from the total

assignable power budget. Lastly, the Chassis Manager notifies the blade of its power budget via the *Set Power Level (Power levels)* command to the blade, as described in section 6.6.2.2.

6.6.2.2 Set Power Level Command

This command is used by the Chassis Manager to notify a blade of the available power budget for each of its occupied slots. It can be used to both allocate power as well as to reclaim power (by setting all power levels to 0).

During blade activation power negotiation (M3 state), the Chassis Manager **shall** always send a *Set Power Level* command to the blade, even if there is no power available (power level 0 – off).

During blade activation power negotiation (M3 state), on receipt of the *Set Power Level (power levels)* command with non-zero power level(s), the blade powers up its payload, performs initialization tests, and proceeds to *Active* (M4) state.

Table 6-20: Set Power Level Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	2	FRU Device ID (**shall** be 0)
	3	Power Level – slot offset 0 00h = Power OFF, only standby power is consumed 01h – FFh = Power level value being granted, with same multiplier as advertised for this slot in the Get Power Level command.
	(…)	
	(3+N)	Power level - slot offset N One power level entry is present for each slot occupied by the module. Format of this byte is the same as in response data byte 3. A maximum of 8 slot offsets are possible.
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.

If the power level being granted is higher than the current level, the blade can increase its power consumption to the level granted anytime after the receipt of the command. In all cases, the blade has the responsibility to ensure that it does not consume more power than what it was granted by the Chassis Manager.

6.6.3 Power Renegotiation

Power renegotiation is the process of changing a blade's power budget while the blade is in the *Active* (M4) state. This can be initiated by either the Chassis Manager or the blade itself.

In order to participate in power renegotiation, a blade **shall** be able to dynamically change its power usage without interrupting payload service. This is indicated to the Chassis Manager via the *Dynamic power configuration* bit in the *Power Properties* byte of the *Get Power Level* command response.

If the power renegotiation fails for some reason, such as insufficient power budget in the chassis to allocate to the blade, the blade continues to execute with the power level already assigned to it.

6.6.3.1 Chassis Manager Initiated Power Renegotiation

In normal operational mode (M4), it is possible that the Chassis Manager will determine (perhaps based on external Management console/applications) that a blade supporting dynamic reconfiguration of power needs to execute with either a lower or higher power. In such cases, the Chassis Manager can renegotiate power.

If the Chassis Manager has not cached the blade's advertised power levels, it can send the blade *Get Power Level (Current power levels)* commands to determine the possible power level options.

The Chassis Manager will then send the blade a *Set Power Level* command with the new levels. If the new levels are lower than the previous ones, the blade **shall** reduce its power usage to the new levels.

6.6.3.2 Compute Blade Initiated Power Renegotiation

In some cases it may be necessary for a running (M4) blade to modify its power usage. Such conditions may occur when there is an increase in workload on the blade requesting more power, or when a blade is able to relinquish some power allotted to it back to the Chassis Manager. This could happen if the blade required a higher power level at initial power on due to components (e.g., disk drives) startup power requirements. After the power-on transient is over, the blade can tell the Chassis Manager of its reduced steady state power needs.

A blade initiates power renegotiation by sending a *Renegotiate Power* command (section 6.6.3.3) to the Chassis Manager. The Chassis Manager then proceeds to send the blade *Get Power Level (Desired power level)* commands (one per occupied slot) to determine the newly desired power level(s). The Chassis Manager will then send the blade a *Set Power Level* command either satisfying the blade's desired power level(s) or repeating the current power levels if the blade's request is denied.

6.6.3.3 Renegotiate Power Command

To initiate power renegotiation, the blade sends the *Renegotiate Power* command to the Chassis Manager.

Table 6-21: Renegotiate Power Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	2	FRU Device ID (**shall** be 0)
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.

On receipt of the *Renegotiate Power* command, the Chassis Manager performs power negotiation as described above in section 6.6.3.2.

6.6.4 Power Related Sensors

On a blade, various power related failures can occur that need to be monitored for and reported by the blade management controller. This section details the blade required power related sensors.

6.6.4.1 Input Voltage

A blade management controller **shall** monitor the blade input voltage. The sensor **shall** be a threshold-based sensor (event/reading type 01h) of type *Voltage* (02h). Voltage values below the *under voltage warning threshold*, and an equivalent positive excursion, specified in section 4.3 **shall** trigger a *Non-critical* event, lower or upper as appropriate. Voltages below the *under voltage shutdown* value and above the *voltage surge* value **shall** trigger a *Critical* event, again lower or upper as appropriate.

The voltage regulator providing power to the management sub-system needs to be tolerant to excursions larger than the critical values to allow the blade to log the generated events and forward them to the Chassis Manager.

6.6.4.2 Over-Current / Voltage Regulator Failure

A blade management controller **shall** monitor the various voltage regulators on the blade. This can be done either by monitoring health signals from the regulators or the regulator output voltages. Each voltage regulator **shall** be represented by a sensor, either threshold-based or digital-discrete, as appropriate. The sensors **shall** generate events when the voltage regulators are not operating as required.

Note that although the voltage regulator providing the management sub-system power can be monitored, its failure will not be able to be logged since the management controller will no longer be operating.

6.7 Chassis Interconnect Management

Module Signal Interconnect OEM FRU Records **shall** be used on both the baseboard and Mezzanine card to determine the interconnect capabilities. Auto-negotiation will be handled in-band, and DC blocking caps are used to electrically protect the components in the case of fabric incompatibility. The Hardware Address Table below is used to store any hardware addresses (in the case of Ethernet, MAC addresses) that are associated with the given links below. In the tables below, the term "Link" represents a single logical link that may comprise a single channel (such as KX) or an aggregation of channels (such as KX4).

Table 6-22: Module Signal Interconnect Record

Offset	Length	Definition
0	1	*Record Type ID* – C0h (OEM) **shall** be used
1	1	*End of List/Version* [7] - End of list. Set to 1 for the last record [6:4] - Reserved, write as 0h [3:0] - Record format version = 2h for this definition
2	1	*Record Length* = 1Bh
3	1	*Record Checksum* – holds the zero checksum for the record.
4	1	*Header Checksum* – holds the zero checksum for the header
5	3	*Manufacturer ID* **– LS byte first.** Write as the three byte ID assigned to SSI – 7244h for this specification (44,72,00 in the record in offset 6,7,8)
8	1	*SSI Record ID – Module Signal Interconnect Record* = 27h
9	1	*Record Format Version* – version for this record type = 00h
10	1	*Link 0 (First Primary)Technology* (must be 02h - Ethernet)
11	1	*Link 0 Technology Detail* 00h = Not Implemented (See Table 6-23 for specific values)
12	1	*Link 0 Width* Number of Lanes used for the link 00h = Not Implemented 01h = x1 02h = x2 03h = x3 04h = x4

Offset	Length	Definition
13	1	*Link 0 Reserved*
14	1	*Link 1 (Second Primary) Technology* (must be 02h - Ethernet)
15	1	*Link 1 Technology Detail*
16	1	*Link 1 Width*
17	1	*Link 1 Reserved*
18	1	*Link 2 (Channel 5 or 5 & 6) Technology* 00h = Not Implemented 01h = PCI Express 02h = Ethernet 03h = Fibre Channel 04h = Infiniband® 05h = SAS 06h = SATA 07h - FFh = Reserved
19	1	*Link 2 Technology Detail*
20	1	*Link 2 Width*
21	1	*Link 2 Reserved*
22	1	*Link 3 (Channel 6) Technology*
23	1	*Link 3 Technology Detail*
24	1	*Link 3 Width*
25	1	*Link 3 Reserved*
26	1	RSVD

Table 6-23: Technology-specific Support

Technology	Value	Type
PCI Express (01h)	01h	PCIe Gen1
	02h	PCIe Gen2
		reserved for future use
Ethernet (02h)	01h	1000BASEKX
	02h	10GBASEKR
	03h	10GBASEKX4
		reserved for future use
FibreChannel (03h)	01h	2GFC
	02h	4GFC
	03h	8GFC
		reserved for future use
Infiniband (04h)	01h	InfiniBand* SDR
	02h	InfiniBand DDR
	03h	InfiniBand QDR
		reserved for future use
SAS (05h)	01h	3Gb
	02h	6Gb
		reserved for future use
SATA (06h)	01h	1.5Gb
	02h	3Gb
	03h	6Gb
		reserved for future use

Table 6-24: Hardware Address Table

<table>
<tr><th>Offset</th><th>Length</th><th>Definition</th></tr>
<tr><td>0</td><td>1</td><td>Record Type ID – C0h (OEM) shall be used</td></tr>
<tr><td>1</td><td>1</td><td>End of List/Version
[7] - End of list. Set to 1 for the last record
[6:4] - Reserved, write as 0h
[3:0] - Record format version = 2h for this definition</td></tr>
<tr><td>2</td><td>1</td><td>Record Length = 0Bh + N (length of address table)</td></tr>
<tr><td>3</td><td>1</td><td>Record Checksum – holds the zero checksum for the record.</td></tr>
<tr><td>4</td><td>1</td><td>Header Checksum – holds the zero checksum for the header</td></tr>
<tr><td>5</td><td>3</td><td>Manufacturer ID – LS byte first. Write as the three byte ID assigned to SSI – 7244h for this specification (447200h in the record in offset 6,7,8)</td></tr>
<tr><td>8</td><td>1</td><td>SSI Record ID – Address Table = 28h</td></tr>
<tr><td>9</td><td>1</td><td>Record Format Version – version for this record type = 00h</td></tr>
<tr><td>10</td><td>1</td><td>Number of Addresses</td></tr>
<tr><td>11</td><td>N</td><td>Address Table Entries
The intent of this table is to store hardware addresses associated with the links described in the Module Signal Interconnect Record. As an example, a dual Ethernet NIC might have 2 MAC addresses defined here (one for Link0 and one for Link1). If the device supports more than one address per link, then this table can be “one-to-many” by having multiple addresses assigned to the same Link ID.
<table>
<tr><th>Offset</th><th>Length</th><th>Definition</th></tr>
<tr><td>0</td><td>1</td><td>Link ID (0 – 3) as enumerated in the Module Signal Interconnect Record</td></tr>
<tr><td>1</td><td>1</td><td>Address Length (L)</td></tr>
<tr><td>2</td><td>L</td><td>Hardware Address
Entries shall be stored as a series of individual bytes, first byte first. These bytes shall be the actual bytes of the address without any "formatting" characters. Example: A mac address could be 00 0a 0b 0c 0d 0e (raw bytes without any colons or other punctuation marks represented)</td></tr>
</table>
…. Additional Table Entries starting at offset 2+L+1</td></tr>
</table>

6.8 Multi-slot Compute Blades

Blades that span multiple chassis slots are allowed. However there are management and hardware constraints on such blades. A multi-slot blade is allowed to use chassis resources (power, interconnect) from each of the slots it occupies.

The Chassis Manager discovers the number of slots occupied by a blade via the *Get Compute Blade Properties* command. Other commands are qualified by relative slot number, allowing for resource negotiation on a per slot basis.

A multi-slot blade **shall** have one management controller that communicates with and represents the blade to the Chassis Manager for activation, and resource negotiation. That controller is associated with the slot with the lowest numbered Slot Identifier and communicates on the BMI via the connections on that slot.

One double-slot blade implementation strategy is to join two single compute blades, either by putting two compute blade boards in a single double-wide blade enclosure, or by making a private connection between two adjacent single-wide blades. In either case, the management controller on the board with the higher Slot Identifier **shall** disable its BMI interface. The connection between the boards **shall** include an IPMB connection, allowing the management controller on the primary blade to communicate with the management controller on the auxiliary blade. An inter-board connection presence indication or some other mechanism should also be provided to allow the controllers to determine that they are in a double-slot configuration.

The management controller on the auxiliary blade, if present, **shall** continue to manage its blade, but it **shall** act as an IPMI satellite controller to the primary blade management controller. The initial negotiation between the controllers on the two boards should take into account that they may take different times to initialize after insertion or chassis power-on.

A multi-slot blade may have other management controllers, but they are considered blade internal and they do not communicate directly with the Chassis Manager.

A multi-slot blade **shall** implement a single *Managed Module Operational State* sensor for itself.

A multi-slot blade **shall** negotiate for chassis resources with the Chassis Manager on a per-slot basis. If a blade does not draw resources from one of its slots, it **shall** indicate this during the negotiation process.

6.9 Managed Module Payload Control

There are often cases where, for administrative or recovery purposes, a blade's payload needs to be reset, rebooted, interrupted, or power cycled. This could be to boot to another operating system, initiate a maintenance operation, or recover from a hang or other problem. An optional IPMI command is defined to support these operations.

6.9.1 Module Payload Control Command

The *Module Payload Control* command provides the ability to reset the payload functionality on the blade. The exact mechanism of implementation is outside of the scope of this specification.

This command is only accepted when the blade is in the *Active* (M4) or *Deactivation Requested* (M5) states.

The operational state of the blade as represented by the Managed Module Operational State sensor is not affected by execution of this command (e.g., a power cycle does NOT cause a transition out of M4).

If this command is implemented, at a minimum, the *Cold Reset* (00h) control option **shall** be supported.

The blade **shall** return a completion code of *Invalid data field in request* (CCh) when receiving a request for an unsupported payload control option.

Table 6-25: Module Payload Control Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.
	2	FRU Device ID (**shall** be 0)
	3	Payload ID. Identifies payload instance on the FRU Device ID. 00h = All payload instances (if any) are operated upon.
	4	Payload Control options 00h = Cold Reset 01h = Warm Reset 02h = Graceful reboot 03h = Issue Diagnostic interrupt 04h = Power Cycle 05h-FFh = Reserved The Payload ID field is ignored when Payload Control option field is set to either 03h or 04h and the operation is initiated for entire module identified by the FRU Device ID.

Field	Byte	Content / Purpose
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h **shall** be used.

6.10 Ethernet-based Management Services

Compute blades support Ethernet-based management services implemented by the blade management controller that allow for remote access to certain blade payload interfaces. These include:

- **Serial Console Redirection**

  This service provides access to the blade payload serial console interface allowing for access to BIOS screens, operating system messages, and other important information. Multiple serial interfaces may be supported.

- **Keyboard / Video / Mouse Redirection**

  This service allows remote out of band access to the blade's video output, keyboard and mouse interfaces.

- **Storage Media Redirection**

  This service allows remote files and storage devices to be used by the blade payload's BIOS and operating system as if they were local, allowing for remote boot and installation capabilities without BIOS or operating system specific support.

In addition, vendors may define additional Ethernet-based management services.

This specification defines discovery, identification, and management interfaces for Ethernet-based management services allowing a Chassis Manager to determine a Compute Blade's service capabilities so as to advertise them to external management clients.

6.10.1 Service Identification and Discovery

In order to support blade implementations with different sets of provided services and services with different or proprietary implementations, services need to be discoverable and identifiable.

Ethernet-based management services are identified by the SSI defined class, network transport protocol, transport port number(s), and high-level protocol. In addition, a service has a vendor specified ASCII string Service Name associated with it.

Each defined service has a Service Index.

Multiple instantiations of the same service class and protocol may be defined. Each has its own Service Index, must be offered at a unique transport port number (or set of numbers as appropriate), and **shall** have a different Service Name.

If multiple service entries (different Service Indices) have the same Service Name and differ only in their transport types and transport port numbers, then these entries must be considered the same service with multiple service access points. For example, the same service may be offered via TCP and TLS/TCP.

6.10.1.1 Service Classes

The following table specifies the defined Ethernet Service Classes.

Table 6-26: Ethernet Service Classes

Class Name	Class Number
Serial Redirection	0
KVM Redirection	1
Storage Redirection	2
Vendor Defined	3
Reserved	4-255

6.10.1.2 Transports

The Ethernet Service Transports are defined in the following table. Transport port numbers are in the range of 0-65535 (2-byte values).

Table 6-27: Ethernet Service Transports

Transport Name	Transport Number
IPv4 UDP	0
IPv4 TCP	1
IPv4 TLS/TCP	2
IPv4 IPsec	3
Reserved	4-255

6.10.1.3 Protocols

Protocols are defined by various bodies and some protocols are defacto standards without standards body sponsorship. Due to this situation, this specification defines service protocols as a pair consisting of the scoping entity,

identified by the IANA Enterprise number, and protocol number, as defined by that entity.

The IANA Enterprise number "0" represents the protocols registered with IANA in the port number assignment registry (http://www.iana.org/assignments/port-numbers). For example, if the Enterprise number for a protocol is "0" and the protocol number is 80 (decimal), then the protocol referenced is "http", as specified in the above referenced document. If multiple protocol numbers (ports) are allocated to a specific protocol, the lowest numbered value **shall** be used. Note that the actual port number and transport used are specified separately, as indicated in the previous sections. For instance, one could define IANA enterprise 0 (std), protocol number 80 (http), port number 20080 (meaning http protocol listening on port 20080).

6.10.1.4 Sessions

A service may support multiple simultaneous sessions. The maximum number supported by a particular service is available via the *Get Service Info* command. Sessions are identified by per service identifiers from 1 to <Max Sessions> for that service. These identifiers are used by the Service State sensors as well as the *Stop Service Session* command.

6.10.1.5 Service Applets

The BMC software vendor must provide a Service Applet Package that the Chassis Manager will use to serve the Applets to the Management Client. The specification provides an IPMI command that allows a Compute Blade to provide a URI. The URI must point back to the device and allow for either the http, https, or tftp protocol to retrieve the applet. It must be in the standard format of:

<http|https|ftp|tftp>://<bmc ip address[:port]>/<path>/applets.zip

The URI above **shall not** require authentication and in the case of ftp, the user access should be anonymous. The applet invocation parameters and overall applet model details are defined in the Service Applet Package Definition (section 6.11).

Applets must be either signed or launched in such a way that the applet may talk directly back to the blade IP address if launched from the Chassis Manager. For Java Applets, this typically means that the Applet must be signed with a CA authorized certificate. The purpose of this is to support architectures where the Chassis Manager launches the applet, but the applet does not proxy or port-forward through the chassis manager. In this design case, the applet would communicate directly to the blade rather than the Chassis Manager.

6.10.2 Service Security

This specification defines a Single Sign-on scheme (in section 6.10.3) for the purpose of authentication and authorization.

Blade management services may encrypt the data between the client applet and the BMC.

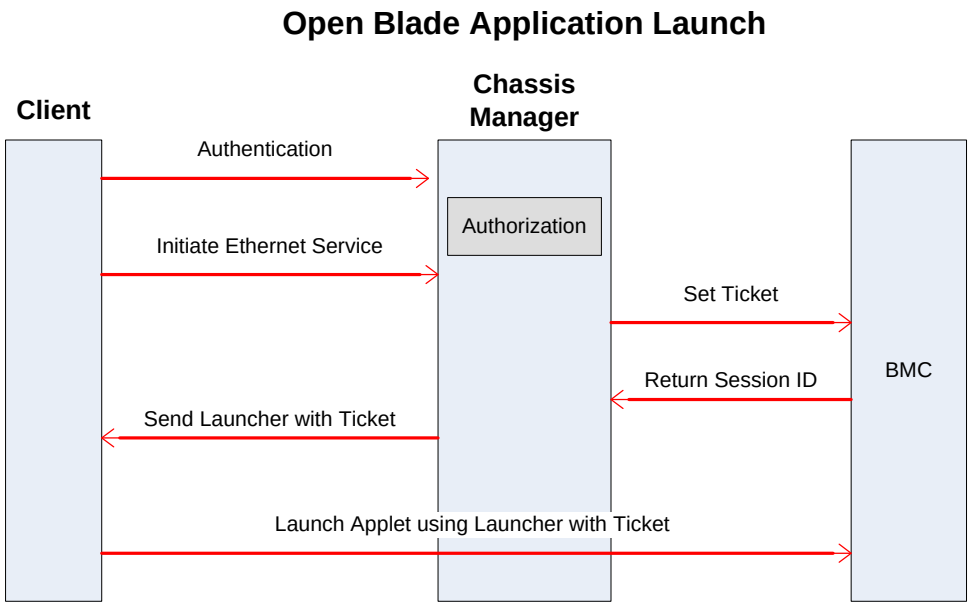
Some secure transports (e.g., TLS) require that a service server provide a digital certificate to provide keying material for encryption and authentication purposes. All such certificates are self-managed and no explicit management support is defined or provided by this specification.

6.10.3 Compute Blade Application Launch

The Compute Blade Application Launch (CBAL) protocol defines a Security Ticket Protocol that allows chassis external management clients to, in effect, be pre-authenticated and authorized, by the Chassis Manager, thereby allowing these clients to avoid authenticating themselves to both the Chassis Manager and the Compute Blades.

The key to this protocol is a secure time-limited ticket or "cookie" that is created by the Chassis Manager and transferred to both the client applet and the Compute Blade management controller. When the client initiates communication with a Compute Blade service that implements this protocol, it passes the ticket to the Compute Blade. It is recommended that the applet and BMC use a secure method (like CHAP) to communicate the ticket, authenticate itself, and, if desired, establish a session encryption key before initiating actual service communication/activities.

The Ethernet Services API allows services to be flagged as implementing the Security Ticket Protocol and defines an IPMI command to allow a ticket to be associated with a service.



6.10.4 IPMI Management of Ethernet Services

The following SSI IPMI commands are defined to allow management services to be enumerated, enabled or disabled globally, and enabled or disabled on a per-user basis.

6.10.4.1 Get Service Info

Compute blade Ethernet-based management services are enumerated via the *Get Service Info* command. The Chassis Manager issues this command repeatedly until it has enumerated all the blade provided services. A completion code of 0xCBh means the requested record does not exist and, therefore, indicates the end of the list.

Table 6-28: Get Service Info Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI defined extension command. A value of 02h must be used.
	2	Service Index
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI defined extension command. A value of 02h must be used.
	3	Service Class This value defines the service as either a specified service type or a Vendor specific service. See section 6.10.1.1 for defined service classes and their values.
	4	Transport This is the base transport used by the service. See section 6.10.1.2 for defined transports and their values.
	5:7	IANA Enterprise Number This number provides the scope/registry of the following protocol number for the service. A value of 000000h means that the Protocol Number should be mapped to the IANA Port-number registry. See section 6.10.1.3.
	8:9	Protocol Number This number identifies the overall protocol implemented by the service.
	10	Protocol Capabilities This field describes the security capabilities of the service, allowing a Chassis Manager or Administrator to determine if the service is compatible with the management deployment model. [7:4] = reserved [3] = Implements CBAL [2] = Provides Privacy (encryption) [1] = Provides Authentication [0] = Provides Authorization
	11	Max Session Support

Field	Byte	Content / Purpose
		This field defines the maximum number of simultaneous sessions supported by this service.
	12-43	Service Name This is a 32 byte,ASCII string providing a name that can be used by user interfaces when displaying information about available services. This name is vendor defined.
	44	Number of Port Numbers in bytes 45-X
	45:-	Port Numbers This is the set of transport specific port numbers that the service is available on. It is a variable length array of 2-byte port numbers (little-endian). The number and order of ports is protocol specific. The quantity of port numbers are specified in the previous byte.

6.10.4.2 Get Applet Package URI

This command retrieves an ASCII URI string defining the scheme and location of an applet binary that can be used by a chassis external client to access the service. If no such applet exists, the command **shall** return the completion code *Requested Sensor, data, or record not present* (CBh).

Table 6-29: Get Applet Package URI Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h must be used.
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h must be used.
	3-34	Applet package version string (up to 32 bytes padded with 0x00 to reach 32 bytes in length)
	35-xxx	Applet Package URI This is an up to 128 byte variable length string providing the location and access scheme for the applet package as defined in 6.10.1.5. <http|https|ftp|tftp>://<ip address[:port]>/<path>/applets.zip

6.10.4.3 Get Service Enable State

This command retrieves the current global enable state of a management service. Each service is referenced as a bit in a bit mask, where a service with Service Index *i* is represented by the i^{th} bit in the bit mask.

Table 6-30: Get Service Enable State Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h must be used.
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h must be used.
	3-6	Service Index State Bitmap This bitmap defines the global enable state for each management service, identified by index. Service with index *i* is represented by the i^{th} bit in the bitmap. Least significant byte first. For each bit: 0 = globally disabled 1 = globally enabled Bit pattern for index i is as indicated in the bytes below: byte: 3 4 5 6 -------- -------- -------- -------- i: 01234567 89111111 11112222 22222233 012345 67890123 45678901

6.10.4.4 Set Service Enable State

This command sets the current global enable state of management services. A globally disabled service is not operational. A globally enabled service is operational and the per-user access state is in force. Each service is referenced as a bit in a bit mask, where a service with Service Index *i* is represented by the i^{th} bit in the bit mask.

Table 6-31: Set Service Enable State Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h must be used.
	2	State Operation [7:6] – Operation 00b = Enable. All services with bitmap bit value = 1 are enabled. Services with bitmap bit value = 0 are not changed. 01b = Disable. All services with bitmap bit value = 1 are disabled. Services with bitmap bit value = 0 are not changed. [5:0] – reserved
	3-6	Service Index Control Bitmap This bitmap defines a service global enable state change. A Service with index *i* is represented by the i^{th} bit in the bitmap. Least significant byte first. For each bit:

Field	Byte	Content / Purpose
		0 = do not modify enable state for this service 1 = perform requested operation for this service Bit pattern for index i is as indicated in the bytes below: byte: 3 4 5 6 -------- -------- -------- -------- i: 01234567 89111111 11112222 22222233 012345 67890123 45678901
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h must be used.
	3-6	Service Index State Bitmap This bitmap defines the current global enable state (after command operation) for all management services, identified by index. Service with index *i* is represented by the i^{th} bit in the bitmap. Least significant byte first. For each bit: 0 = service disabled globally 1 = service enabled globally Bit pattern for index i is as indicated in the bytes below: byte: 3 4 5 6 -------- -------- -------- -------- i: 01234567 89111111 11112222 22222233 012345 67890123 45678901

6.10.4.5 Set Service Ticket

This command associates a Security Ticket with a particular service to allow the service to identify an authenticated external client. This command reserves a Session ID and counts towards the total number of active sessions. The Service Ticket is an opaque 128 byte binary string.

If the maximum number of allowed sessions for the service is already active, the completion code 01h (Device Specific OEM Code. Refer to IPMI Specification Section 5.2) **shall** be returned.

If the service does not implement the Security Ticket Protocol, the completion code 02h (Device Specific OEM Code. Refer to IPMI Specification Section 5.2) **shall** be returned.

Table 6-32: Set Service Ticket Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined

Field	Byte	Content / Purpose
		extension command. A value of 02h must be used.
	2	Service Index
	3	Ticket Expiration Time in seconds
	4-xxx	Service Ticket This is an up to 128 byte variable length string providing a security ticket.
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h must be used.
	3	Service Session ID This is the Session ID that will be used when a client presents the provided ticket.

6.10.4.6 Stop Service Session

Table 6-33: Stop Service Session Command

Field	Byte	Content / Purpose
Request Data	1	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h must be used.
	2	Service Index
	3	Service Session ID This is the session to stop, which may be an in-use session or a session reserved by setting a Security Ticket, but not yet initiated by a client.
	4-xxx	Up to 128 byte variable length message string. This string gives a reason for stopping the session and is intended to be displayed to the user of the session that was stopped.
Response Data	1	Completion Code
	2	SSI Identifier. Indicates that this command is an SSI-defined extension command. A value of 02h must be used.

6.10.4.7 Service State Sensor

A sensor of this type is defined for each service. It provides the activation state for each service session. These sensors are readable and generate events on service session state transitions.

Compact Sensor records (type 2 SDRs) for these sensors **shall** be provided. The sensors **shall** be associated with the specific services via the sensor Entity ID and instance indicated in the SDR. An Entity ID of D0h **shall** be used for these sensors and the Entity Instance **shall** be device-relative where the instance number corresponds to the service index.

For each Service State Sensor, a bitmap **shall** exist that represents up to 4 session states as defined below. The initial state of each session **shall** be 000b.

Table 6-34: Service State Sensor

<table>
<tr><th>Sensor Type</th><th>7-bit Event/ Reading type code</th><th>Event/ Sensor</th><th>Offset/Description</th></tr>
<tr><td>Service State (F0h)</td><td>6Fh</td><td>Discrete</td><td>0 session state bitmap 15
<table>
<tr><td>Session1
bits 1-3
000</td><td>Session2
bits 4-6
000</td><td>Session3
bits 7-9
000</td><td>Session4
bits 10-12
000</td><td>Reserved

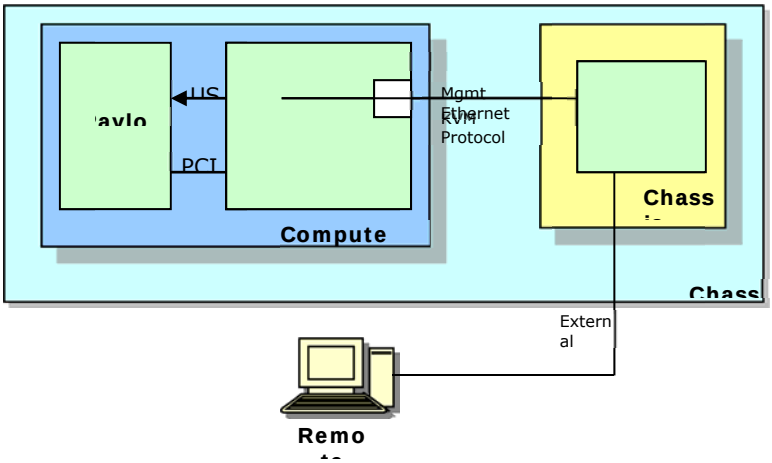
001</td></tr>
</table>
Each Session shall encode the status bits
000 = Inactive
001 = Initializing
010 = Active
011 = Ended Normally
100 = Ticket Expiration
101 = Lost Heartbeat
110 = Forcibly Terminated
111 = Unknown Ticket</td></tr>
</table>

6.10.5 Keyboard/Video/Mouse (KVM) Redirection

A powerful management feature is the redirection of blade payload keyboard, video, and mouse traffic over a network, providing a remote administrator with the ability to interact with software running on the blade, such as firmware (BIOS) or an operating system graphical user interface (GUI).

Mouse and keyboard data are transmitted from the remote management software to the blade for processing by the payload operating system and applications. Video data is captured by the blade management sub-system and forwarded to the remote management software.

Figure 6-12: Typical KVM Implementation



The preceding figure shows a deployment model in which the chassis implements a private management network, either via VLAN or other routing scheme. The Chassis Manager acts as a proxy for setting up a KVM session. A remote management application will request, via the Chassis Manager's external interface(s), that a KVM session be established. The Chassis Manager in turn will establish a KVM session with the appropriate blade and forward KVM data between the blade and remote application.

An SSI Compute Blade management controller **shall** support a KVM service and advertise it via the *Get Service Info* command. This service **shall** have Service Class *KVM* (1).

It is highly recommended that this service be a standard protocol (such as VNC as defined by RealVNC Ltd). It is also highly recommended that this service implement privacy (encryption) as some Chassis Manager implementations may make the KVM data from a blade directly visible on public networks and some deployment models expose the management Ethernet outside the chassis as well.

Since the Chassis Manager can discover the existence and protocol used by the blade KVM service, it can advertise the service to external clients via CIM and, possibly, implementation dependent mechanisms.

6.10.6 Serial Console Redirection

This specification defines a standard mechanism for providing remote access to a blade's payload processor serial console stream (e.g., COM port). This may be implemented internally as a serial connection between a payload serial port and a management controller port, or it may be a "virtual" serial port implemented by the management controller hardware.

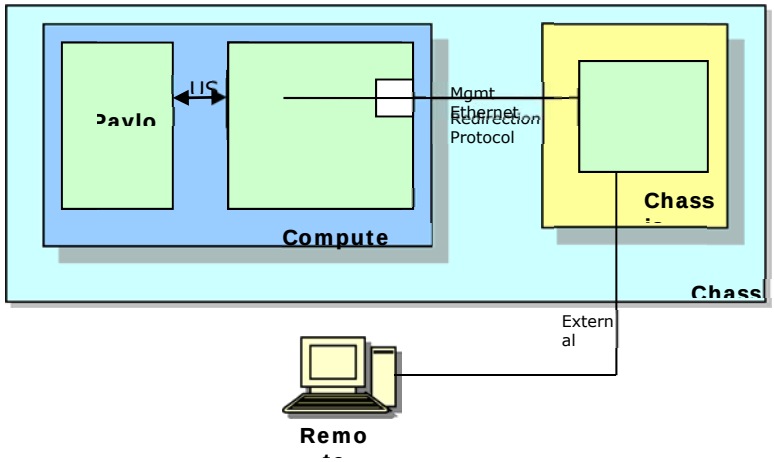
An SSI Compute Blade management controller **shall** support at least one Serial Redirection service and advertise it via the *Get Service Info* command. This service **shall** have Service Class *Serial Redirection* (0).

An SSI Compute Blade management controller may support multiple Serial Redirection services. The service name strings for these services should help a user distinguish between them (e.g., "COM1", "COM2", "Console"). One such service **shall** be considered the *primary* service and that service **shall** have the lowest service index of all defined Serial Redirection services.

6.10.7 Storage Media Redirection

Storage Media Redirection is the ability to make a remote storage media device, such as a CD drive or USB flash drive, available to a blade's payload as if it were a local device, from the blade payload's perspective. This allows an administrator to perform software installations on the blade using the resources at a desktop or other remote system. This virtualization of storage devices can be extended to totally virtualize a drive and so access a remote disk file (e.g., an ISO format CD image file) as a storage media device.

Figure 6-13: Typical Storage Media Redirection Implementation



The preceding figure shows a typical implementation of this service where the Chassis Manager provides a forwarding service allowing external management applications the ability to provide remote media to a blade. Here, the blade management controller implements hardware that allows it to emulate one or more USB target devices to the blade payload system. These are transmitted over the management Ethernet to the Chassis Manager which forwards them to a remote system running an application that can interpret the command stream and emulate a USB storage device.

The Storage Media Redirection services, as defined by this specification, are implemented by a Compute Blade management subsystem emulating one or more mass-storage drives, such as a CD/DVDROM or hard drive, to the payload.

A Compute Blade management controller **shall** implement emulation of and define a media redirection management service for at least one mass storage device. Only read capability is required. A Storage Media Redirection management service **shall** have Service Class *Storage Redirection* (2).

A Storage Media Redirection service represents a separate mass storage device. The type of device a service represents is not fixed, but determined by the remote management application providing the device type data.

Multiple Storage Media Redirection services may be supported by a Compute Blade management controller. Each is represented by a separate management service index.

Multiple simultaneous Storage Media Redirection services may be active on a Compute Blade at any time, but only one is required. A Compute Blade management controller **shall** support simultaneous KVM and Storage Media Redirection sessions.

In order to support booting from redirected media, a Compute Blade management controller **shall** allow Storage Media Redirection establishment before the blade payload has been powered on.

If the Compute Blade implements Storage Media Redirection, the management service **shall** have a Protocol IANA of 7244h (SSI) and a Protocol Number of 0 (Storage Media Redirection).

Although the media redirection protocol is not defined, SCSI over USB is recommended.

6.11 Service Applet Package Definition

The Service Applet Package (SAP) is a zip file stored on the Blade BMC that contains the necessary applets required for the management client to access the BMC ethernet services. The Chassis Manager should maintain an unzipped and up to date version of this package that is accessible by the Management Client through the __BASEURI__ (see Launch File below).

6.11.1 SAP Zip File contents

The zip file **shall** contain the following:

- A config file called config.xml that defines the applets and how they are to be used. It also includes information that may be used to prompt the user for data entry prior to launching the applet.
- A "launch" file for each applet as defined by the config file
- All applet binaries that access the services supported by the Blade.

6.11.2 The config file

The config.xml file **shall** be XML encoded consistent with the following example:

```
<?xml version="1.0"?>
<package version="1.1.0">
    <applet
        name="KVM"
        caption="KVM"
        description="Keyboard Video Mouse"
         launcher="kvmlaunch.htm">
        <service name="kvm" index="1"/>
        <dialog>
            <parameter
                name="__RESOLUTION__"
                label="Resolution"
                datatype="int"
                displaytype="radio"
                default="1">
                <option prompt="Low" value="0"/>
                <option prompt="High" value="1"/>
                <help>Select High or Low resolution</help>
            </parameter>
            <parameter
                name="__MOUSEMODE__"
                label="Mouse Mode"
                datatype="int"
                displaytype="radio"
                default="0">
                <option prompt="Relative" value="0"/>
                <option prompt="Absolute" value="1"/>
                <help>
                    Select the Mouse Mode the OS Requires
                </help>
            </parameter>
        </dialog>
        <help>
            This dialog will allow you to choose how the KVM applet
will run based on your bandwidth and OS requirements.
        </help>
    </applet>
</package>
```

The following DTD **shall** be used to validate config.xml above.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT package (applet)*>
<!ATTLIST package version CDATA #REQUIRED>

<!ELEMENT applet ((service)*,(dialog)*,(help)*)>
<!ATTLIST applet
    name CDATA #REQUIRED
    caption CDATA #REQUIRED
    description CDATA #REQUIRED
    launcher CDATA #REQUIRED>

<!ELEMENT service EMPTY>
<!ATTLIST service
    name CDATA #REQUIRED
    index CDATA #REQUIRED>

<!ELEMENT dialog ((parameter)*,(help)*)>
<!ELEMENT parameter ((option)*,(help)*)>
<!ATTLIST parameter
    name CDATA #REQUIRED
    label CDATA #REQUIRED
```

```
    datatype (int|string|bool|float) #REQUIRED
    displaytype (hidden|input|select|checkbox|radio|textarea)
#REQUIRED
    default CDATA #REQUIRED>

<!ELEMENT option EMPTY>
<!ATTLIST option
    prompt CDATA #REQUIRED
    value CDATA #REQUIRED>

<!ELEMENT help (#PCDATA)>
```

6.11.3 config.xml Explanation

tag **package**

Envelopes the entire contents and sets a version for this package

attributes:

version: version number of SAP (see 6.10.4.2)

tag **applet** (parent must be package)

add this tag for each applet in package

attributes:

name: variable name for this applet
caption: short description (button label in a UI) for applet
description: long description of applet
launcher: name of file to use for launching applet

tag **service**

references the service index(ices) as defined in section 6.10.4.1

attributes:

name: variable name of the service
index: service index

tag **dialog** (parent must be applet)

Indicates a dialog that a UI should present

attributes: none

tag **parameter** (parent must be dialog)

dialog parameter to be substituted in the launcher

attributes:

name: variable name
label: variable name
datatype: int, string, bool, or float
displaytype: hidden, input, select, checkbox, radio, textarea
default: default value

tag **option** (parent must be parameter)

prompts with "prompt" and assigns selected value

attributes:

prompt: display name to represent the option value
value: value of the option

tag **help**

Attaches help text to the parent element

attributes: none

6.11.4 Launch File

The launch file should have all information required for a browser launched file to properly launch the applet. An example might be a jnlp or html file that contains the proper information for launching the applet binary and passing all of the appropriate parameters.

Prior to serving the launch file to the client, the following substitutions will be made by the Chassis Manager:

__BASEURI__
The location of the (unzipped) package contents with respect to the Management Client (MC).

__TICKET__
The authentication issued by the Chassis Manager to the MC

__IPADDRESS__
IP Address of Chassis Manager

__PORT_<service index>_<port>__
Redirected ports for a given service id (see 6.10.4.1)

__CAPTION__
Short Description (window title) of applet

__DESCRIPTION__
Long Description of applet

__SLOT__
Slot number of Blade

__SESSIONID__
Session ID (see 6.10.4.5)

Additionally, parameter tags in config.xml that are surrounded with __ will also be substituted in the launcher prior to launching the applet.

6.11.5 Applet binaries

Applet binary base filenames **shall** (at a minimum) include version numbers and be prepended with the vendor and blade model name and appended with the applet version number. This is the pattern that must be used.

<vendor>_<blademodel>_<appletname>_<major>.<minor>.<ext> .

This is to prevent caching of old versions and application namespace collisions of the applet in the end users browser.

example: CompanyX_BladeModelY_KVM_2.3.jar

6.12 Blade Management Controller Resets

During operational cycle of a blade, it is possible that management controller(s) on the blade may get reset due to various reasons, including firmware upgrades, firmware watchdog timer reset, etc.

These resets/reboots **shall not** affect the blade's operational state. When the management controller returns to normal operation, it **shall not** affect the blade's operational state or payload functionality and **shall** reestablish the blade's operational and associated state, as defined by this specification, as it was at the time of the reset. The state to be reestablished includes, but is not necessarily limited to the blade's *Managed Module Operational State* sensor state, Chassis Interconnect enable state, power budget and power state.

For example, this means that if a management controller resets while the blade is in the *Active* (M4) state, then when the management controller restarts, the *Managed Module Operational State* sensor for the blade should be initialized to the previous M4 state.

This requirement does not pertain to management controller resets caused by blade insertion/extraction or failure of chassis provided power.

6.13 Chassis Manager-less Operation

This specification does not support Chassis Manager-less deployment models. However, it is possible that a Chassis Manager may fail on chassis power-on or during system operation. This is especially an issue for non-redundant

deployment models, or in the hopefully unlikely case of a double Chassis Manager failure.

A Compute Blade management controller **shall** wait at least 2 minutes after initialization before deciding that no Chassis Manager is available due to total lack of communication. If a Chassis Manager is unavailable, a blade may implement OEM behaviors, such as autonomous activation. However, these should be well thought out in order to avoid inadvertent system failure due to chassis resource over-commitment such as power use beyond the current power domain capacity.

At no point may a blade implement OEM behaviors due to a loss of Chassis Manager communication (as opposed to never having communicated).

6.14 Compute Blade Fault Aggregation

A blade **shall** implement an aggregated fault sensor that represents the overall health of the blade. The choice of sensors and other state to aggregate into this sensor is not specified, but should, at a minimum, include blade temperatures, voltages.

This sensor **shall** support reading via the IPMI *Get Sensor Reading* command.

The blade **shall** provide an SDR for this sensor that enables assertion and de-assertion events for all defined offsets.

This sensor may be used to control the blade *Fault LED* state.

The following table defines the sensor type and event/reading type.

Table 6-35: Blade Aggregated Fault Sensor Definition

Sensor Type	7-bit Event/Reading type code (OEM)	Event/ Sensor	Offset/Description
Module/Board (15h)	7Fh	Discrete	00h = All is OK 01h = Non critical (warning) 02h = Critical 03h = Non recoverable

6.15 Compute Blade Cooling Management

The blade management controller is responsible for instrumentation of thermal sensors on the blade that it represents. As such, the blade may have one or more thermal sensors that may have varying thresholds and operating ranges depending upon hardware design and configuration. When thermal conditions exceed normal operating ranges, the management controller will need to

request additional cooling from the Chassis Manager. This is done by forwarding thermal sensor events to the Chassis Manager. Also, since a blade may occupy multiple slots, and cooling management may be slot specific, the Chassis Manager will need to track thermal state independently for each slot a blade occupies. In addition, although likely to be the exception, it is possible for a chassis to implement multiple cooling zones per module.

To implement a simple and consistent thermal management scheme, the *SSI Compute Blade Specification* defines an aggregate thermal sensor to provide comprehensive blade thermal state. The blade manufacturer chooses which module temperature sensors contribute to the aggregate thermal sensor state, and is free to use other thermal sensor state as well as thermal information that is not otherwise represented by other IPMI sensors. This aggregation frees the Chassis Manager from having to understand which module sensors should affect cooling state. This sensor divides the thermal state into Normal, Non-critical, Critical, and Non-recoverable operating regions. The Normal is divided into a gradient of 8 magnitude values helping the system designer to tune a chassis to be thermally or acoustically optimal (see Table 6-36 and Table 6-37). The sensor event data includes both the module relative slot that the sensor represents as well as the associated cooling zone for that sensor (Table 6-38)

A blade **shall** aggregate the state from all available thermal sensors on a per-occupied slot basis into per-slot separate *Blade Aggregated Thermal Sensors.*

This sensor **shall** support reading via the IPMI *Get Sensor Reading* command.

The blade **shall** provide SDRs for these sensors that enable assertion and de-assertion events for all defined offsets.

The Chassis Manager may use these sensors to drive the chassis cooling subsystem state.

Table 6-36: Blade Aggregated Thermal Magnitude Sensor Definition

Sensor Type	7-bit Event/Reading type code (OEM)	Event/ Sensor	Offset/Description
Temp (01h)	7Fh	Discrete	00h = Off 01h = Cold 02h = Cool 03h = Warm 04h = Hot 05h = Hotter 06h = Warning 07h = Critical

Table 6-37: Thermal State Conditions and Values

Value	State Description
0	Off The blade will set this value when the payload power is off.
1	Cold The blade is well below normal operating temperature.
2	Cool The blade is operating at a temperature that is most optimal for hardware longevity and stability. The CMM should attempt to hold the blade at this level if acoustics are not a factor and hardware stability and longevity is the only concern. (normal)
3	Warm The blade is operating at a temperature that is within an acceptable envelope at the high end and may sustain operations at this temperature. The CMM should attempt to hold the blade at this level for systems attempting to be acoustically optimized. This level is the optimal balance between acoustics and stability. (normal)
4	Hot The blade is operating at a temperature slightly above Warm and outside of the required range of sustained reliability. The CMM should increase the cooling system (fans) to reduce the temperature to 2 or 3. (normal)
5	Hotter The blade is operating at the high edge of the threshold yet still in a non-critical state. Any increase in temperature will cause a degraded state. The CMM should increase the cooling system (even more so than level 4) to reduce the temperature to 2 or 3. (non-critical)
6	Warning The blade is operating outside of the proper operating range and may already be experiencing a degraded performance. If available, the blade should "throttle down" to reduce temperature. (non-critical) The CMM should increase cooling to 100%.
7	Critical State -- action required The blade is operating at a temperature that hardware damage may be imminent. The blade should attempt to shut down. The CMM should attempt to shut down the blade in addition to increase cooling to 100%. (critical)

Table 6-38: Blade Aggregated Thermal Sensor Event Data

Field	Content / Purpose
Event Type /Direction	[7] - Event Direction = *0b (Assertion)* or *1b (Deassertion)* [6:0] - Event Type = 7Fh (OEM)
Event Data 1	[7:4] - previous state and/or severity in Event Data 2, OEM code in Event Data 3 = 6h. [3:0] - Current State.
Event Data 2	[7:4] - Optional offset from 'Severity' event/reading code (or Fh) [3:0] - Previous State.
Event Data 3	[7:4] - Reserved [3:0] - Relative slot number for associated sensor. 0 = primary.

6.16 Managed Face Plate Indicators and Switches

The blade face plate provides users with a direct view of the blade's activation and fault state. It also provides a way for a user to interact with the blade for the purpose of maintenance operations – de-activating a blade in preparation for extraction or causing a blade to re-activate after a requested or forced shutdown.

6.16.1 LEDs

This section describes the behavior of management controlled face plate LEDs.

Power LED

The Power LED provides a user with a view of the blade's activation state. It shows whether the blade is inactive, active, or transitioning between these states and can provide a user pressing the activation request button feedback as to the effect of his request.

The *Power, Storage, and Fault* table in section 9.4.1 provides a mapping of LED behavior to generic blade state. The mapping of power LED state in that table to activation state is as follows:

Table 6-39: Power LED Activation State Mapping

Power LED State	Activation (Module Operational) State	Behavior
Off	M1	Off
Transition	M2, M3, M5, M6	Slow blink
Standby/Sleep	M4	Spaced blink
Power + Activity	M4	Steady on

Fault State LED

The Fault State LED provides users with a visual indication of the overall health of the blade. It provides an aggregation of fault state from various blade sensors as chosen by the OEM. The *Power, Storage, and Fault* table in section 9.4.1 provides a mapping of fault state to LED behavior.

A blade **shall** implement the IPMI *Chassis Identify* command including the *Force Identify On* feature. This command will assert the identify state of the Fault State LED and overrides any other displayed fault state. When the identify function is turned off or times-out, the Fault State LED will display the current blade fault state.

6.16.2 Activation / Deactivation Request Button

This is sometimes called the *Power Button* or *Power On Button*. However, this button does not directly affect blade power. The button is monitored by the blade's management controller and button presses are considered requests to change blade *Managed Module Operational State*. Button presses cause the state of the *Activation Policy* state variable to toggle, potentially causing a change of *Operational State.*

For example, if a blade were in the *Active* (M4) state, the *Activation Policy* state variable would be *Activate* causing the blade to stay in that state. A press of the Activation/Deactivation Request button would toggle the *Activation Policy* state variable to the *Deactivate* value, causing the blade to transition to the *Deactivation Requested* (M5) state. Further state transitions would depend on Chassis Manager action.

Similarly, if a blade were in the *Inactive* (M1) state after having been deactivated, a press of the Activation/Deactivation Request button would toggle the *Activation Policy* state variable to the *Activate* value, causing the blade to transition to the *Activation Requested* (M2) state. Further state transitions would depend on Chassis Manager action, or another button press would return the blade to the *Inactive* (M1) state.

If a button press causes an operational state change, the sensor event generated by the blade *Managed Module Operational State* sensor **shall** indicate a state change cause value of *State Change due to operator pressing the hot swap push button* (3h).

If an Activation/Deactivation Request button is implemented, a blade should also implement the IPMI *Set Front Panel Button Enables* command to allow management software to control its use, for physical security purposes.

6.17 Compute Blade Resource Update

Blades often have several components that can be field updateable or upgradeable. This includes management controller firmware, FPGA configuration, and various configuration devices such as EEPROMs.

This specification does not define any mechanisms to affect this update. However, there are some requirements relative to the way such updates can affect blade operation. See section 6.10 for requirements.

A blade should implement IPMI *Version Change* (2Bh) sensors for updateable entities to allow any subsequent failures to be correlated with updates.

6.18 Required Management Connectivity

The following signals are required for management connectivity on the compute blade:

- IPMB_A signals - SMB_SDA_A, SMB_SCL_A
- IPMB_B signals - SMB_SDA_B, SMB_SCL_B
- Slot Identifier signals - BLSLOT_ID_[0:5]
- Blade Present signal - BLPRES_N

These signals **shall** be supported by each blade. See section 3.4.1 for connector pin details.

6.19 Compute Blade IPMI Command Support

This section specifies the implementation requirements for both IPMI and SSI-defined commands that are implemented by the compute blades and Chassis Manager. Some commands are mandatory in certain contexts and optional otherwise.

6.19.1 IPMI v2.0 Defined Commands

In the following table:

- *Chassis Manager Req'd* and *Blade Req'd* columns
  O = optional command
  TX = required to transmit the command
  RX = required to receive/accept the command
- Footnotes define exceptions and conditions and are at the end of the table.
- For optional commands: Unless otherwise specified, blade optional commands are Rx and Chassis Manager optional commands are Tx.
- All Blade commands that are Rx and implemented optional commands shall be accepted over the Primary Ethernet Interfaces and the BMI.

Table 6-40: IPMI v2.0 Defined Commands

Commands	IPMI v2.0 Spec Section	NetFn	CMD	Chassis Manager Req'd	Blade Req'd
IPM Device "Global" Commands					
reserved	-	App	00h	-	-
Get Device ID	20.1	App	01h	Tx / Rx	Rx
Broadcast 'Get Device ID'[1]	20.9	App	01h	Tx / Rx	Rx
Cold Reset	20.2	App	02h	O	O
Warm Reset	20.3	App	03h	O	O
Get Self Test Results	20.4	App	04h	Tx	Rx
Manufacturing Test On	20.5	App	05h	O	O
Set ACPI Power State	20.6	App	06h	O	O
Get ACPI Power State	20.7	App	07h	O	O
Get Device GUID	20.8	App	08h	Tx / Rx	Rx
reserved	-	App	09h-0Fh	-	-
BMC Watchdog Timer Commands					
Reset Watchdog Timer	27.5	App	22h	NA	O[1]
Set Watchdog Timer	27.6	App	24h	NA	O[1]
Get Watchdog Timer	27.7	App	25h	NA	O[1]
BMC Device and Messaging Commands					
Set BMC Global Enables	22.1	App	2Eh	O	O
Get BMC Global Enables	22.2	App	2Fh	O	O
Clear Message Flags	22.3	App	30h	O	O[1]
Get Message Flags	22.4	App	31h	O	O[1]
Enable Message Channel Receive	22.5	App	32h	O	O
Get Message	22.6	App	33h	O	O[1]
Send Message	22.7	App	34h	O	O[1]
Read Event Message Buffer	22.8	App	35h	O	O
Get BT Interface Capabilities	22.10	App	36h	O	O
Get System GUID	22.14	App	37h	O	O
Get Channel Authentication Capabilities	22.13	App	38h	O	O[3]
Get Session Challenge	22.15	App	39h	O	O[3]
Activate Session	22.17	App	3Ah	O	O[3]
Set Session Privilege Level	22.18	App	3Bh	O	O[3]
Close Session	22.19	App	3Ch	O	O[3]

Commands	IPMI v2.0 Spec Section	NetFn	CMD	Chassis Manager Req'd	Blade Req'd
Get Session Info	22.20	App	3Dh	O	O[3]
unassigned	-	App	3Eh	-	-
Get AuthCode	22.21	App	3Fh	O	O
Set Channel Access	22.22	App	40h	O	Rx
Get Channel Access	22.23	App	41h	O	Rx
Get Channel Info Command	22.24	App	42h	O	Rx
Set User Access Command	22.26	App	43h	O	O[3]
Get User Access Command	22.27	App	44h	O	O[3]
Set User Name	22.28	App	45h	O	O[3]
Get User Name Command	22.29	App	46h	O	O[3]
Set User Password Command	22.30	App	47h	O	O[3]
Activate Payload	24.1	App	48h	O	O
Deactivate Payload	24.2	App	49h	O	O
Get Payload Activation Status	24.4	App	4Ah	O	O
Get Payload Instance Info	24.5	App	4Bh	O	O
Set User Payload Access	24.6	App	4Ch	O	O
Get User Payload Access	24.7	App	4Dh	O	O
Get Channel Payload Support	24.8	App	4Eh	O	O
Get Channel Payload Version	24.9	App	4Fh	O	O
Get Channel OEM Payload Info	24.10	App	50h	O	O
unassigned	-	App	51h	-	-
Master Write-Read	22.11	App	52h	O	O
unassigned	-	App	53h	-	-
Get Channel Cipher Suites	22.15	App	54h	O[2]	O[2]
Suspend/Resume Payload Encryption	24.3	App	55h	O	O
Set Channel Security Keys	22.25	App	56h	O	O[3]
Get System Interface Capabilities	22.9	App	57h	O	O
Chassis Device Commands					
Get Chassis Capabilities	28.1	Chassis	00h	O	O
Get Chassis Status	28.2	Chassis	01h	O	O
Chassis Control	28.3	Chassis	02h	O	O
Chassis Reset	28.4	Chassis	03h	O	O
Chassis Identify	28.5	Chassis	04h	Tx / Rx	Rx
Set Chassis Capabilities	28.7	Chassis	05h	O	O

Commands	IPMI v2.0 Spec Section	NetFn	CMD	Chassis Manager Req'd	Blade Req'd
Set Power Restore Policy	28.8	Chassis	06h	O	O
Get System Restart Cause	28.11	Chassis	07h	O	O
Set System Boot Options	28.12	Chassis	08h	Tx	Rx
Get System Boot Options	28.13	Chassis	09h	O	O
Set Front Panel Button Enables	28.6	Chassis	0Ah	O	O
Set Power Cycle Interval	28.9	Chassis	0Bh	O	O
unassigned	-	Chassis	0Ch-0Eh	-	-
Get POH Counter	28.14	Chassis	0Fh	O	O
Event Commands					
Set Event Receiver	29.1	S/E	00h	Tx	Rx
Get Event Receiver	29.2	S/E	01h	Tx / Rx	Tx / Rx
Platform Event (a.k.a. "Event Message")	29.3	S/E	02h	Rx	Tx
unassigned	-	S/E	03h-0Fh	-	-
PEF and Alerting Commands					
Get PEF Capabilities	30.1	S/E	10h	O	O
Arm PEF Postpone Timer	30.2	S/E	11h	O	O
Set PEF Configuration Parameters	30.3	S/E	12h	O	O
Get PEF Configuration Parameters	30.4	S/E	13h	O	O
Set Last Processed Event ID	30.5	S/E	14h	O	O
Get Last Processed Event ID	30.6	S/E	15h	O	O
Alert Immediate	30.7	S/E	16h	O	O
PET Acknowledge	30.8	S/E	17h	O	O
Sensor Device Commands					
Get Device SDR Info		S/E	20h	Tx	Rx
Get Device SDR		S/E	21h	Tx	Rx
Reserve Device SDR Repository		S/E	22h	Tx	Rx
Get Sensor Reading Factors		S/E	23h	O	O
Set Sensor Hysteresis		S/E	24h	O	O
Get Sensor Hysteresis		S/E	25h	Tx	Rx
Set Sensor Threshold		S/E	26h	O	O
Get Sensor Threshold		S/E	27h	Tx	Rx
Set Sensor Event Enable		S/E	28h	O	O
Get Sensor Event Enable		S/E	29h	O	O

Commands	IPMI v2.0 Spec Section	NetFn	CMD	Chassis Manager Req'd	Blade Req'd
Re-arm Sensor Events		S/E	2Ah	O	O
Get Sensor Event Status		S/E	2Bh	O	O
Get Sensor Reading		S/E	2Dh	Tx	Rx
Set Sensor Type		S/E	2Eh	O	O
Get Sensor Type		S/E	2Fh	O	O
FRU Device Commands					
Get FRU Inventory Area Info		Storage	10h	Tx / Rx	Tx / Rx
Read FRU Data		Storage	11h	Tx / Rx	Tx / Rx
Write FRU Data		Storage	12h	O	O
SDR Device Commands					
Get SDR Repository Info		Storage	20h	O	O
Get SDR Repository Allocation Info		Storage	21h	O	O
Reserve SDR Repository		Storage	22h	O	O
Get SDR		Storage	23h	O	O
Add SDR		Storage	24h	O	O
Partial Add SDR		Storage	25h	O	O
Delete SDR		Storage	26h	O	O
Clear SDR Repository		Storage	27h	O	O
Get SDR Repository Time		Storage	28h	O	O
Set SDR Repository Time		Storage	29h	O	O
Enter SDR Repository Update Mode		Storage	2Ah	O	O
Exit SDR Repository Update Mode		Storage	2Bh	O	O
Run Initialization Agent		Storage	2Ch	O	O
SEL Device Commands					
Get SEL Info		Storage	40h	Tx / Rx	Tx / Rx
Get SEL Allocation Info		Storage	41h	Tx / Rx	Tx / Rx
Reserve SEL		Storage	42h	Tx / Rx	Tx / Rx
Get SEL Entry		Storage	43h	Tx / Rx	Tx / Rx
Add SEL Entry		Storage	44h	O	Rx
Partial Add SEL Entry		Storage	45h	O	O
Delete SEL Entry		Storage	46h	O	O
Clear SEL		Storage	47h	O	Rx
Get SEL Time		Storage	48h	Tx / Rx	Tx / Rx
Set SEL Time		Storage	49h	O	Rx

Commands	IPMI v2.0 Spec Section	NetFn	CMD	Chassis Manager Req'd	Blade Req'd
Get Auxiliary Log Status		Storage	5Ah	O	O
Set Auxiliary Log Status		Storage	5Bh	O	O
LAN Device Commands					
Set LAN Configuration Parameters		Transport	01h	Tx	Rx
Get LAN Configuration Parameters		Transport	02h	Tx	Rx
Suspend BMC ARPs		Transport	03h	O	O
Get IP/UDP/RMCP Statistics		Transport	04h	O	O
Serial/Modem Device Commands					
Set Serial/Modem Configuration		Transport	10h	O	O
Get Serial/Modem Configuration		Transport	11h	O	O
Set Serial/Modem Mux		Transport	12h	O	O
Get TAP Response Codes		Transport	13h	O	O
Set PPP UDP Proxy Transmit Data		Transport	14h	O	O
Get PPP UDP Proxy Transmit Data		Transport	15h	O	O
Send PPP UDP Proxy Packet		Transport	16h	O	O
Get PPP UDP Proxy Receive Data		Transport	17h	O	O
Serial/Modem Connection Active		Transport	18h	O	O
Callback		Transport	19h	O	O
Set User Callback Options		Transport	1Ah	O	O
Get User Callback Options		Transport	1Bh	O	O
SOL Activating		Transport	20h	O	O
Set SOL Configuration Parameters		Transport	21h	O	O[5]
Get SOL Configuration Parameters		Transport	22h	O	O[5]
Bridge Management Commands (ICMB)					
Get Bridge State	[ICMB]	Bridge	00h	O	O
Set Bridge State	[ICMB]	Bridge	01h	O	O
Get ICMB Address	[ICMB]	Bridge	02h	O	O
Set ICMB Address	[ICMB]	Bridge	03h	O	O
Set Bridge Proxy Address	[ICMB]	Bridge	04h	O	O
Get Bridge Statistics	[ICMB]	Bridge	05h	O	O
Get ICMB Capabilities	[ICMB]	Bridge	06h	O	O
Clear Bridge Statistics	[ICMB]	Bridge	08h	O	O
Get Bridge Proxy Address	[ICMB]	Bridge	09h	O	O
Get ICMB Connector Info	[ICMB]	Bridge	0Ah	O	O

Commands	IPMI v2.0 Spec Section	NetFn	CMD	Chassis Manager Req'd	Blade Req'd
Get ICMB Connection ID	[ICMB]	Bridge	0Bh	O	O
Send ICMB Connection ID	[ICMB]	Bridge	0Ch	O	O
Discovery Commands (ICMB)					
PrepareForDiscovery	[ICMB]	Bridge	10h	O	O
GetAddresses	[ICMB]	Bridge	11h	O	O
SetDiscovered	[ICMB]	Bridge	12h	O	O
GetChassisDeviceId	[ICMB]	Bridge	13h	O	O
SetChassisDeviceId	[ICMB]	Bridge	14h	O	O
Bridging Commands (ICMB)[8]					
BridgeRequest	[ICMB]	Bridge	20h	O	O
BridgeMessage	[ICMB]	Bridge	21h	O	O
Event Commands (ICMB) [8]					
GetEventCount	[ICMB]	Bridge	30h	O	O
SetEventDestination	[ICMB]	Bridge	31h	O	O
SetEventReceptionState	[ICMB]	Bridge	32h	O	O
SendICMBEventMessage	[ICMB]	Bridge	33h	O	O
GetEventDestination (optional)	[ICMB]	Bridge	34h	O	O
GetEventReceptionState (optional)	[ICMB]	Bridge	35h	O	O
OEM Commands for Bridge NetFn					
OEM Commands	[ICMB]	Bridge	C0h-FEh	O	O
Other Bridge Commands					
Error Report (optional)	[ICMB]	Bridge	FFh	O	O

[1] Required if a payload management interface is implemented.

[2] Chassis Manager Tx required if Chassis Manager supports Ethernet-based management Blade Rx required if blade supports Ethernet-based management.

[3] Required for LAN Session support.

[4] Required if an IPMI RMCP+ Payload-based feature is supported.

[5] Required if an IPMI Serial Over LAN is supported.

6.19.2 SSI-defined IPMI Commands

Table 6-41: SSI-defined Commands

Command	Section	NetFn	Cmd	Chassis Manager Req'd	Blade Req'd
Get Compute Blade Properties		SSI	00h	Rx/Tx	Tx/Rx
Get Address Information		SSI	01h	Rx/Tx	Tx/Rx
SSI Platform Event Message		SSI	02h	Rx	Tx
Module BMI Control		SSI	03h	Tx	Rx
Module Payload Control		SSI	04h	Tx	Rx
Set System Event Log Policy		SSI	05h	O	O
Set Module Activation Policy		SSI	0Ah	Tx	Rx
Get Module Activation Policy		SSI	0Bh	Tx	Rx
Set Module Activation		SSI	0Ch	Tx	Rx
Set Power Level		SSI	11h	Tx	Rx
Get Power Level		SSI	12h	Tx	Rx
Renegotiate Power		SSI	13h	Rx	Tx
Get Service Info		SSI	16h	Tx	Rx
Get Applet Package URI		SSI	17h	Tx	Rx
Get Service Enable State		SSI	18h	Tx	Rx
Set Service Enable State		SSI	19h	Tx	Rx
Set Service Ticket		SSI	20h	Tx	Rx
Stop Service Session		SSI	21h	Tx	Rx

[1]Note that the actual net function is 2Ch (Group Extension) with group ID 02h (SSI Compute Blade).

6.19.3 Command Groupings and Privilege Levels

This specification divides commands into two groups:

- Informative - Commands providing information about state and configuration
- Administrative - Commands used to change state or configuration, or view private state (e.g., keying material).

6.19.3.1 IPMI v2.0 Specification Commands

For IPMI defined commands, the IPMI *User* and *Callback* privilege levels map to the *Informative* group. The IPMI *Operator* and *Administrator* privilege levels map to the *Administrative* group

6.19.3.2 SSI-defined Commands

Table 6-42: SSI-defined Command Grouping

Command	Group	IPMI Privilege Level
Get Compute Blade Properties	Informative	User
Get Address Information	Informative	User
SSI Platform Event Message	Informative	User
Module BMI Control	Administrative	Operator
Module Payload Control	Administrative	Operator
Set System Event Log Policy	Administrative	Operator
Set Module Activation Policy	Administrative	Operator
Get Module Activation Policy	Informative	User
Set Module Activation	Administrative	Operator
Set Power Level	Administrative	Operator
Get Power Level	Informative	User
Renegotiate Power	Informative	User

All Administrative and Informative commands **shall** be accepted by blades over any secure management interface (e.g., BMI, admin authenticated LAN session), while Informative commands shall be accepted over all management interfaces that a blade implements. See section 6.3.3.3 for related information.

7 Compute Blade BIOS Requirements

This section specifies the BIOS requirements for SSI Compute Blades. These requirements define common interfaces and standards for compute blades to achieve inter-operate capability among different vendor blades and Chassis Managers. The requirements include industry standards, technologies, and interfaces (such as ACPI, SMBIOS), and additional requirements for common manageability, for better reliability, availability and serviceability (RAS), fault handling, and provisioning of blades.

7.1 BIOS Interfaces

The key functions of the BIOS are to perform all platform hardware initialization, such as hardware components (e.g. chipset, CPU, memory, etc), initialize I/O busses, I/O devices (e.g. VGA, LAN, SAS, etc) using device vendors option ROMs, and provide BIOS runtime interfaces and industry-standard interfaces to assist OS loading and operation.

7.1.1 Legacy BIOS

The blade BIOS **shall** support the legacy BIOS interfaces for IA32 architecture-based blades. Legacy BIOS interfaces include support for legacy option ROMs, booting via legacy interrupt 0x19 and all runtime legacy BIOS interrupt interfaces, such as Int 13, Int 1A, Int 15 and Int 16 services. The blade BIOS **shall** also support industry standard specification such as ACPI, BBS, SMBIOS and MPS in the legacy BIOS.

7.1.2 UEFI

The blade BIOS should have Unified Extensible Firmware Interface (UEFI) version 2.0 or above on IA32 architecture based blades.

On 64-bit architecture-based blades, the blade BIOS **shall** support UEFI interface version 2.0 or above. UEFI support provides extended data interface tables and extended boot and runtime interface functions for the operating system loader and operating system.

An UEFI-compatible BIOS also supports EBC (EFI Byte Code) drivers for I/O devices that are processor architecture independent, such as LAN and storage, rather than the legacy option ROM. The UEFI-compatible BIOS can implement a pre-boot shell environment and run blade maintenance applications and tools, such as a BIOS update.

7.2 Compute Blade Error Handling

System component error handling and reporting is essential to improve reliability and reduce down time. Error monitoring allows predicting component failures.

The compute blade BIOS **shall** support handling and reporting of correctable, recoverable, uncorrectable, and fatal errors in platform components. At a minimum, the blade BIOS **shall** handle and report errors on the memory and the PCI / PCI Express sub-system. The blade BIOS **shall** log any of these platform errors into the IPMI System Event Log (SEL).

Note: System events logged by the BIOS will be forwarded to the Chassis Manager by the blade management controller over BMI interface.

7.2.1 Uncorrectable / Fatal Errors

The blade BIOS **shall** handle and log uncorrectable and fatal errors to the IPMI SEL. In addition to logging, the blade BIOS **shall** contain uncorrectable and fatal errors to prevent any data corruption. The blade BIOS should assert or should allow the blade hardware to generate a critical error interrupt after logging the uncorrectable or fatal error.

The critical error interrupt will allow coordinating error containment with the operating system. The critical error interrupt may be a non-maskable interrupt, a machine check architecture (MCA) interrupt, or a similar critical event that the operating system can handle. If an uncorrectable or fatal error causes the blade to hang before the error can be logged, then the blade BIOS should attempt to identify the error by analyzing blade hardware on the next boot and log the error to SEL.

7.2.2 Correctable Errors

The blade BIOS **shall** monitor and log correctable errors to the IPMI SEL. The blade BIOS **shall** implement a threshold method for correctable errors. The threshold method can be implementation-specific. It can be a threshold limit count, an error rate limit, or a leaky-bucket count. Once the threshold is reached, the BIOS **shall** stop further error reporting to SEL after reporting a threshold reached event to the SEL. The blade BIOS **shall** restart logging correctable errors when the blade is rebooted.

7.2.3 SEL Format and Data

The section specify the SEL records that blade BIOS **shall** support to log errors in a system component. The Blade BIOS **shall** populate SEL data as specified in Table 7-1 and Table 7-2. These tables specify SEL data format for key component errors and will allow to decode the SEL log and identify the blade component failure by Chassis manager or application independent of blade manufacturer. Note that Sensor numbers for these SEL events are

implementation specific, but must be unique for the events specified in Table 7-2.

Table 7-1: SEL Record Generator ID

Field	IPMI Definition	BIOS Implementation
Generator ID	7:1 System software ID or IPMB slave address. 1 = ID is system software ID 0 = ID is IPMB slave address	If BIOS is the source: Bit 7:4 0x3 for the system BIOS. Bit 3:1 1 = Format revision of the data format for OEM data bytes 2 and 3, For this revision of the specification, set this field to 1. All other revisions are reserved. Bit 0 1 = ID is system software ID. As a result, the generator ID byte will start from 0x31 and go up to 0x3f, in increments of 2 for events logged by the BIOS.

Table 7-2: SEL Records for Component Errors

Error Type	Sensor Type	Event/Reading Type Code	OEM Data1	OEM Data2	OEM Data3
Memory Uncorrectable or Fatal	0x0C	0x6F	0x21 OEM code in data3 & Uncorrectable Error	0xFF	bit[7:6] = Memory array Number (e.g. memory board) bit[5:0] = DIMM Slot Number
Memory Correctable or recoverable	0x0C	0x6F	0x20 OEM code in data3 & Correctable Error	0xFF	bit[7:6] = Memory array Number (e.g. memory board) bit[5:0] = DIMM Slot Number
Memory Correctable threshold reached	0x0C	0x6F	0x25 OEM code in data3 & Correctable Error threshold reached	0xFF	bit[7:6] = Memory array Number (e.g. memory board) bit[5:0] = DIMM Slot Number
Memory Correctable logging disabled	0x10	0x6F	0x80 OEM code in data2 & Correctable Error logging disabled	bit[7:6] = Memory array Number (e.g. memory board) bit[5:0] = DIMM Slot Number if disabled for all memory use 0xFF	0xFF

Error Type	Sensor Type	Event/Reading Type Code	OEM Data1	OEM Data2	OEM Data3
				with OEM data1 bit 7:6 zero.	
PCI PERR	0x13	0x6F	0xA4 OEM code in data2/data3 & bit 3:0 unused	PCI bus number if known, otherwise 0xFF with OEM Data1 bit 7:6 zero	Bit[7:3] = PCI device number if known bit[2:0] = PCI function number is known If PCI device/function is unknown, then 0xFF with OEM Data1 bit 5:4 is zero
PCI SERR	0x13	0x6F	0xA5 OEM code in data2/data3 & bit 3:0 unused	PCI bus number if known, otherwise 0xFF with OEM Data1 bit 7:6 zero	Bit[7:3] = PCI device number if known bit[2:0] = PCI function number is known If PCI device/function is unknown, then 0xFF with OEM Data1 bit 5:4 is zero
PCIe Correctable error	0x13	0x6F	0xA0 OEM code in data2/data3 & PCIe correctable	PCI bus number if known, otherwise 0xFF with OEM Data1 bit 7:6 zero	Bit[7:3] = PCI device number if known bit[2:0] = PCI function number is known If PCI device/function is unknown, then 0xFF with OEM Data1 bit 5:4 zero
PCIe Uncorrectable or fatal	0x13	0x6F	0xA8 OEM code in data2/data3 & PCIe uncorrectable	PCI bus number if known, otherwise 0xFF with OEM Data1 bit 7:6 zero	Bit[7:3] = PCI device number if known bit[2:0] = PCI function number If PCI device/function is unknown, then 0xFF with OEM Data1 bit 5:4 zero
PCIe Correctable threshold reached	0x13	0x6F	0xA9 OEM code in data2/data3 & PCIe correctable threshold reached	PCI bus number, 0xFF and OEM Data1 bit 7:6 zero means all PCIe devices	Bit[7:3] = PCI device number if known bit[2:0] = PCI function number If 0xFF and OEM Data1 bit 5:4 zero means all PCIe devices.
PCIe Correctable error logging disabled	0x10	0x6F	0xA0 OEM code in data2/data3 & PCIe correctable error logging disabled	PCI bus number, 0xFF and OEM Data1 bit 7:6 zero means all PCIe devices	Bit[7:3] = PCI device number if known bit[2:0] = PCI function number If 0xFF and OEM Data1 bit 5:4 zero means all PCIe

Error Type	Sensor Type	Event/Reading Type Code	OEM Data1	OEM Data2	OEM Data3
					devices.
Configuration boot status event	0xF0	0x6F	0xA0 OEM code in data2/data3	Configuration Boot status: 0x01 – started 0x02 – in progress 0x03 – completed successfully 0x04 – failed Other values reserved	Reserved for OEM use. OEM can use it to send OEM specific fine granular status or progress code. If not used, it must be zero.
Update boot status event	0xF1	0x6F	0xA0 – OEM code in data2/data3	Update boot status: 0x01 – started 0x02 – in progress 0x03 – completed successfully 0x04 – failed Other values reserved	Reserved for OEM use. OEM can use it to send OEM specific fine granular status or progress code. If not used, it must be zero.
Reserved	0xF2 to 0xFF	-	-	-	-

7.2.4 RAS

In addition to error handling, the blade and blade BIOS should support memory RAS functions such as sparing, mirroring and scrubbing.

7.3 IPMI Standard

The Intelligent Platform Management Interface (IMPI) is industry-standard platform management technology in servers. The blade BIOS **shall** support IPMI revision 2.0 or above compatible server management controller and functions. The blade BIOS **shall** initialize the host to IPMI compatible server management controller communication interface supported in the blade. These interfaces include KCS (Keyboard controller style), BT (Block transfer), and SMIC as specified in the IPMI specification. At least one of these three platform interfaces **shall** be supported.

In addition, at a minimum the blade BIOS **shall** support the following IPMI server management functions:

- Serial over LAN (SOL)
- System Event Log (SEL)
- Watchdog timer during BIOS POST
- FRU inventory information

In addition, the blade BIOS **shall** expose the IPMI 2.0-compatible server management controller through ACPI interface (SPMI table) for ACPI compatible OS and application use.

7.4 POST Error and Progress

The blade BIOS **shall** support reporting BIOS POST (power-on self-test) progress and errors in the IPMI System Firmware Progress sensor. The System Firmware Progress sensor is type 0Fh with sensor offsets 00h, 01h, and 02h as specified in IPMI version 2.0 specification. This is intended to be event-only rather than a readable sensor.

The blade BIOS POST error **shall** be reported following the POST error code specified in offset 00h and 01h.

The blade BIOS **shall** report the POST progress following the ASF (Alert Standard Forum) standard progress code specified in sensor offset 02h.

The Blade BIOS should use all applicable POST error code and progress code for the blade.

The Blade BIOS **shall** implement fault resilient boot using the IPMI Watchdog timer. The blade BIOS **shall** use BIOS FRB2 timeout or BIOS POST timeout to prevent an indefinite hang during BIOS POST. The blade BIOS should also provide setup-based options to configure OS Load timeout. It is strongly recommended that this is enabled by default to prevent a hang during the operating system load.

7.5 Trusted Platform Module (TPM) Support

Platform security an important function, required for servers to prevent against any malicious attack to the servers and to authenticate the pre OS boot environment. The trusted platform module (TPM) device is a commonly used security device to achieve these security functions on platforms.

The blade should implement a TPM device on-board. When implemented, the blade TPM **shall** be implemented as defined in the following specifications:

- Trusted Computing Group (TCG) TPM PC Client Specification, version 1.2 or above

- TCG PC Client Specific TPM Interface Specification, version 1.2, revision 1.00 or above

When implemented, the blade BIOS **shall** support Static Core Root of Trust (SCRTM) and measured boot. The SCRTM and measured boot will enable the OS to implement pre-OS boot environment authentication and also any type of drive encryption for stronger data security.

When implemented for a legacy boot environment, the blade BIOS **shall** comply with the following implementation specifications:

- TCG PC Client Specific Implementation Specification for Conventional BIOS, version 1.2, revision 1.00 or above.
- TCG PC Client Specific Physical Presence Interface Specification, version 1.00 or above.

When implemented for UEFI boot environment, the blade BIOS **shall** implement the TCG interface as per following specification.

- TCG EFI Protocol Specification, version 1.2, revision 0.912 or above.

7.6 Windows Hardware Error Architecture (WHEA)

Windows Hardware Error Architecture (WHEA) is the error handling interface used by Microsoft operating systems. WHEA specifies various interfaces for the BIOS and platform hardware to expose error handling mechanisms and report extensive error information to the operating system. The WHEA interface takes advantage of additional error reporting capability available in devices such as PCI Express AER, etc. WHEA enables the OS to handle platform hardware errors effectively and complementary to standard IPMI System event logging with extensive error data. If the blade is targeted to be used with Microsoft Windows Longhorn Server, then the blade BIOS **shall** support Windows Hardware Error Architecture.

7.7 OEM Activation

OEM activation is a Microsoft Windows® anti-piracy technology and an alternate solution to Windows Product Activation. The OEM activation methods will enable volume deployment of Microsoft Windows OS for OEMs. In the blade environment, OEM activation also will allow migration of hard drives with OS image between blades.

If the blade is targeted to be used with a Microsoft Windows® environment, then the blade BIOS **shall** support Microsoft OEM activation mechanisms, also known as system locked pre-installation, SLP 1.0 and OEM Activation 2.0.

7.8 SMBIOS

DMTF SMBIOS (system management BIOS) is the industry standard interface to export platform hardware and inventory information to the OS and OS-level applications. SMBIOS is a table interface provided by the BIOS. It consists of various types of data records to provide system information.

The blade BIOS **shall** support DMTF SMBIOS version 2.3.4 or above. The blade BIOS **shall** implement all required type records to comply with SMBIOS 2.3.4 or above. The blade BIOS **shall** expose blade hardware inventory information in Type 1, 2, and 3 records from the FRU data available through the blade management controller. The blade BIOS **shall** also implement type 38 (IPMI device information) record to expose IPMI interface to operating system.

7.9 ACPI

Advance Configuration and Power Interface (ACPI) is industry standard interface for the OS to configure the platform devices and also to power manage the devices and platform itself. ACPI allows the platform to keep the legacy configuration interfaces (such as PnP BIOS, APM, MPS, etc) along with ACPI.

For details about ACPI, see the *Advanced Configuration and Power Interface Specification 2.0b* or above.

7.9.1 Power States

The blade BIOS **shall** support ACPI version 2.0b or above. At a minimum, the blade **shall** implement S0, S1 and S5 ACPI power states. The blade should support S3 power state for DP and UP blades. The blade BIOS should also support ACPI interfaces for processor power states such as p-states and C-states to allow ACPI compatible OS driver power management.

7.9.2 IPMI ACPI Power State

In blade systems, the blade management controller works closely with the blade BIOS in power and configuration managing. The Blade management controller **shall** know whenever the OS transitions blade power states. Therefore, the blade **shall** implement the *Get ACPI Power State* and *Set ACPI Power State* IPMI global commands. The blade BIOS **shall** also indicate the power state transitions through the *Set ACPI Power State* command to the blade management controller.

7.9.3 ACPI Tables

The blade BIOS **shall** support the following standard ACPI tables: RSDP, RSDT, XSDT, FADT, DSDT, MADT, SPMI (expose IPMI interface), MCFG (if applicable) and TCPA (if applicable).

The blade BIOS should also support DBGP (debug port interface) tables to aid debugging the operating system.

7.10 MPS Support

The Multi-Processor Specification specifies a table interface to provide multi processor configuration to the OS. The multiprocessor configuration information includes processor local APICs, I/O APICs, I/O bus (such as ISA, PCI) and platform interrupt assignment.

A non-ACPI compliant OS will require MPS table interface to configure the multiprocessor system. The blade BIOS **shall** support Multi Processor Specification version 1.4 or above when the blade supports two or more logical processors.

7.11 BIOS/Firmware Update

The blade **shall** support a method to update the blade system BIOS. The blade should support methods to update the firmware for other major components, such as LAN controllers and storage controllers. The blade **shall** support upgrading these components in pre-OS and optionally under the OS environment.

7.11.1 Fault Tolerant Update

Updates to the blade BIOS are critical for the blade to be operational, so the blade **shall** support fault tolerant update mechanism of these components. The fault-tolerant update mechanism can be implementation specific, but it **shall** protect blade failure due to unexpected power failure or reset during a BIOS and firmware update.

7.11.2 Recovery

The blade BIOS **shall** support a recovery update mechanism to recover the blade from failure due to BIOS image corruption. The recovery mechanism can be implementation-specific.

7.12 PXE Boot

Preboot execution environment (PXE) boot is for remote network boot functions. PXE boot is an important function, required for servers. The blade BIOS **shall** support network PXE boot from LAN channels available in the blade. The blade BIOS can use the legacy PXE option ROM in a legacy boot or EFI network stack (PXE base, DHCP driver) and LAN UNDI driver in UEFI boot.

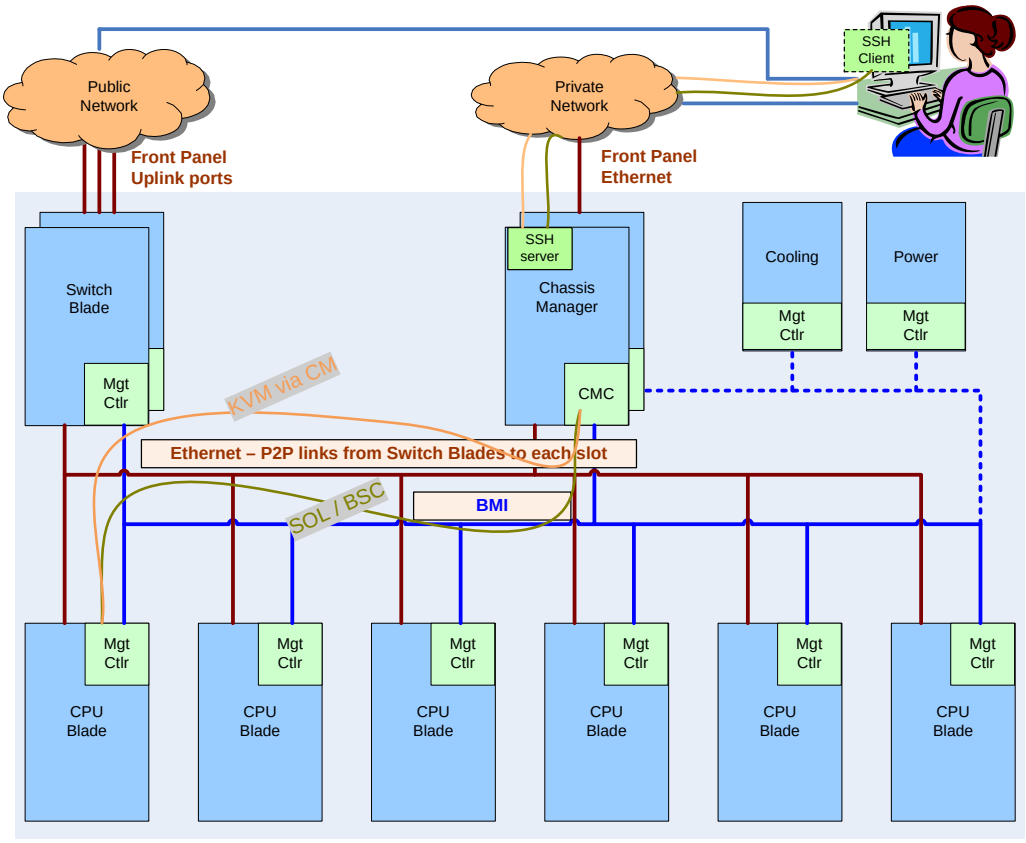
7.13 BIOS Requirements for Blade Management

The following sections provide the required BIOS and firmware functionalities for managing the blade in-band and out-of-band. The required functions in this section allow the blade to be managed in a multi-vendor environment, in which the blade and Chassis Manager are provided by different vendors.

The key ingredient for Blade management is the redirection of the blade console to a remote client. Using the redirected blade console the Chassis Manager or a remote managing software can acquire control of the blade and perform management functions such as monitoring system events, perform an SW upgrade, provision the blades for various payload.

The following figure shows the typical methods for blade console redirection.

Figure 7-1: Typical Console Redirection Methods



7.14 Compute Blade Serial Console

Compute Blade serial console (BSC) is the redirection of the local console (keyboard and video) through a serial port. This feature is essential to manage

the blade remotely via the Chassis Manager. The blade BIOS **shall** support blade serial console on the available serial ports in blade.

The serial port can be a physical external port, an internal port, or a virtualized serial port. The console redirection screen may be limited to redirecting text output from the blade during BIOS POST screen, pre-OS environment text screen (such as EFI Shell, MS-DOS), or any other text from the OS.

The serial and console redirection baud-rate, flow control, and terminal type parameters may be configurable in the BIOS. At a minimum, the blade BIOS **shall** support a baud rate of 9600 bbs, Hardware flow control with no parity, 8/1 data/stop bits and VT100 terminal type. It should support 115.2 kbs baud rate and VT-UTF8 terminal to provide optimum blade serial console performance.

7.14.1 Serial Redirection over LAN

Serial over LAN (SOL) enables redirection of the blade server console over management LAN. The Chassis Manager can utilize the SOL console for blade management and configuration. The blade BIOS extension functions, such as blade configuration boot and blade update boot, require the blade server console. The blade server may support the SOL mechanism and authentication method as specified in section 6.10.6. The blade BIOS **shall** support SOL when enabled and also should configure the serial port to match SOL configuration. In addition, the blade BIOS should enable console redirection on that serial port without requiring additional configuration by the user. Note that a small performance impact at the pre-OS screen may occur in some implementations, and are acceptable. Virtual serial port implementation may not require serial port configuration, but may require additional platform hardware initialization that the blade BIOS should perform.

7.14.2 KVM Redirection

The blade BIOS **shall** support a keyboard/video/mouse (KVM) redirection mechanism as specified in Section 6.10.5 and work transparently when a remote KVM session is established. The Blade BIOS **shall** always enable the KVM function if required and **shall not** require additional configuration.

7.14.3 Media Redirection

The blade BIOS **shall** support a media redirection mechanism as specified in Section 6.10.7 and **shall** support redirected CD/DVD devices and USB portable storage devices (such as a USB key) at a minimum. The blade BIOS **shall** enumerate redirected media transparently and **shall** include it in the boot device order configuration. Blade BIOS media redirection should support read-only at a minimum.

7.15 Compute Blade BIOS Extensions

The Chassis Manager should be able to manage compute blade to configure the blade for different payloads, alter blade boot targets, and update the blade BIOS and other updatable components. The challenge is to achieve above so that interoperability between different Chassis Managers and blades are ensured. This section specifies Blade BIOS extension interfaces that are added to standard BIOS and IPMI standards to establish interoperability. The Blade BOS extension interfaces utilize the OEM parameters 120 to 127 in IPMI System Boot options. OEMs and Vendors should assume these OEM parameters (120 to 127) are reserved for SSI blade architecture and should not use for other implementations.

7.15.1 Compute Blade OEM Parameter Access

The OEM parameters in IPMI System boot options used by Blade BIOS extensions may vary in size and may also be larger to accommodate within single IPMI Get/Set system option command depending on IPMI interface used. In order to facilitate accessing larger OEM parameter, the SSI Compute Blade boot option extension OEM parameters **shall** be accessed as multiple blocks, using Block selector in Get/Set System boot option command. The block size will vary depending on interface and on implementation, so to achieve interoperability in accessing SSI OEM parameters; a compute blade management controller **shall** implement the Parameter Block Size table as OEM parameter 120 to specify implemented block size for OEM parameters 121 to 127. The compute blade's management controller must implement the Parameter Block Size table in OEM parameter 120 as read only and should fail any attempt to write the Parameter Block Size. Table 7-3 shows the Parameter Block Size table structure in OEM parameter 120.

By using the Parameter Block Size table, the consumers of the blade parameter (such as the blade BIOS, OS, system software and Chassis manager) can split the larger OEM parameter data into multiple data blocks and use multiple Get/Set System boot option commands to access entire OEM parameter data. The Parameter Block Size table size is within the minimum required block size, so multiple Get System option commands are not required.

Table 7-3: Parameter Block Size table

	Offset	Name	Length	Description
Header	00h	Anchor String	4 BYTES	Signature string "_PB_" identifying OEM Parameter Block size table. The four ASCII character value is 5F 50 42 5F.
	04h	Checksum	1 BYTE	Byte value to obtain zero checksum.
	05h	Major Revision	1 BYTE	The major revision number in BCD format. For this version of the specification the revision is 01h.
	06h	Minor Revision	1 BYTE	The minor revision in BCD format. For this version of the specification the revision is 00h.
	07h	Length	1 WORD	Total length of the table in bytes; stored in little endian byte

	Offset	Name	Length	Description
				order. The value is 16 for this revision table.
	09h	Reserved	1 BYTE	Reserved
Data	0Ah	System Interface Block Size	1 BYTE	Block size in bytes implemented over System interface (KCS/SMIC/BT) for access to Boot Option OEM parameters 121-127. Valid values are 16 to 128 bytes, other values are reserved. An even value recommended. The BIOS and OS software that consume blade OEM parameters via System interface should be capable of supporting the maximum block size (128 bytes) to allow for interoperability.
	0Bh	Reserved	1 BYTE	Reserved, must be zero
	0Ch	IPMB Interface Block Size	1 BYTE	Block size in bytes implemented over the IPMB interface for access to Boot Option OEM parameters 121-127. Valid values are 16 to 20 bytes; other values are reserved. An even value is recommended. The consumers of OEM parameters over IPMB (such as Chassis manager) should be capable of supporting the maximum block size (20 bytes) to allow for interoperability.
	0Dh	Reserved	1 BYTE	Reserved, must be zero
	0Eh	LAN Interface Block Size	1 BYTE	Block size in bytes implemented over IPMI LAN (RMCP) interface for access to Boot Option OEM parameters 121-127. Valid values are 16 to 128 bytes; other values are reserved. An even value is recommended. The consumers of OEM parameters over IPMI LAN (such as Chassis manager, Remote management client) should be capable of supporting the maximum block size (128 bytes) to allow for interoperability.
	0Fh	Reserved	1 BYTE	Reserved, must be zero

To access the blade OEM parameters, the Set/Get system boot option command **shall** be used following IPMI 2.0 specification. In addition, the set selector and block selector **shall** be passed in as shown in table 7-4 and table 7-5. Since multiple sets are not used, the set selector 0 **shall** always be passed. In addition, each Set/Get system boot option command **shall** always read or write a complete block. If the data is only the partial length of the block size (such as the last block of a blade OEM parameter), the consumer of the blade OEM parameters must pad the remaining bytes with zeros on write and must expect to read padded bytes beyond the actual data.

Table 7-4: Set System Boot Option for Blade OEM Parameter

	Byte	Data field
Request data	1	Parameter valid Bit 7 = Parameter valid 1 – mark parameter invalid/locked [325]

		0 – mark parameter valid/unlocked [325] Bits 6:0 = Parameter number, 121 to 127 for Blade extensions
	Parameter data byte 1	Set selector, selects a particular set of blocks under the given parameter. Multiple sets are not implemented, so this value is always 0.
	Parameter data byte 2	Block selector, selects a particular block within the given set of blocks. Valid values are 1 to 255.
	Parameter data byte 3 to (3 + BlockSize - 1)	OEM parameter data to be stored in a particular block. Note: The parameter valid bit alone can be changed without sending Set selector, Block selector and OEM parameter data in parameter data and without affecting current parameter data.
Response data	1	Completion Code. Generic IPMI plus the following command-specific completion codes: 80h = parameter not supported. 81h = attempt to set the 'set in progress' value (in parameter #0) when not in the 'set complete' state. (This completion code provides a way to recognize that another party has already 'claimed' the parameters.) 82h = attempt to write read-only parameter

Table 7-5: Get System Boot Option for Blade OEM Parameter

	Byte	Data field
Request data	1	Parameter selector Bit 7 - reserved Bits 6:0 = Parameter number, 121 to 127 for Blade extensions
	2	Set selector, selects a particular set of blocks under the given parameter. Multiple sets are not implemented, so this value is always 0.
	3	Block selector, selects a particular block within the given set of blocks. Valid values are 1 to 255.
Response data	1	Completion Code. Generic IPMI plus the following command-specific completion codes: 80h = parameter not supported.
	2	Bits 7:4 = reserved Bits 3:0 = parameter version. 1h for this specification unless otherwise specified.
	3	Bit 7 = Parameter valid 1 – mark parameter invalid/locked [325] 0 – mark parameter valid/unlocked [325] Bits 6:0 = Parameter number
	4 to (4 + BlockSize - 1)	OEM parameter data residing in the particular block.

7.15.2 IPMI Directed Boot

One function of the IPMI system boot option is to select an override boot device. The selected override boot device is alternate to the BIOS-selected boot device. The IPMI system boot options allow the Chassis Manager or management software to direct the blade boot to a specific device.

The blade BIOS **shall** support the IPMI system boot option and boot to override BIOS device that is enabled in the boot selector of the boot flag parameter. The blade environment requires more boot targets in addition to IPMI standard boot devices. Thus an extended Blade Bootflag is specified in Table 6-4, which is complimentary boot device selection to standard IPMI Boot flags. This blade Bootflag is available in OEM parameter 126. The blade BIOS **shall** support the extended Bootflags and boot to the selected device when enabled.

7.15.3 Compute Blade Configuration Boot

The blade BIOS **shall** boot to a configuration environment when the Chassis Manager overrides the BIOS boot by setting the extended Blade Bootflag to 1 (Table 6-4). The configuration environment **shall** be a Pre-OS environment such as MS-DOS or EFIshell or PXE-boot. In configuration boot mode, BIOS **shall** enable LAN and load appropriate option ROMs and drivers to enable LAN connection to Chassis Manager.

Once the BIOS boots to the pre-OS environment, the BIOS **shall** use TFTP or SFTP or SCP to download configuration scripts and associated tools from the Chassis Manager. The configuration script and associated tools can be implementation-specific and depend on the configuration boot type. In order to indicate progress, the BIOS and configuration tools **shall** use the configuration boot status or update boot status events specified in Table 7-2.

To achieve this configuration boot, the blade requires chassis manager's IP address, port number, and FTP path to download configuration scripts. The Chassis Manager's IP and port information are stored in the Slot Configuration Table (SCT). The SCT **shall** be stored in the IPMI System Boot option OEM parameter 127 as specified in Table 7-7. Both the blade and CM utilizing the SCT in OEM parameter 127 will ensure inter operability in the configuration boot environment.

The BIOS **shall** use the IP address and FTP parameters specified in the SCT to download scripts to execute configuration boot. The chassis manager should be able to set the SCT parameters of the blade according to required configuration boot. For example, the Chassis Manager should be able to provision the blade for specific payloads by booting a configuration mode before updating the blade BIOS, blade management controller, or other updated components and configurations.

Table 7-6: Compute Blade System Boot Options – Slot Configuration Table

Table Type	Parameter #	Parameter Data
SCT	127	Slot Configuration Table as specified in table 7.3.

Table 7-7: Slot Configuration Table (SCT) Structure

Offset	Name	Length	Description
00h	Anchor String	4 BYTES	Signature string "_SC_" identifying Slot Configuration Table (SCT). The four ASCII character value is 5F 53 43 5F.
04h	SCT Checksum	1 BYTE	Byte value to obtain zero checksum.
05h	SCT Major Revision	1 BYTE	The revision in BCD number. For this version of the specification, the revision is 00h.
06h	SCT Minor Revision	1 BYTE	The minor revision in BCD number, For this version of specification revision is 95h.
07h	Length	1 WORD	Length of the SCT table in bytes, which is 128 for this revision of the SCT. Stored in little endian byte order.
09h	Reserved	1 BYTE	Reserved for OEM use
0Ah	NIC IP	4 BYTES	IP address, that the blade's NIC should use to establish connection to the Chassis Manager.
0Eh	NIC Netmask	4 BYTES	Netmask that the blade's NIC should use to establish connection to Chassis Manager.
12h	NIC Gateway	4 BYTES	Gateway (if any) that the blade's NIC should use to establish connection to the Chassis Manager.
16h	Mgmt VLAN	1 WORD	VLAN ID to use when communicating to Chassis Manager's TFTP server. Stored in little endian byte order.
18h	Mgmt IP	4 BYTES	IP address of the Chassis Manager to use when contacting Chassis Manager's TFTP server.
1Ch	Configuration Script Name	48 BYTES	Name of slot startup script to be downloaded and executed. Zero padded.
4Ch	Chassis GUID	16 BYTES	A unique identifier for the chassis in which the blade is installed.
5Ch	Configuration boot type	1 WORD	Bitmap of types of configuration boot valid. Bit 0 – Startup configuration Bit 1 – Update boot Bits 2-15 – reserved for future use Stored in little endian byte order.
5Eh	TFTP port	WORD	Port number used by TFTP server in Chassis Manager. 0 if Chassis Manager does not support TFTP. Stored in little endian byte order.
60h	SFTP port	WORD	Port number used by secure FTP server in Chassis Manager. 0 if Chassis Manager does not support secure FTP. Stored in little endian byte order.
62h	SCP port	WORD	Port number used by secure CP server in Chassis Manager. 0 if Chassis Manager does not support secure CP. Stored in little endian byte order.
64h	Reserved	28 BYTES	Reserved field.

7.15.4 Compute Blade Update Boot

The Chassis Manager **shall** be able to use configuration boot and scripts to update the blade BIOS and other updatable components firmware. The blade BIOS **shall** allow such updating mechanism in the blade configuration boot environment. When an update boot is required, the Chassis Manager will update the fields in the SCT and modify the configuration script name to the boot script name and restart or power-on the blade.

7.15.5 BIOS Parameter Configuration

The blade BIOS **shall** support a scriptable mechanism (e.g. command-line EFI tool) to modify the following BIOS configuration parameters:

- Load BIOS default setting on next boot
- Enable/disable POST error pause
- Set BIOS Admin password
- Set BIOS User password
- Get/Set BIOS Boot order
- Get/Set Serial port settings
- Enable/disable console redirection
- Save entire BIOS Configuration to files for deployment
- Load entire BIOS configuration from files for deployment
- Reset the blade

Updates to the BIOS configuration **shall** apply to the blade beginning with the next boot.

7.15.6 Boot Order

The blade BIOS **shall** allow configurable boot device order. The boot device selection should contain the following devices when present:

- CD/DVD (redirected or local)
- Local hard drive
- External storage
- USB storage (redirected or local)
- PXE boot
- Floppy (if applicable)
- Forced to configuration boot

The blade BIOS **shall** support the boot order configuration through BIOS setup and also through a command line based tool for scripting, as specified in Section 7.15.5, BIOS Parameter Configuration.

7.15.7 Boot Order Table

The BIOS **shall** support the boot order table specified in this section. The boot order table is an extension in the IPMI system boot option's OEM parameter.

Table 7-8 and Table 7-9 show the details of the boot order table. The boot order table specifies a common interoperable mechanism to modify the blade boot device order by the Chassis Manager and to enable the blade to inform the Chassis Manager about the existing boot order information.

Table 7-8: Compute Blade System Boot Options – Boot Order Table

Table Type	Parameter#	Parameter Data
BOT	125	Boot order configuration table as specified in Table 7-9.

Table 7-9: Boot Order Table (BOT) Structure

	Offset	Name	Length	Description
Header	00h	Anchor String	4 BYTES	Signature string "_BO_" identifying Boot Order Table (BOT). The four ASCII character value is 5F 42 4F 5F.
	04h	BOT Checksum	1 BYTE	Byte value to obtain zero checksum.
	05h	BOT Major Revision	1 BYTE	The revision in BCD number. For this version of the specification, the revision is 00h.
	06h	BOT Minor Revision	1 BYTE	The revision in BCD number. For this version of the specification the revision is 95h.
	07h	Length	1 WORD	Length of the Boot Order table structure. Stored in little endian byte order.
	09h	Reserved	1 BYTE	Reserved
Data	0Ah	Update flag	1 BYTE	Flag to indicate Boot Order has been updated. Bit 0 = 1 if BIOS has updated Boot order Bit 1 = 1 if Chassis manager has updated Boot order and requested new Boot order. Other bits are reserved. All other times BIOS retains existing boot order.
	0Bh	Boot Order information	Varies	Boot order information has two parts of information, Boot order data followed by Boot device name and path data. Boot order data specifies system boot order and also device order within a particular device type. Each boot order data starts with an Order type, Order length and is followed by an ordered list of device numbers. Boot order data ends with End boot order type 0FFh and is followed by 00h order length. Device name/data contains description names of devices and device specific hardware path, which will uniquely identify the boot devices in the compute server.
Boot Order Data	0Bh	Order type	1 BYTE	This field specifies the type of order. 00h = System Boot order, this type will specify order in which each boot device type should attempt to boot. This is a mandatory order when BOT is implemented.

				The order of devices within each device type is specified by the following Order types. The order type definition includes legacy and EFI boot devices. Legacy boot device orders within a particular device class are optional. When absent, the BIOS **shall** follow the default enumeration for that device class. Compute servers that support UEFI/EFI, must implement the EFI boot order. 01h = Floppy disk drive (FDD) order 03h = CD/DVD drive order 05h = USB removable media order 06h = Network device order 08h = Local Hard disk drive (HDD) order 09h = External HDD order 80h = BEV device order 10h = EFI boot order that specifies the order of EFI boot targets. 0C0h to 0DFh = OEM device types that can be used for OEM specific devices. 0FFh = End of boot order type, which marks the end of boot order lists and must be followed by 00h to indicate zero length order. Other boot order types are reserved.
	Varies	Order length	1 BYTE	Number of boot device in a particular order list.
	Varies	Device order list	Order length	This field contains the ordered list of boot device types or boot devices. For System Boot order type, the Device list **shall** contain the ordered list of device types. Each device type is a byte value of one of the following device types: 01h = FDD; 08h = Local HDD; 03h = CD/DVD; 05h = USB removable media; 06h = Network Device (PXE); 09h = external HDD; 80h = BEV; 10h = EFI Boot device. For legacy or OEM device order type, the Device list **shall** contain an ordered list of device numbers within a particular device type. Each device number is a byte value and may have a name associated in the Device name/path field. For an EFI Boot order type, the Device list **shall** contain an ordered list of EFI boot targets. Each boot target is a two byte number as in the EFI BootOrder variable. The boot target number associates the EFI boot device to Boot#### EFI variable, where '####' represents the boot target number. Each boot target must have a device path in the Device name/path field.
Device name and Path data	0Bh + <Boot Order data> Size	Device Name/Path	Varies	This field contains the device name (user readable description) and the hardware path data for all devices in the boot order data. The name is optional for legacy and EFI boot devices, but mandatory for OEM devices. The device name/path data should be used when boot devices are reported to the user. Each device name/path entry starts with a 3 byte device code: first byte represents order type that the device is part of and the following 2 bytes represent device number. For legacy devices and OEM devices this field must be in the following format: Size

			<table><tr><td>Type</td><td>1 BYTE</td><td>Order type</td></tr><tr><td>Number</td><td>1 WORD</td><td>Device number. Stored in little endian byte order.</td></tr><tr><td>Name</td><td>Varies</td><td>Null terminated ASCII string</td></tr></table> For EFI devices, this field additionally contains the EFI device path to boot target, which is mandatory. The device path must comply with the UEFI 2.0 specification. This field should have the following format for EFI devices:<table><tr><td></td><td>Size</td><td></td></tr><tr><td>Type</td><td>1 BYTE</td><td>Order type = 10h</td></tr><tr><td>Number</td><td>1 WORD</td><td>Device number. Stored in little endian byte order.</td></tr><tr><td>Path length</td><td>1 WORD</td><td>Size of Device path in bytes. Stored in little endian byte order.</td></tr><tr><td>Name</td><td>Varies</td><td>Null terminated Unicode string. The string is UTF-16 encoding format as specified in Unicode 1.2 standard.</td></tr><tr><td>Device path</td><td>Path length</td><td>EFI Device path of particular device</td></tr></table> The total size of the device name/path field should be BOT size less the rest of the data.

8 High-speed I/O Signal Specifications

8.1 High Speed I/O Signaling Background

The systems interconnect for the SSI Compute Blade is a backplane SERDES interface. The basic parameters for the signals are defined in the IEEE Std. 802.3ap-2007 "Backplane Ethernet" standard. For implementation on the Compute Blade, the SERDES I/O connections are terminated with a series AC coupling capacitor (.01uF) near the receiver pins of the primary fabric controllers.

The directions of the high-speed signals are defined as "Compute Blade"-centric. As defined by the midplane interface connector, the transmit pairs on the midplane connector should connect to the transmit pins on the compute blade's primary fabric Ethernet controller. Similarly, the receive pins on the midplane connector should connect to the receive pins on the compute blade's primary Ethernet controller. It is the responsibility of the midplane designer to properly connect the transmit pins from the compute blade to the receive pins on the switch module.

Midplane connector pin skew length compensation **shall** be handled on the midplane. The Compute Blade trace design should not compensate for the midplane connector signal pin skew length mismatch.

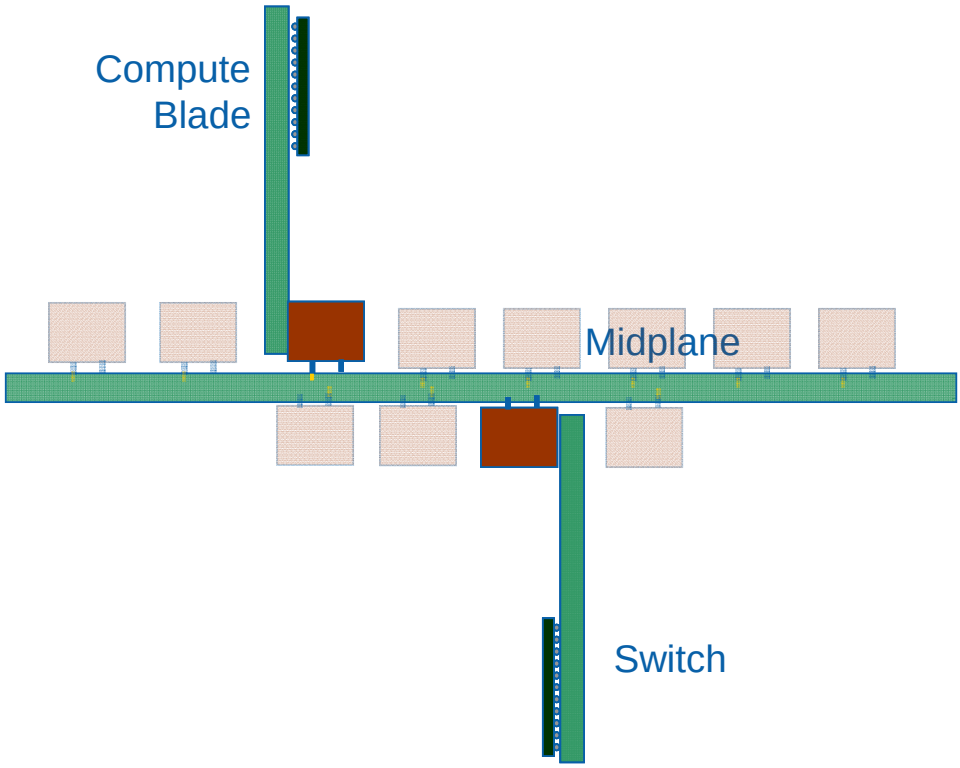
8.2 Introduction

The system interconnects, shown below, for the SERDES interconnect has three basic classifications of PBAs: a compute blade, a midplane / backplane, and a switch. Each PBA **shall** meet the specified frequency domain parameters. Hereafter, references to midplane or backplane are considered the same. This section covers the high-speed signal interconnect requirements for the Compute Blade PBA.

For specifications related to the midplane and Switch PBAs, see the specifications for those components.

Each board **shall** meet certain specified eye diagram test requirements specified in the Compliance and Interoperability Specification.

Figure 8-1: SERDES Fabric System Interconnect



8.3 Compute Blade PBA I/O Connector Pin Definitions

Aside from power, there is one high-speed I/O connector on the blade PBA that connects with the midplane. If a mezzanine is present, then there will be another high-speed I/O connector on the mezzanine card that connects to the midplane (Flexi-channel interconnects). The mezzanine connects to the blade via one required low-profile connector and an optional expansion low-profile connector.

8.4 Ethernet SERDES Device Characteristics

See IEEE Std 802.3ap-2007 "Backplane Ethernet" standard KR, KX4, or KX for transmitter and receiver specifications. The 1000BASEKX specification supports 1 Gb/s serial operation, 10GBASE-KX4 supports 10Gb/s 4-lane operation, and 10GBASE-KR supports 10Gb/s serial operation. Future XAUI devices **shall** comply with IEEE Std 802.3ap-2007 "Backplane Ethernet" standard 10BASE-KX4 specifications.

8.5 Compute Blade Electrical Design Guidelines

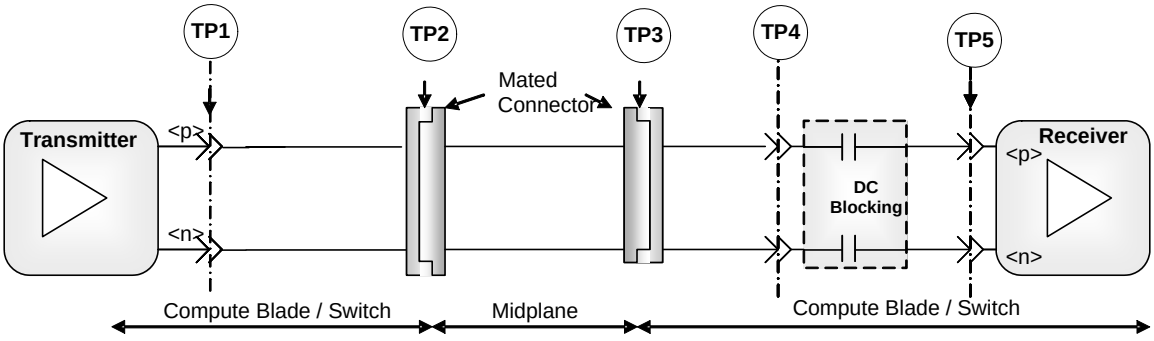
These general guidelines only pertain to 10GBASE-KR SERDES. The specifications for 10GBASE-KX4 and 1000BASEKX are subject to change.

8.5.1 Channel Definition

In general, a channel is the end-to-end interconnection between the transmitter and receiver die pad. However, in the context of this document, a channel is defined between the pins of two SERDES chips. The pins of a chip are the exterior electrical connection points. This document defines channel between the transmitter pin (TP1) and receiver pin (TP5). See Figure 8-2. IEEE Std 802.3ap-2007 "Backplane Ethernet" standard defines the channel between the transmitter pin (TP1) and the AC coupling capacitor's receiver pin (TP4).

For PCI Express, the blocking capacitor is associated with the transmitter as opposed to other common interfaces which has the association with the receiver. In any case it is undesirable to have capacitors on both ends of the channel for 5Gb/s and above.

Figure 8-2: Channel Definition



To facilitate the test of modules and midplanes, test cards are required for frequency domain testing.

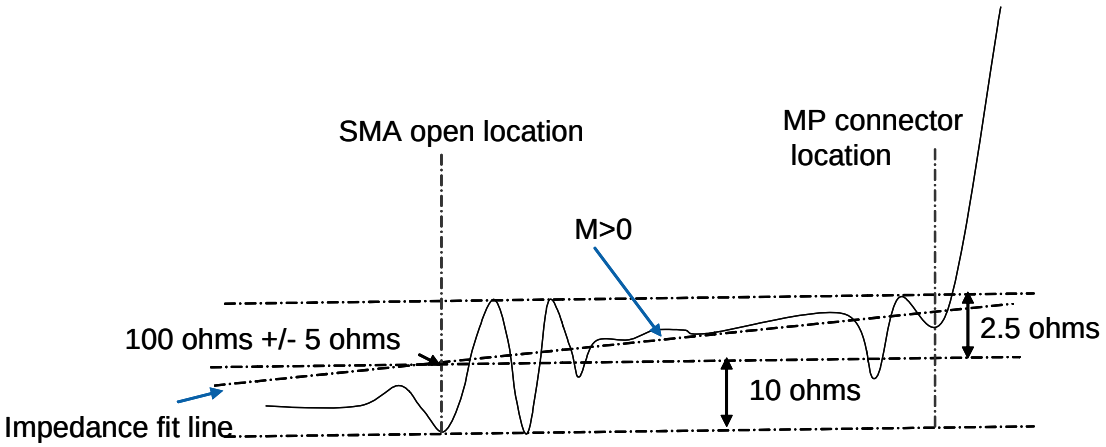
8.5.2 Test Cards

The card to test a switch is called a blades-switch test card – midplane connector version. (BTSC-MC). The TP1/TP5 side of the test connector utilizes high-frequency picoprobes.

The BTSC-MC board material **shall** be Nelco 4000-13SI or Nelco 4000-12SI or the equivalent. The electrical impact of via and launch structures **shall** be minimized as show in the TDR. Rosenberger SMCC 32K243-40M edge mount microstrip SMA connectors or the equivalent **shall** be used. The traces on the test card **shall** be 3 inches long +/- 2 mils. TDR impedance variation max to

min **shall not** exceed +2.5/-10ohms. The slope of the average fitted impedance **shall** be greater than 0. The target differential impedance of the test line is 100 ohms +/- 5% as verified with TDR. The test card TDR specification is illustrated in.

Figure 8-3: Test Card Differential TDR Requirements from the SMA Connector



8.5.3 Test Configurations

Figure 8-4: Switch Testing with VNA

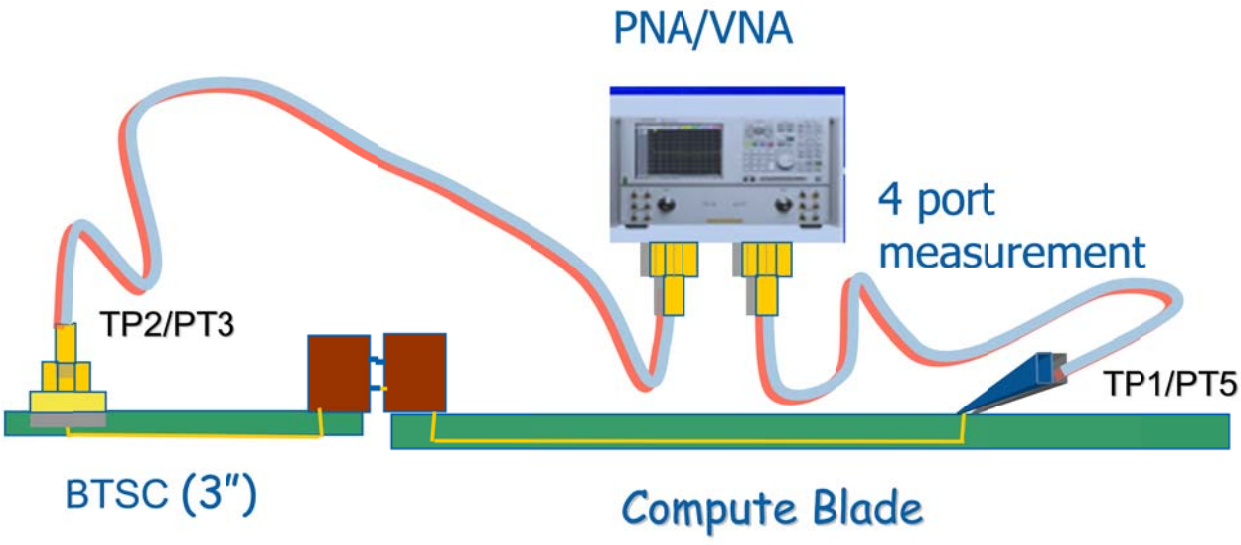


Figure 8-4 illustrates the configuration to use for performing frequency domain measurements.

8.5.4 Frequency Domain Specifications

Only the compute blade parameters are specified here. See the channel definition for the midplane and switch parameters.

8.5.4.1 Interconnect Insertion Loss

A series of parameters are defined for use in interconnect channel design. These parameters address the channel impairments, such as insertion loss and crosstalk. They are derived from differential channel s-parameters. The range for these parameters is given in Table 8-1.

Table 8-1 also lists the baseline maximum insertion loss parameters and frequency ranges for respective interconnect components and ranges for the evaluation of parameters.

The 1000BASEKX specification supports 1 Gb/s serial operation, 10GBASE-KX4 supports 10Gb/s 4-lane operation, and 10GBASE-KR supports 10Gb/s serial operation. PCI Express 2.0 supports 2.5Gb/s and 5 Gb/s serial operation.

Table 8-1: Compute Blade Maximum Attenuation and Frequency Range Parameters

Para-meter	1000BASE-KX	10GBASE-KR	10GBASE-KR	PCI Express* 2.0	Units	Notes
f_{min}	0.05				GHz	S parameter measurement low freq
f_{max}	15				GHz	S parameter measurement high freq
b_1	2.00E-05				$dB/Hz^{1/2}$	baseline maximum insertion loss parameter
b_2	1.10E-10				dB/Hz	baseline maximum insertion loss parameter
b_3	3.20E-20				dB/Hz^2	baseline maximum insertion loss parameter
b_4	-1.20E-30				dB/Hz^3	baseline maximum insertion loss parameter
f_1	0.125	.312	1	0.5	GHz	Low frequency for evaluating IL
f_2	1.25	3.125	6	5	GHz	High frequency for

Para-meter	1000BASE-KX	10GBASE-KR	10GBASE-KR	PCI Express* 2.0	Units	Notes
						evaluating IL
f_a	0.1	0.1	0.1	0.1	GHz	Low frequency for evaluating ICR
f_b	1.25	3.125	5.15626	5	GHz	High frequency for evaluating ICR

Additional parameters designated in Table 8-2 for a compute blade are used to adjust the evaluation limits for attenuation, insertion lost deviation (ILD), and insertion loss to crosstalk ratio (ICR). These parameters preserve the form of specification for blade, backplane, and switch.

Table 8-2: Compute Blade Interconnect Channel Adjustment Parameters

Parameter	Compute Blade	Notes
AK_{ic}	0.25	Max attenuation adjustment
DK_{ic}	0.25*2/dB(A(2 GHz))	Deviation adjustment
$ICR_{min}B$	20.5	ICR log-log slope limit
$ICR_{min}K$	32	ICR log-log intercept limit

Note: The midplane and switch parameters are supplied in their respective specifications.

8.5.4.2 Fitted Attenuation

A matched uniform transmission line exhibits a smooth attenuation with frequency. The fitted attenuation, A(f), is represents the equivalent average matched transmission line for a given channel. It is determined with a least mean squares fit of to the insertion loss over the frequency range f1 to f1 and stated in Appendix B.

The transmission line insertion loss in dB, IL(f) is measured at N uniformly-spaced frequencies fn spanning the frequency range f1 to f2. It is defined in Equation 8-1.

Equation 8-1

$$IL(f) = -20 * \log\left(\left|sdd21(f)\right|\right)$$

The maximum attenuation (A_{max}) in dB is based on trace and stackup material properties for one meter of advanced FR2 with two connectors. This is defined

in Equation 8-2. The parameters AK_{ic} budgets A_{max} for each PBA, AK_{ic} for the compute blade. These are summarized in Table 8-2.

Equation 8-2

$$A_{\max}(f) = 20 \cdot \log 10(e)(b1 \cdot \sqrt{f} + b2 \cdot f + b3 \cdot f^2 + b4 \cdot f^3) \cdot AK_{ic}$$

8.5.4.3 Insertion Loss

Insertion loss, IL(f), is defined as the magnitude, expressed in decibels, of the differential thru channel response measured at respective locations for the compute blade, switch, or midplane according to Table 8-3.

Table 8-3: Insertion Loss Measurement Locations

Interconnect Component	Measurement Location (port 1)	Measurement Location (port 2)	Notes
Compute Blade	TP1/TP3	TT2/TP5	See Figure 8-2
Midplane	TP2	TP3	See Figure 8-2
Switch	TP1/TP3	TT2/TP5	See Figure 8-2

The insertion loss, IL(f), **shall** be within the region above the lines defined in Equation 8-3 and Equation 8-4 and is called the maximum insertion loss (IL_{max}). The insertion loss fit A(f) **shall** be in the region above $A_{max}(f)$ in the frequency range between f1 and f2. $A_{max}(f)$ and IL_{max} limits are adjusted per interconnect. These are illustrated by Figure 8-5.

Figure 8-5 is an illustrative example of the loss of an entire system. This diagram is for reference purposes only.

Equation 8-3

$$IL(f) \le IL_{\max}(f) = A_{\max}(f) + 0.8 + 2.0 \times 10^{-10} f$$

for $f_{min} \le f \le f_2$

Equation 8-4

$$IL(f) \le IL_{\max}(f) = A_{\max}(f) + 0.8 + 2.0 \times 10^{-10} f + 1 \times 10^{-8}(f_1 - f_2)$$

for $f_2 \le f \le f_{max}$

Figure 8-5: Compute Blade Insertion Loss and Attenuation Limit Example

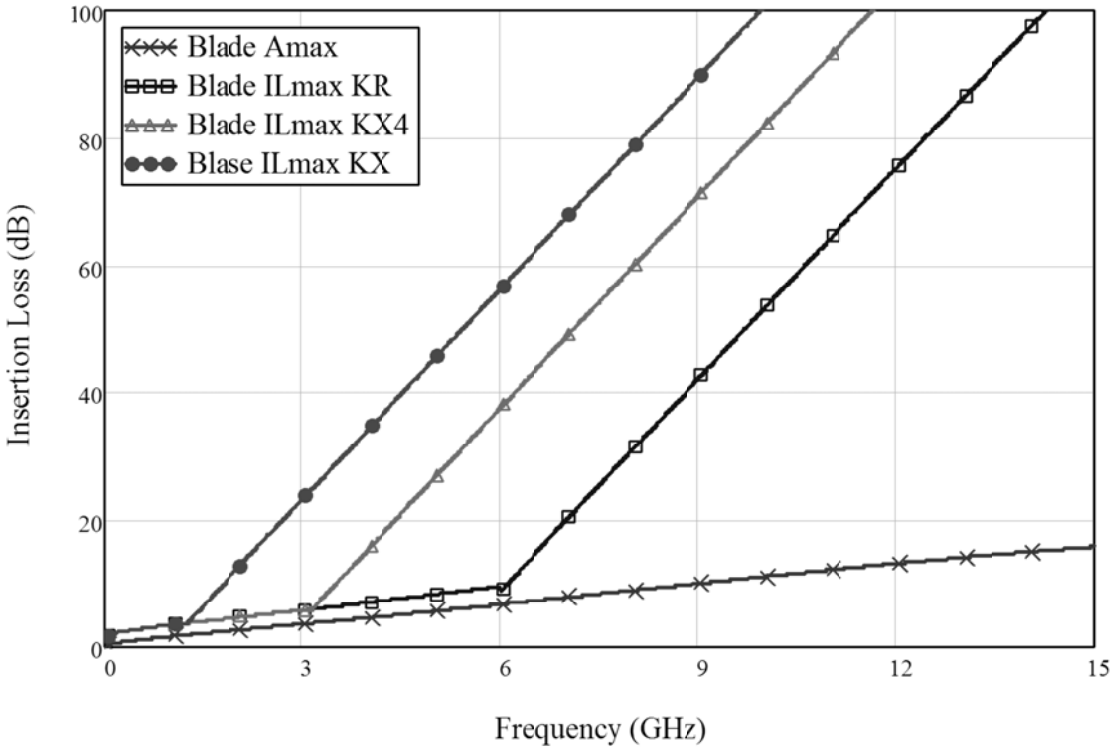
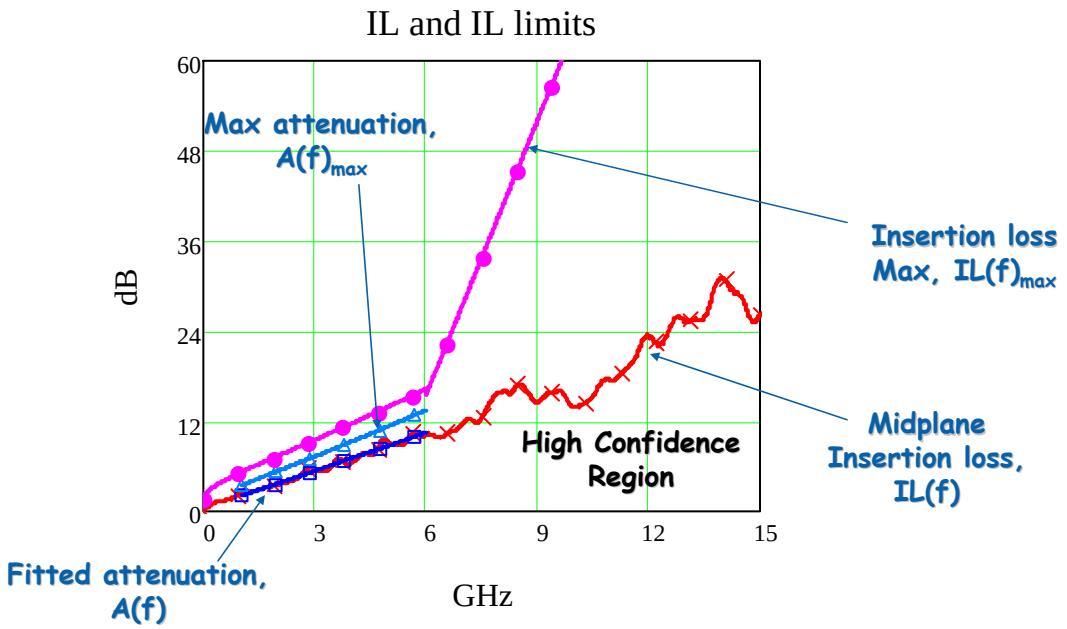


Figure 8-6: KR insertion Loss Example



8.5.4.4 Insertion Loss Deviation

The insertion loss deviation, as defined by Equation 8-5 is the difference between the insertion loss, IL(f), and the least mean squares fit, A(F), defined in Appendix B.

Equation 8-5

$$ILD(f) = IL(f) - A(f)$$

ILD **shall** be within region defined in Equation 8-6 and Equation 8-7.

Equation 8-6

$$ILD(f) \geq ILD_{\min}(f) = (-1.0 - 0.5 \times 10^{-9} f) \times DK_{ic}$$

Equation 8-7

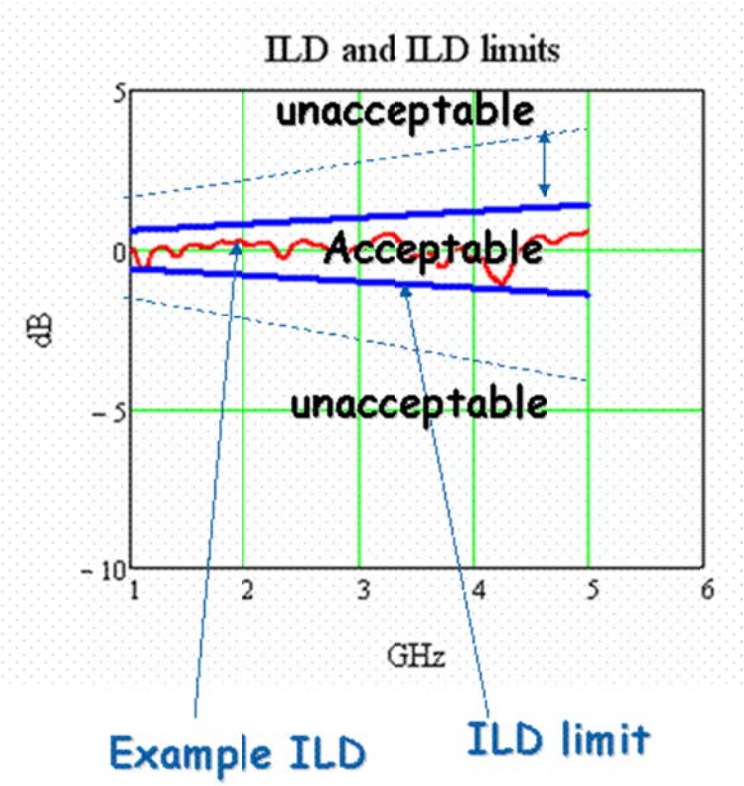
$$ILD(f) \leq ILD_{\max}(f) = (1.0 + 0.5 \times 10^{-9} f) \times DK_{ic}$$

For f1<f<f2

The values of f1 and f2 are dependent on interconnect technology and the data rate and are given in Table 8-1. (PCI Express 2.0 uses the same PBA electrical parameters as KR).

ILD interconnect adjustment factor, DK_{ic}, for respective interconnect component is specified in Table 8-1. and is illustrated in the following figure.

Figure 8-7: Insertion Loss Deviation Limits Example



8.5.4.5 Insertion Loss to Crosstalk Ratio (ICR)

See Appendix B for a description of NEXT, FEXT, PSXT, ICR, x_{avg}, ICR_{avg}, M_{icr}, B_{icr}. These are used to define the insertion loss to crosstalk fit, $ICR_{fit}(f)$. The ICR fit **shall** be greater than or equal to the $ICR_{min}(f)$ with in the frequency range between f_a and f_b.

Equation 8-8

$$ICR_{fit}(f) \geq ICR_{\min}(f) = ICR_{\min}B - ICR_{\min}K \times \log(\frac{f}{5GHz})$$

The limits that control values for $ICR_{min}B$ and $ICR_{min}K$, are specified in Table 8-2.

8.5.4.6 Return Loss Parameters

The interconnect return loss, RL(f), measured in dB at locations specified in Table 8-4, is recommended to be greater than or equal to RL_{min} as defined in Equation 8-9 through Equation 8-11.

Table 8-4: Return Loss Measurement Locations

Interconnect Component	Measurement Location (port 1)	Notes
Compute Blade	TP2/TP3	See Figure 8-2. Made without attached device
Midplane	TP2/TP3	See Figure 8-2.
Switch	TP2/TP3	See Figure 8-2. Made without attached device

Equation 8-9

$$RL(f) \geq RL1$$

for FRL1 < FRL2 and

Equation 8-10

$$RL(f) \geq RL1 - \frac{(RL2 - RL1)}{\log(\frac{FRL2}{FR1})} \log(\frac{f}{FRL2})$$

for FRL2 < f < FRL3

RL1, RL2, FRL1, FRL2, and FRL3 are specified for each interconnect component in Table 8-5.

Equation 8-11

$$RL(f) \geq RL_{\min} = RL2$$

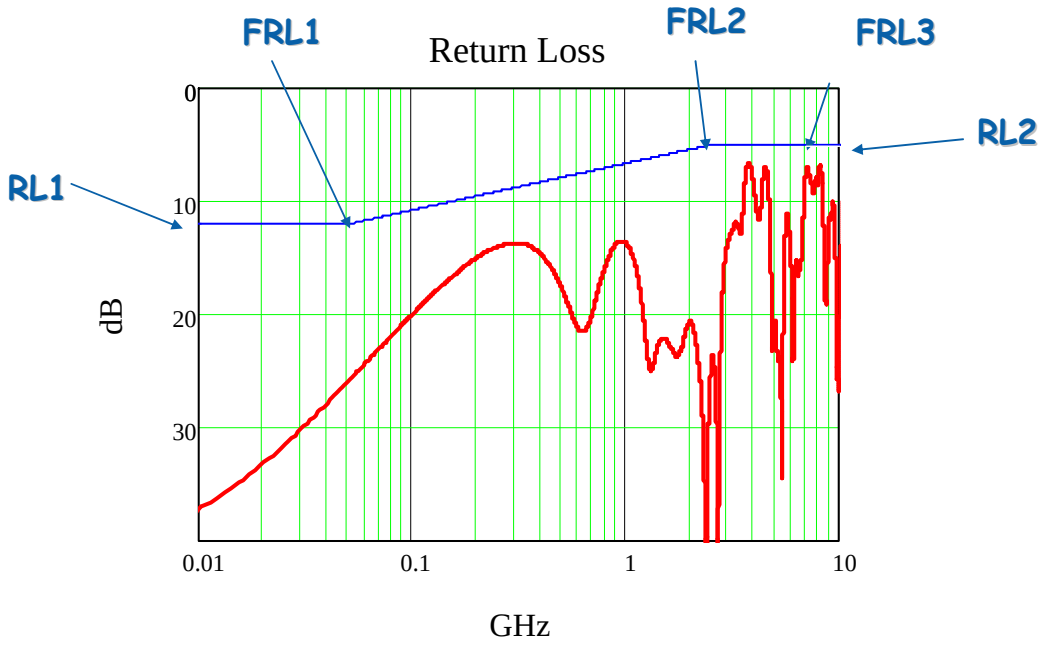
for f > FRL3

Table 8-5: Return Loss Parameters

	Compute Blade	Units
FRL1	50	MHz
FRL2	2500	MHz
FRL3	7500	MHz
RL1	12	dB
RL2	5	dB

Note: The midplane and switch parameters are supplied in their respective specifications

Figure 8-8: Return Loss Example



8.6 Compute Blade PCB Design Guidelines

These PCB guidelines are focused toward to 10GBASE-KR SERDES. PCI Express 2.0 uses the same routing guidelines as 10GBASE-KR. The specification for 10GBASE-KX4 and 1000BASEKX are a superset of 10GBASE-KR SERDES.

8.6.1 PCB Trace Design

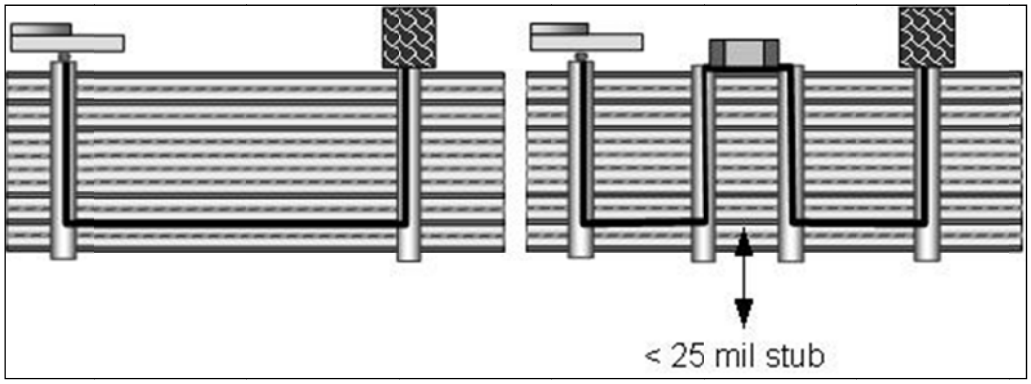
Table 8-6: Compute Blade PBA Trace Design Guidelines

Category	Guideline	Requirement	Notes
Layer assignment	Bottom stripline layer	Required for 10GBASE-KR and PCI Express 2.0	One layer as indicated in **Error! Reference source not found.**Figure 8-9.l
Trace design	Edge coupled differential stripline	Required	Broadside coupled is not recommended
Trace width target	> 6 mil	Required for 10GBASE-KR	

Space between differential pairs (coupling)	4 x the dielectric height for stripline. 7 x the dielectric height for microstrip.	Required	Exception is for quad routing to maintain < 25 mil via stubs.
Corners	Mitered or rounded corners are recommended.	Recommended	This is an EMI reduction suggestion
Min Max Length	2 to 5 inches KR 2 to 10 inch for KX and KX4	Required	
Length Matching	2 mils match between lines in a pair	Required for 10GBASE-KR	5 mils for KX and KX4
Impedance	85 ohm target +/- 15%	Required	Recommend +/-10%
Construction	Ground reference symmetric strip line.	Required	
Copper	Reverse treat copper desirable.	Recommended	
Via anti pad to via hole	> 3 times hole diameter is desirable	Recommended	Tune for minimum return loss
Via stub	< 25 mils	Required for 10GBASE-KR and PCI Express 2.0	
Material	FR4000-6 or better Loss tangent of less than 0.018 for blade environment	Required	FR408 or better recommended for for KR
Board thickness	0.093 mils	Required	

The dark lines in Figure 8-9 represent potential trace routes.

Figure 8-9: TX and RX Compute Blade PBA Layer Routing

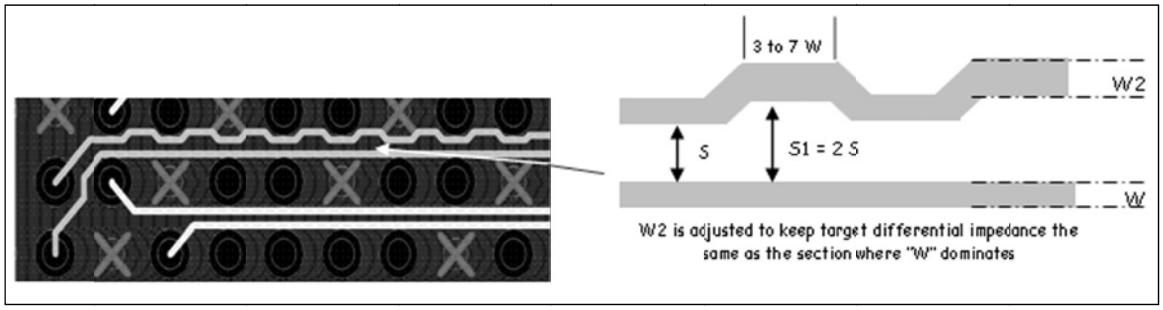


8.6.2 Length Matching

Serpentine routing, as indicated in Figure 8-10, **shall** be utilized to match intra pair length mismatches.

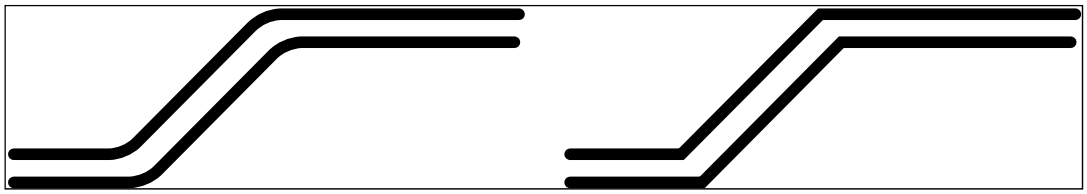
Note: All delay mismatches in connector pairs **shall** be compensated via midplane routing adjustments not on the compute blade.

Figure 8-10: Serpentine Trace Length Compensation



Back-to-back turns, as shown in Figure 8-11 should be used to compensate for corner length mismatches.

Figure 8-11: Back-to-Back Corner Compensation



8.6.3 DC Blocking Capacitor Layouts

The capacitor size **shall** be 0402 or smaller. 0603 and 0805 capacitors are not recommended. The capacitor dielectric **shall** be X7R or better.

The DC blocking structure is only associated with the blade receiver for Ethernet. See the receiver chip specification for the recommended capacitor value and the requirement for the DC blocking capacitor. Some devices may not require a DC blocking capacitor on the PWB. Do not use DC blocking capacitors on lines associated with the blade transmitter

8.6.4 Test Points

No additional vias **shall** be added for test points.

8.6.5 Void, Splits, and Proximal Metal

No trace **shall** run parallel to a split. Either side of a differential trace **shall** be greater than four times the dielectric height linear distance away from any metal or void in adjacent reference place for stripline and six times for microstrip. The exception is the connector and BGA break out.

- 10GBASE-KR and PCI Express 2.0 traces **shall not** cross splits or voids in adjacent planes.
- 10GBASE-KX4 and 1000BASE-KX traces should NOT cross voids in adjacent planes.
  - if traces must cross a plane split, then 0402 size coupling cap(s) **shall** be used adjacent to the traces where they cross the split, and the capacitor **shall** connect the two planes. The caps **shall** have vias connecting directly to the planes and within 100 mils of the crossing.

8.6.6 Connectors

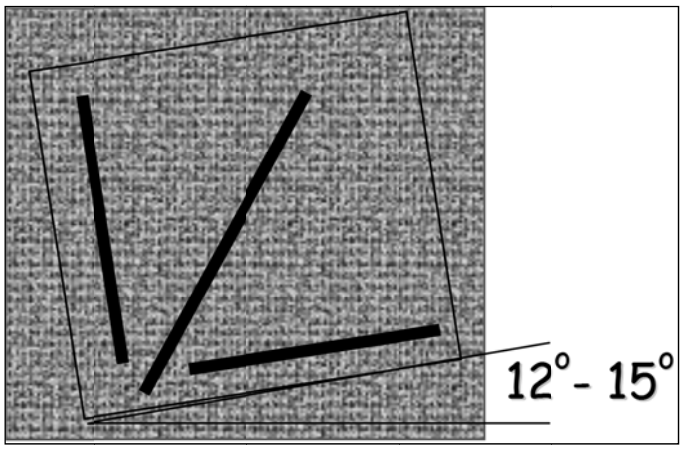
The midplane connector is specified in the mechanical section. Interposer connectors are not recommended for 10GBASE-KR.

NOTE: All delay mismatches in connector pairs **shall** be compensated via Midplane routing adjustments. The midplane designer **shall** consult the connector vendor datasheets for the correct length matching needed to correct the delay mismatches between the mated connector pair between the compute blade and midplane.

8.6.7 Dielectric Weave Compensation

Traces **shall not** run parallel to the dielectric FR4 weave. The recommended compensation method is to rotate the artwork 12 to 15 degrees and shown in Figure 8-12.

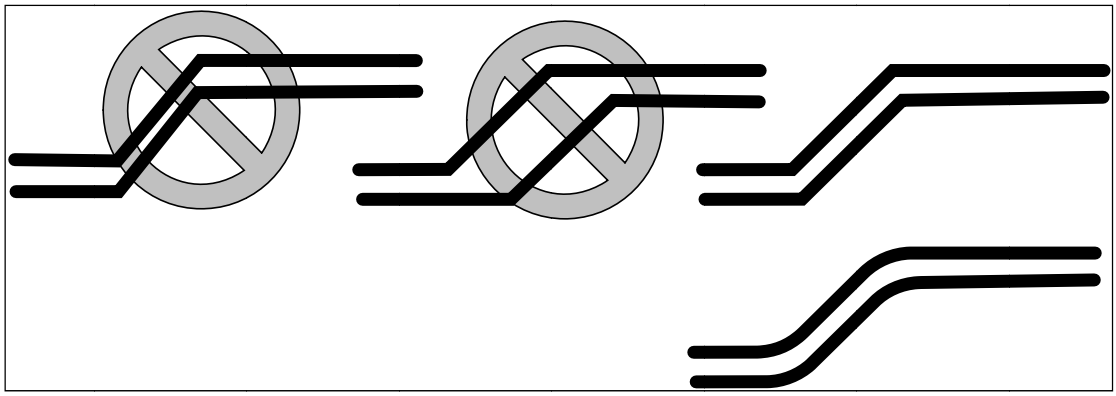
Figure 8-12: Dielectric Weave Compensation



8.6.8 Trace Spacing

Constant differential trace spacing (except in breakouts) **shall** be maintained as show in Figure 8-13.

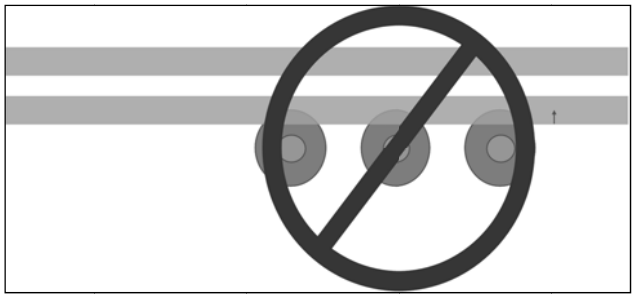
Figure 8-13: Trace Spacing Guidelines



8.6.9 Vias

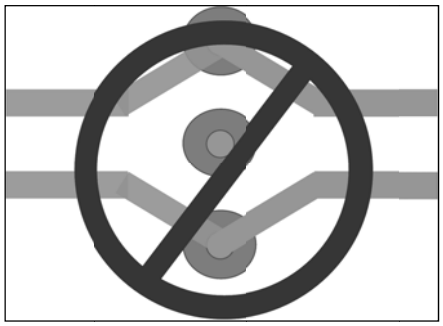
Traces **shall not** have anti pad overlap as shown in Figure 8-14.

Figure 8-14: Anti-via Overlap



There **shall** be no signal vias placed between a differential via pair as shown in Figure 8-15.

Figure 8-15: No Signal Vias Between Differential Vias



Connectors and BGA areas **shall** implement a differential via design and should be implemented for others areas as shown in Figure 8-16. In general, hole within a pair placement should be as close as manufacturing constraints permit. Hole size should be as small as manufacture requirements permit. In an area of the PBA that is unconstrained by other nearby traces, the anti via should be 3 times the diameter of the hole. In other area, the anti pad should be as large as can be accommodated by manufacturing and nearby routing constraints. Conversely the pad should be as small as possible.

Figure 8-16: Differential Via Example 1

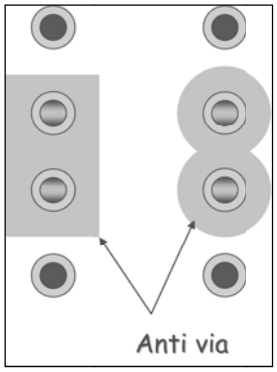
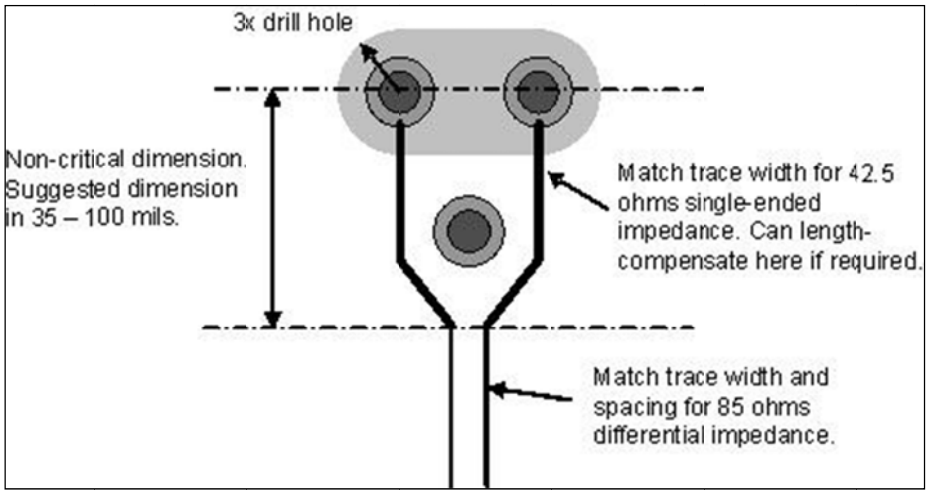


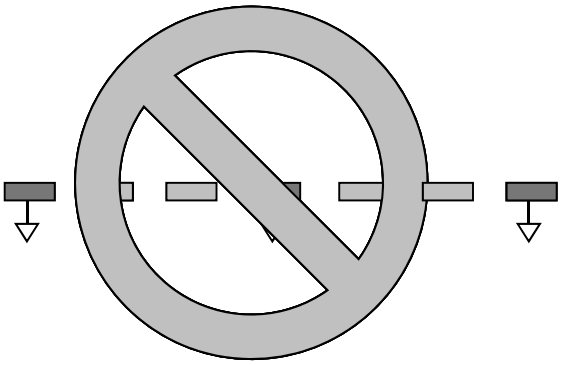
Figure 8-17: Differential Via Example 2



8.6.10 Guard Traces

Guard traces should not be implemented.

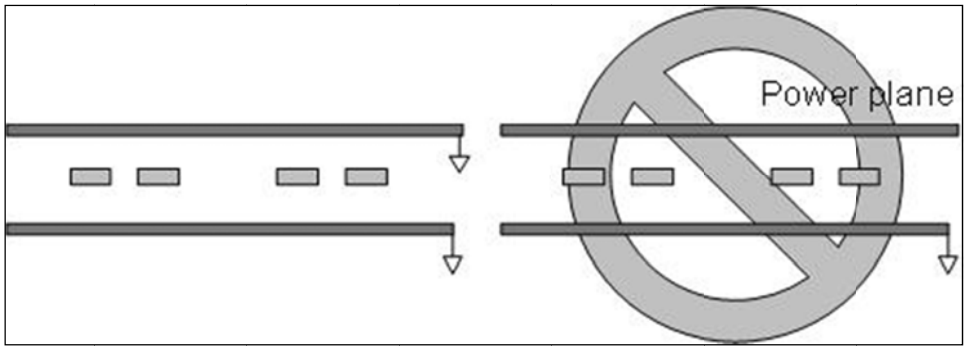
Figure 8-18: No Guard Traces



8.6.11 Signal Referencing

All SERDES traces **shall** be referenced to ground and **shall not** be referenced to power. If striplines are employed, symmetric striplines **shall** be used. Do not route within the profile of a VRM.

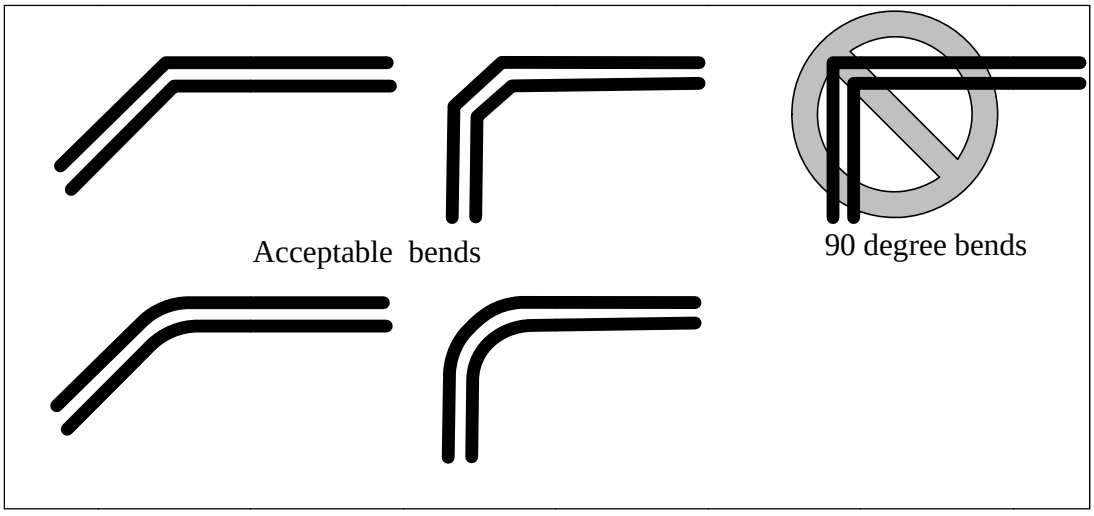
Figure 8-19: Signal Referencing



8.6.12 Corners

Figure 8-20 should be used for routing trace corners.

Figure 8-20: Trace Corner Routing



8.7 Management Signal Specification

8.7.1 I2C Bus Implementation

The management controller **shall** be able to drive up to a maximum capacitive load of 650 pF on the I2C data and clock signals and meet the timing requirements of the I2C specification. The blade must support a minimum I2C speed of 100 kHz.

Note: One possible interconnect topology supported by this load limit: 25” of the midplane trace, 5” of trace on each blade, two vias and one plated through hole per controller device and a total of 20 intelligent devices.

- System pullup resistors **shall** be located on the system midplane board for all of the I2C busses in the system. It is recommended that the minimum value of pullup supported by the I2C specification be used to keep the rise-time of the signal as fast as possible. A 4.7k Ohm resistor is the maximum value recommended for 100 kHz operation with a maximum bus capacitance of 400 pF.
- It is recommended to carefully evaluate your system design and use a value of pullup that is compliant with the devices used on the bus, and one that affords the desired response time to meet 100 kHz bus operation.
- See the I2C bus specification Version 2.1 or later for definition of requirements for 100 kHz operation.

The following are I2C recommendations for I2C busses residing completely on the compute blade:

- Output drivers should be scalable in output drive current in order to better control undershoot, overshoot, and other system noise problems.
- It is recommended that I2C buffers and repeaters should not be used. These devices frequently cause spurious operation of the I2C bus, causing contention and missed or corrupted data.
- For I2C bus loading, it is recommended that I2C busses are divided in a manner that keeps the distributed capacitance of each bus at a minimum.
- The number of device inputs per bus should be controlled to ensure that bus capacitance loading is maintained within the I2C bus specification, which is about 400 pF.
- Trace routing lengths should be monitored in the system such that the distributed capacitance, combined with the number of inputs per bus, does not exceed the I2C bus capacitance loading specification.
- See the I2C bus specification Version 2.1 or later for definition of requirements for 100 kHz operation.

8.7.2 Configuration Signal Implementation

In order to provide management addressing information, SLOT_ID pins are specified for each module in the SSI compliant chassis. The chassis midplane is responsible for properly defining the slot number in a particular chassis by setting the SLOT_ID pins to either 3.3V or GND.

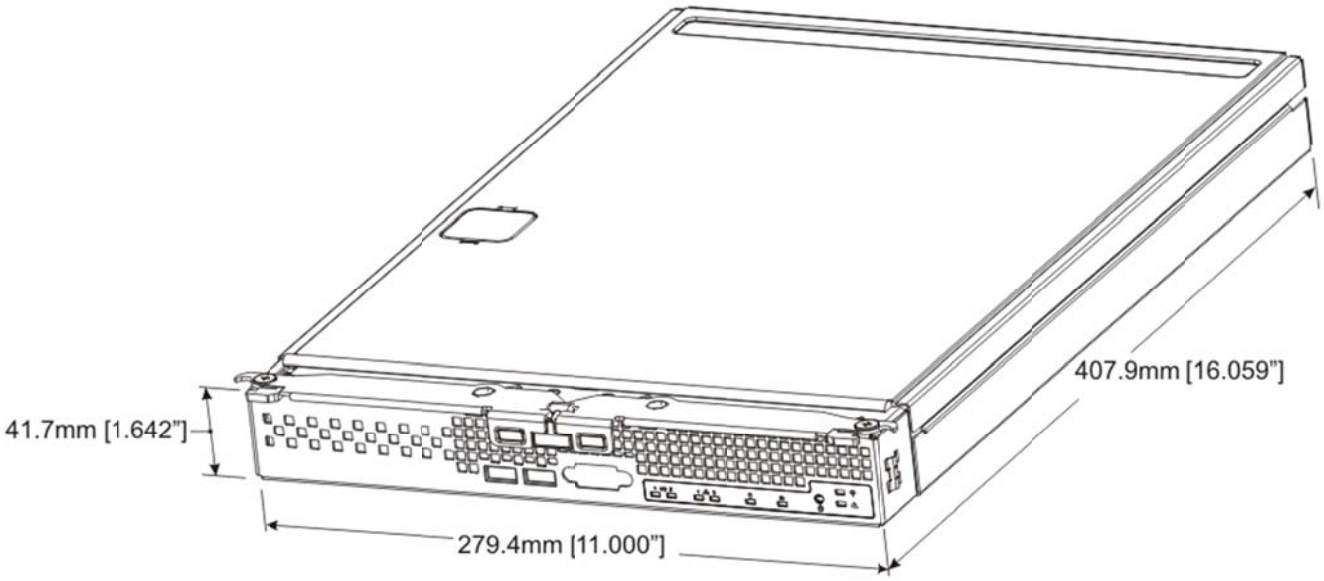
9 Compute Blade Module Mechanical Specifications

9.1 Enclosure Mechanical Specifications

The compute blade module volumetric is defined as 279.40 mm [11.000”], by 407.90 mm [16.059”], by 41.70 mm [1.642”]. See Figure 9-1.

The compute blade enclosure typically includes a top cover, the enclosure base, and two handle(s) for insertion/extraction.

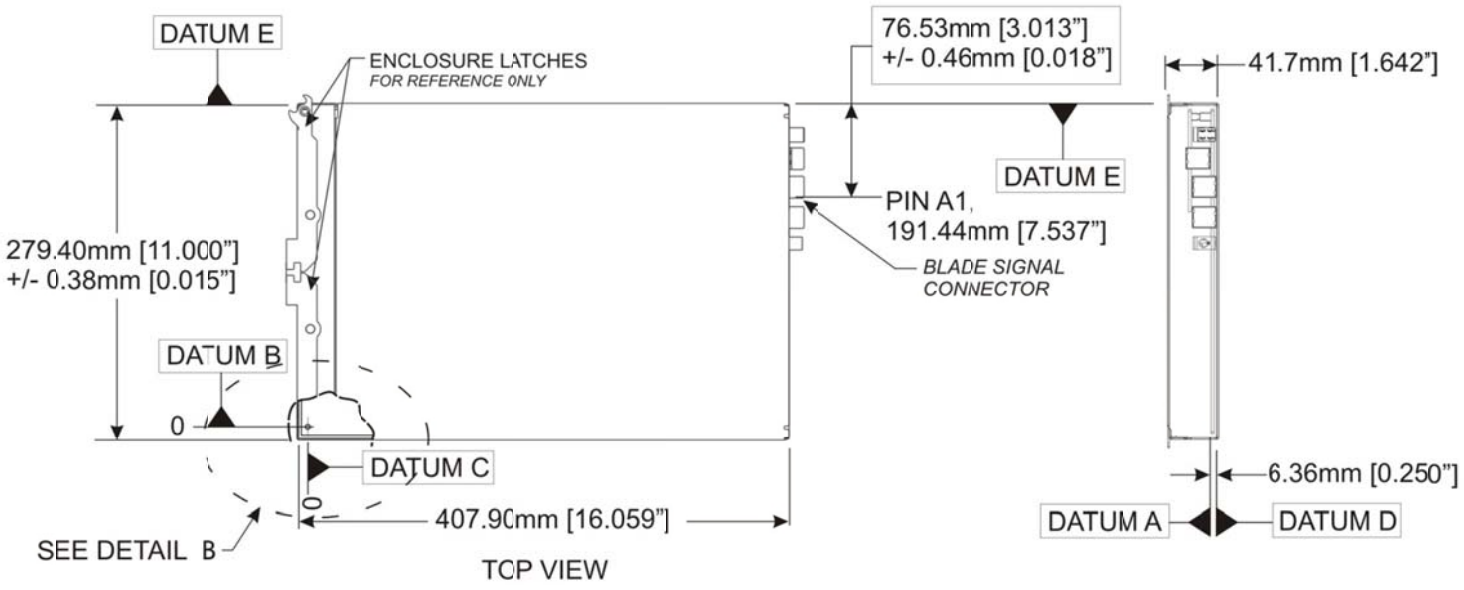
Figure 9-1: Single Wide Blade Module



9.1.1 Compute Blade Interaction with the System Server

Figure 9-2 and Figure 9-3 define the relationship between the compute blade PBA and the compute blade enclosure. Datum ‘D’ and Datum ‘E’ reference the system mounting surfaces for the compute blade. These Datums must be used in order to ensure tolerance capability between the compute blade and the system mounting surfaces and midplane signal connectors.

Figure 9-2: Compute Blade Enclosure



The relationship between the Blade Signal Connector's Pin A1 and the mounting surface, Datum 'E', is defined in Figure 9-2. This dimension must be adhered to within the shown tolerance range to ensure proper blade installation to the blade server chassis.

Figure 9-3 graphically describes the relationship between the compute blade PBA and the compute blade enclosure. Also depicted is the relationship between the compute blade PBA and the mounting latch. The mounting latch must engage the Blade Server chassis at the prescribed mounting surface as shown in Figure 9-3.

Figure 9-3: Detail B: Compute Blade Module Enclosure

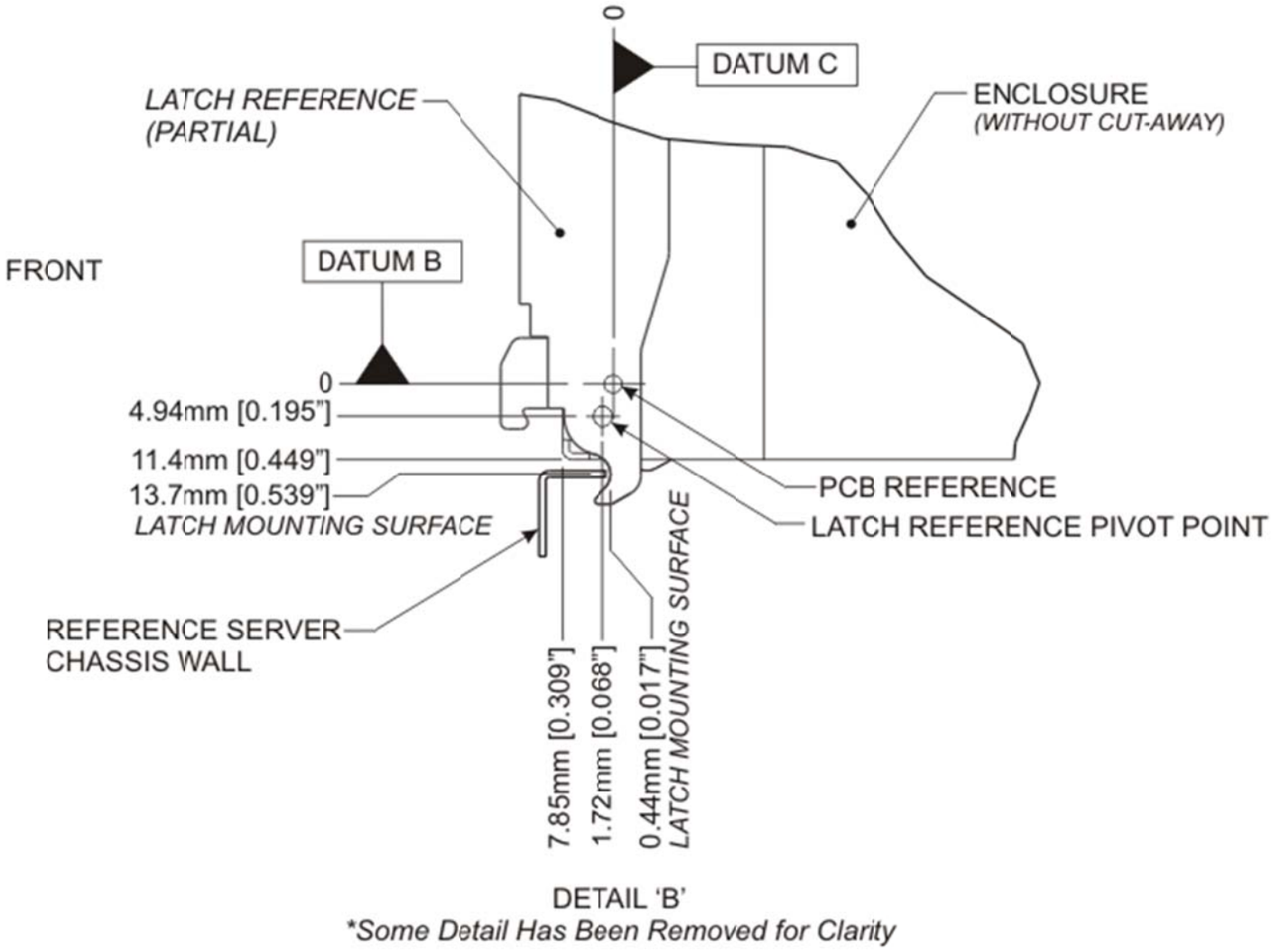


Figure 9-4: Compute Blade Module Enclosure

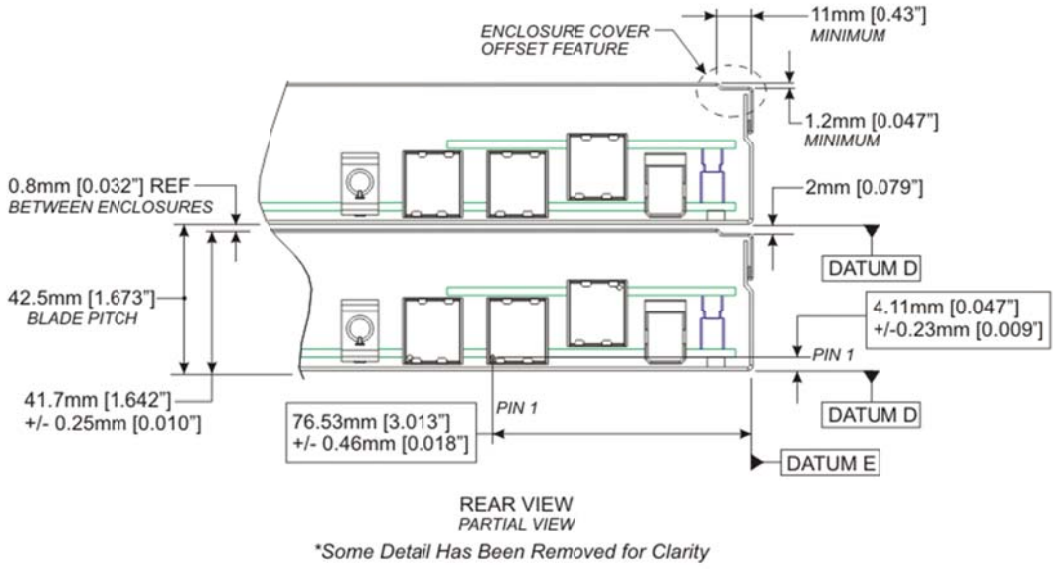


Figure 9-4 depicts two adjacent compute blades in a horizontal, server-installed configuration. Datum 'D' represents the horizontal compute blade-to-server mounting interface. Datum 'E' represents the vertical compute blade-to-server mounting interface. Both Datum's are referenced to the Compute Blade Signal Connector. The blade server chassis must preserve these relationships, with tolerance, by ensuring that these surfaces are properly referenced to the mating signal connectors on the blade server midplane PBA, referenced in Figure 3-4. These relationships must be preserved in order to effectively ensure that all connectors on the compute blade PBA properly engage all mating connectors on the server midplane PBA.

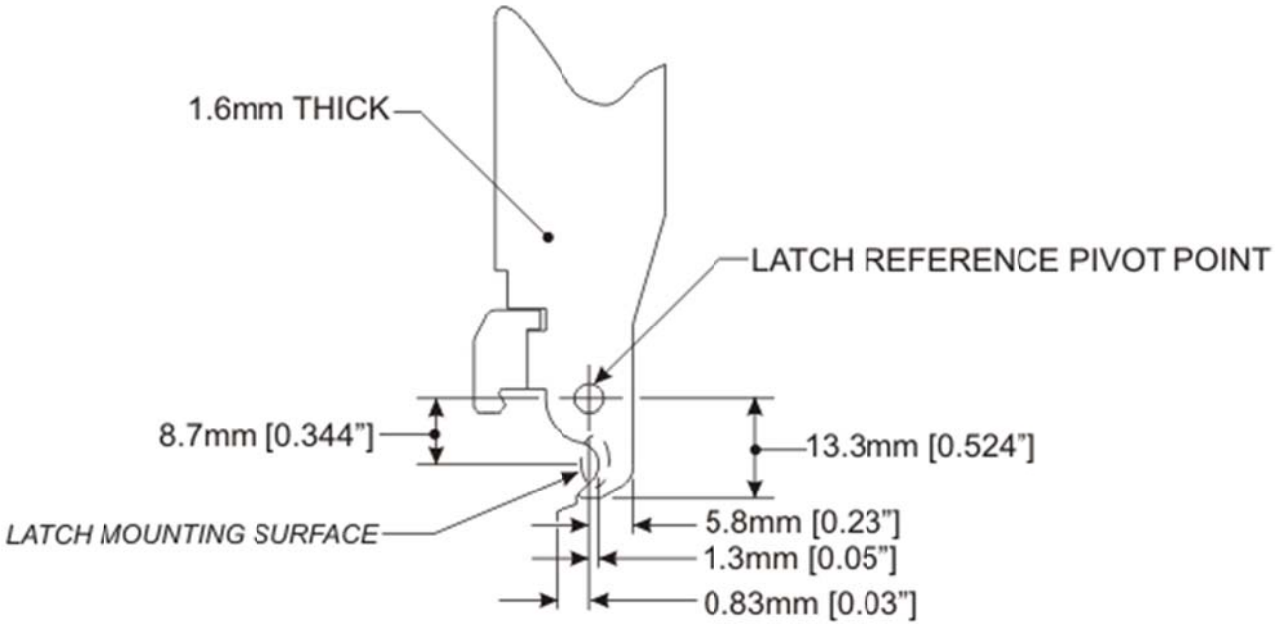
Each blade **shall** utilize an enclosure cover offset feature in order to provide clearance for server guides during installation. These guides must constrain each blade in its proper location within the server and guide it during installation until proper connector-to-connector engagement with the server midplane PBA is achieved.

9.2 Compute Blade Module Enclosure Latches

The latch **shall** perform to 500 N (110 lbs) at 250 cycles without performance degradation, limiting an injection/extraction force per handle to a maximum of 112 N (25 lbs). The handles should have a latching device and **shall** remain in the latched position during and after the earthquake test per ANSI T1E1.329 and IEC 61587-1. All conductive parts of the handles **shall** be connected to chassis ground. Handles **shall** have a provision for activating a switch that detects when the latch is open or closed. The switch **shall** be located on the front surface of the blade.

The enclosure latches should never be used to carry the Compute Blade as they may deform or break. An example latch design is depicted in Figure 9-5. Whereas the overall design of the latch may vary from vendor to vendor, the latching and mounting surfaces should be consistent with what is specified in Figure 9-5.

Figure 9-5: Blade Enclosure Latch Detail



LATCH DETAIL
PARTIAL VIEW
**Some Detail Has Been Removed for Clarity*

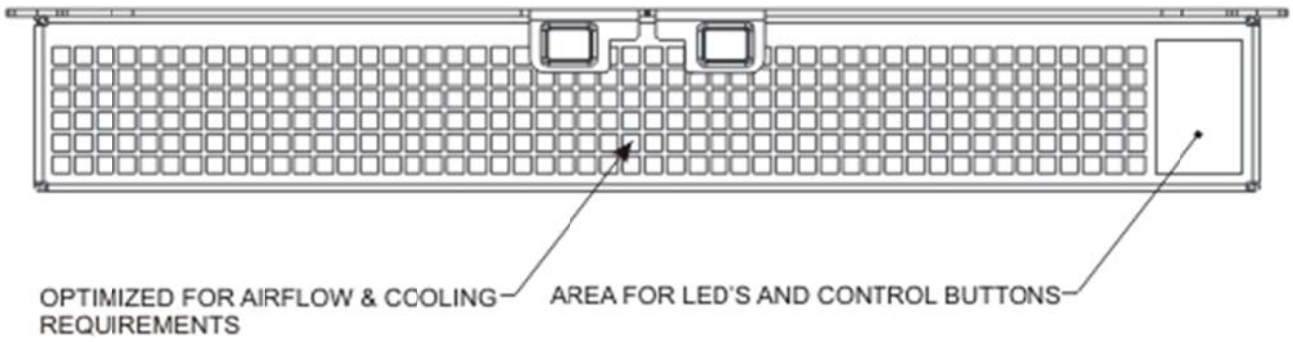
9.3 Covers

To prevent damage during handling and installation, the compute blade enclosure should utilize a top cover [143]. This document assumes that the top cover is fabricated from sheet metal that has a thickness of 0.08 mm [0.032”]. The top cover should securely latch to the blade enclosure and it should be removable without the need for tools.

9.4 Face Plate

The front surface of the blade enclosure should be perforated to allow as much airflow into the blade as required to adequately cool the internal components, such as processors, memory, chipsets, disk drives, etc. This requirement should be balanced with the need for sufficient EMI containment. Hole size and geometry must also be optimized to reduce EMI emissions. The face plate may also be used to increase blade airflow impedance if necessary in order to meet the specified flow impedance requirements. Components that block airflow into the blade **shall** be minimized or eliminated. Such components include latching handles, front panel controls, and labels. See Figure 9-6, below.

Figure 9-6: Compute Blade Enclosure Face Plate



9.4.1 Face Plate LEDs

The front surface of the blade enclosure should utilize LED indicators to signify activity within the blade. A green LED is typically used to indicate the “Power On” state. Another green LED is typically used to indicate “Storage Disk Activity.” An amber LED **shall** be used to indicate an issue within the compute blade that requires immediate attention. At least one set of green LEDs **shall** be used to indicate link activity between the blade and the chassis fabric.

The requirements for the Power, Fault State, and Storage LEDs are listed in the following table.

Table 9-1: Power, Storage, and Fault LEDs

LED	Color	State	Description
Power	Green	On	Steady on
Power	Green	Standby/Sleep	Spaced blink
Power	Green	Transition	Slow blink
Storage	Green/Yellow	Power + Activity	Green
Storage	Green/Yellow	Fault	Yellow
Fault State	Yellow	All good	Off
Fault State	Yellow	Recovering/Rebuilding	Slow blink
Fault State	Yellow	Degraded	Double blink
Fault State	Yellow	Locate	Fast blink
Fault State	Yellow	Fault detected	Steady on

The blink states for the LEDs are specified in the following table.

Table 9-2: LED Blink Speeds

Blink Rate	Description
Slow blink	1 Hz, equal on and off time
Fast blink	4 Hz, equal on and off time
Double blink	0.1 second on, 0.1 second off, 0.1 second on, 2.7 second off
Spaced blink	0.1 second on, 2.9 second off

The following colors are recommended for the LEDs.

Table 9-3: Recommended LED Colors

Color	Nominal Wavelength	Range
Green	565 nm	550-570 nm
Yellow	590 nm	583-593 nm

Green LEDs, indicating a link to the chassis fabric, should be steady on when link is established. When there is activity, the LED **shall** be in fast blink mode.

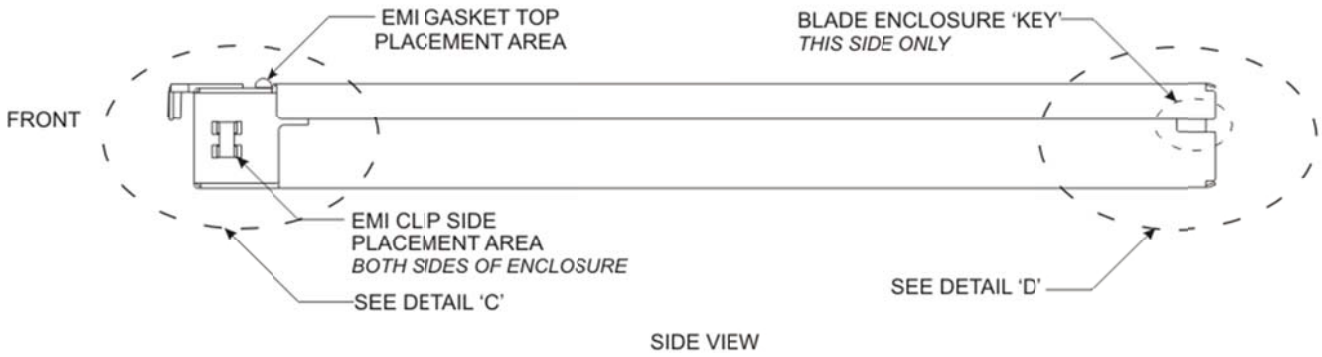
The face plate should also have an activation/deactivation request button. If a user wishes to perform a hot removal or swap of the blade, this button can be used to indicate to the Chassis Manager that a hot removal is being requested. The Chassis Manager can then, if appropriate, proceed to perform an orderly shut down of blade prior to hot removal of blade. Under certain conditions, the

hot removal request may be denied, e.g., if the blade is performing critical functions and should not be shut down.

9.5 EMI Gasket

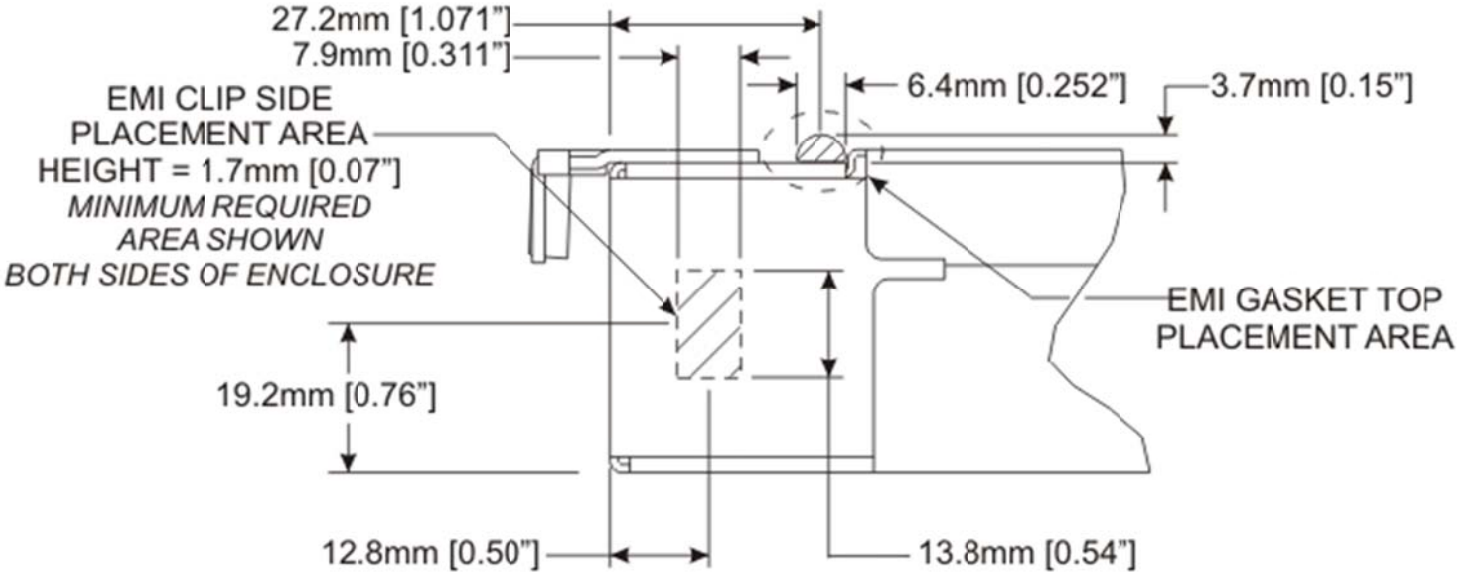
EMI containment **shall** be resolved within the compute blade design. The compute blade designer cannot ensure that the blade server will adequately contain EMI emitting from the compute blade. Flexible EMI gaskets are recommended on the top surface as shown in Figure 9-7 and Figure 9-8, and flexible metal wiping fingers are recommended on both sides of the compute blade enclosure as shown in Figure 9-7 and Figure 9-8. The EMI placement area of the adjoining compute blade should not be painted in order to increase EMI containment [143].

Figure 9-7 Compute Blade Enclosure EMI Locations



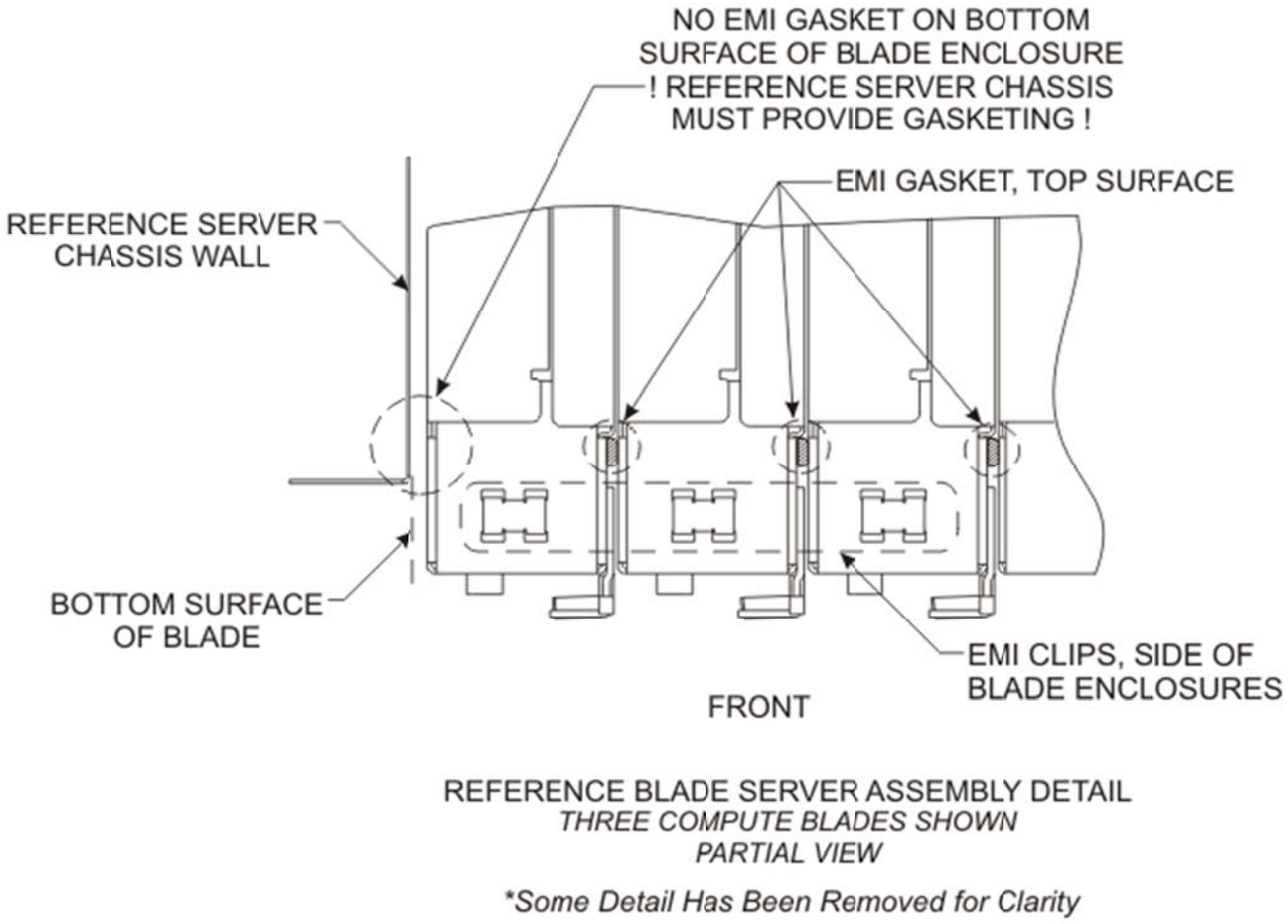
Recommended placement locations for the EMI containment gasket and EMI metal clips are shown in Figure 9-8, below.

Figure 9-8: Compute Blade Enclosure EMI Locations, Detail 'C'



When installed in a Blade Server chassis, the Blade Enclosure side EMI clips must engage the walls of the Blade Server chassis to ensure continuous EMI containment. Furthermore, the EMI gasket on the top surface of the Blade Enclosure must make continuous contact with the neighboring Blade Enclosure. Figure 9-9 demonstrates, however that special provision must be made by the Blade Server chassis to provide additional EMI gasketing for one of the Blade Enclosures due to the fact that no Blade Enclosure has EMI gasketing on the bottom surface of the enclosure. See Figure 9-9 for clarification.

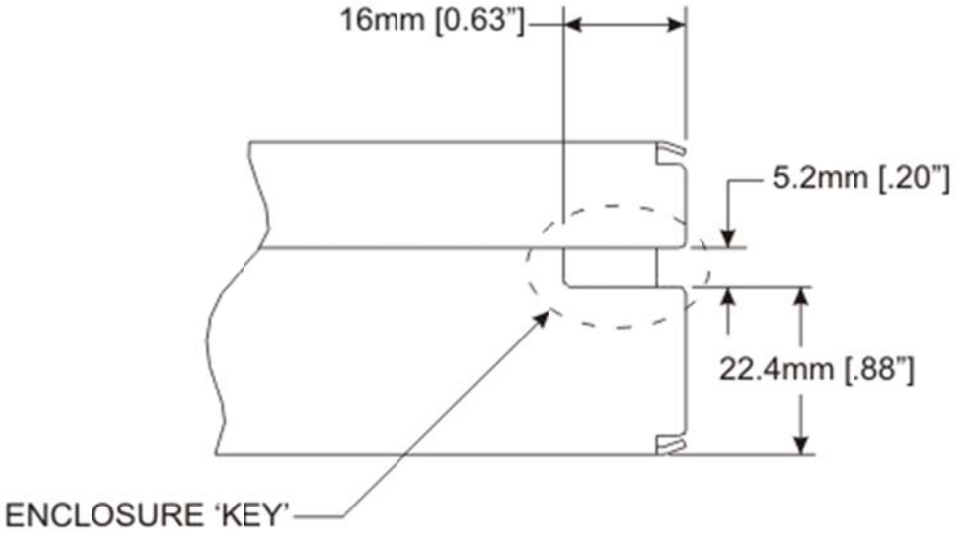
Figure 9-9: Reference Compute Blade Server Assembly Detail



9.6 Compute Blade Enclosure Keying

Improper installation of a Compute Blade Enclosure into the Blade Server chassis could damage the Compute Blade IO Connectors and/or the Blade Server Midplane PBA. In order to prevent damage each Compute Blade Enclosure **shall** incorporate the Enclosure 'key' or 'notch' as shown in Figure 9-7, and detailed in Figure 9-10, below. The compute blade chassis must provide a robust feature that engages the compute blade enclosure key when installed correctly, and prevents engagement when the compute blade is installed improperly.

Figure 9-10: Blade Enclosure Key Location, Detail 'D'



DETAIL 'D'
**Some Detail Has Been Removed for Clarity*

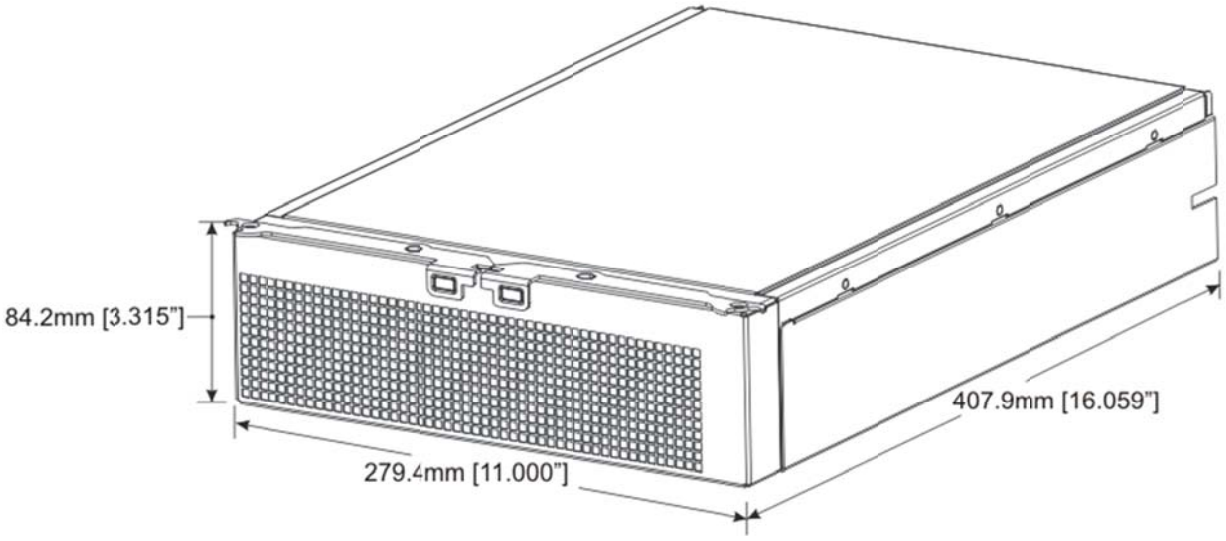
9.7 ESD Discharge Strip

The ESD discharge **shall** be provided through ground connections on the enclosure.

9.8 Provisions for Double-wide Compute Blade Enclosures

A double-wide compute blade, as defined here, is a blade that is comprised of two blades that are connected together to perform additional computing functions. These additional computing functions can include a four CPU socket design using two socket blades or a 4-socket design occupying two slot spaces using more than one PBA. The way in which the blades or PBAs are connected is up to the OEM and is out of scope for this specification. In a double-wide configuration, one blade **shall** perform all of the management functions for both blades. This blade is referred to as the main primary blade. The other blade is referred to as the auxiliary blade. The primary blade is the blade with the lowest Slot ID. The chassis designer should consider how to develop the compute blade guides to accept both single-wide and double-wide enclosures in the same chassis [143].

Figure 9-11: Double-wide Compute Blade Enclosure



9.8.1 Double-wide Compute Blade Mechanical Specifications

The double-wide compute blade module volumetric is defined as 279.40 mm [11.000”], by 407.90 mm [16.059”], by 84.20 mm [3.315”]. See the following figure.

The compute blade enclosure typically includes a top cover, the enclosure base, and latches for insertion/extraction.

Figure 9-12: Double-wide Compute Blade Module

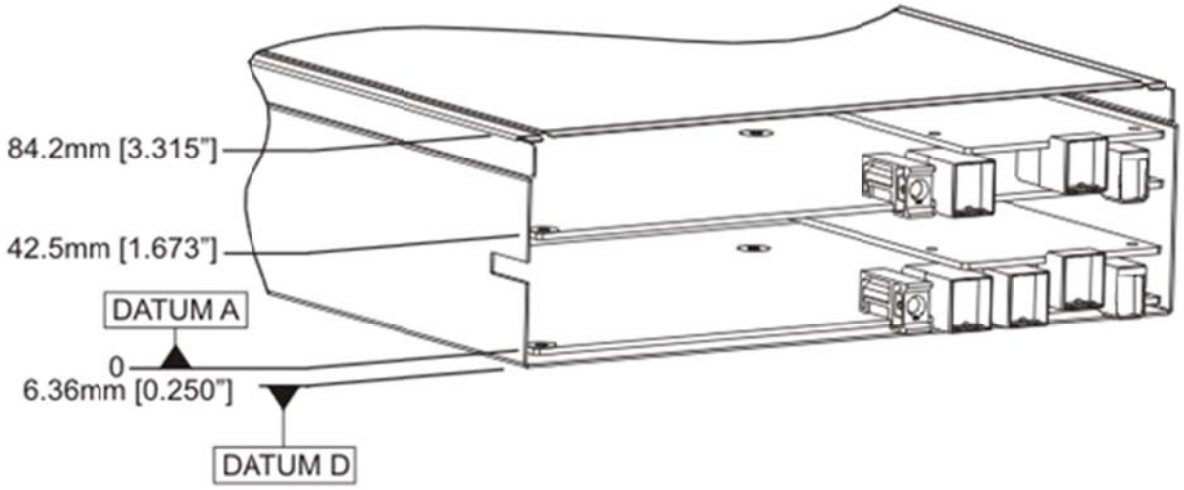
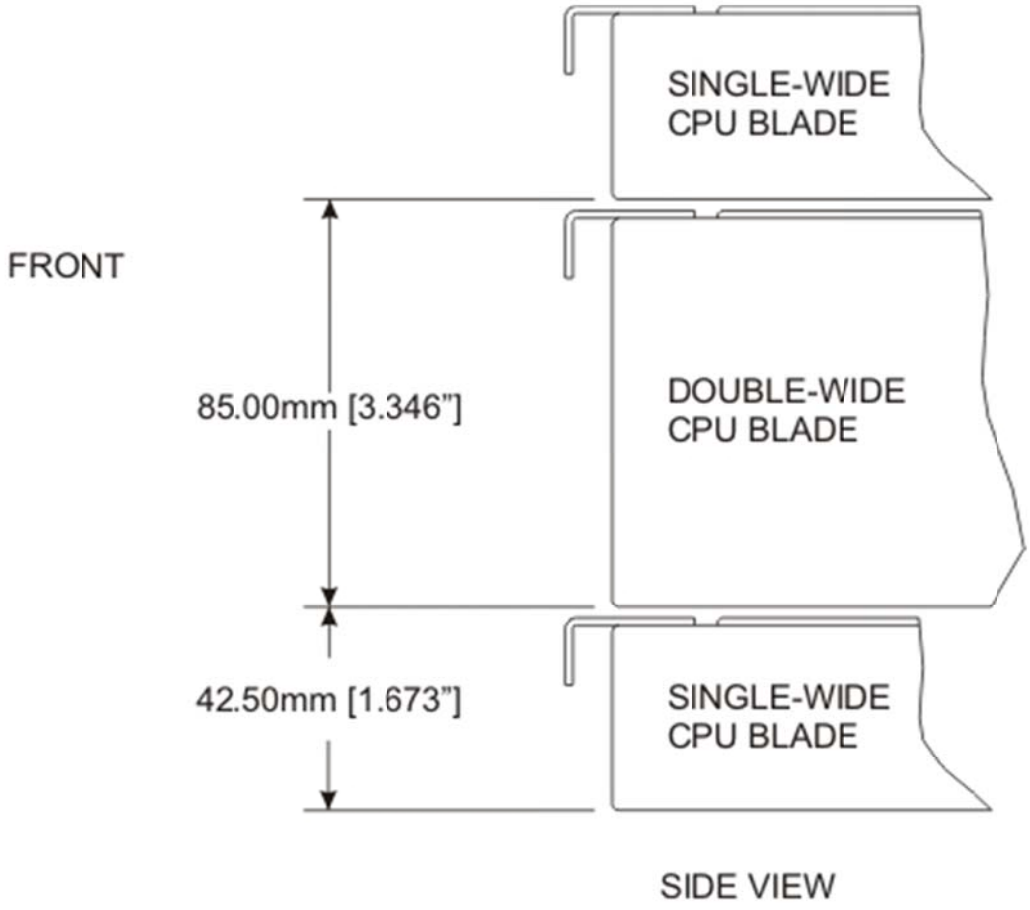


Figure 9-13: Double-wide Compute Blade Module Stack-Up



9.8.2 Double-wide Module Power Budget

Double-wide modules **shall not** exceed a total power dissipation of 900W. Power should be drawn from both of the power connectors. The requirements for these connectors are identical to those for the single-wide blade module. The power draw per-slot during initial power up **shall not** exceed 40W through each blade's auxiliary power circuit.

9.8.3 Power-up and Sequencing

The management controller of the primary blade (the blade with the lowest Slot ID) **shall** represent the module to the Chassis Manager for both blades. Only that management controller **shall** perform the discovery and power negotiation on behalf of the two blades. The module power requirements are negotiated on a per-slot basis.

For more double-wide module management requirements, see section 6.8.

9.8.4 Double-wide Module Form Factors

Double-wide modules are created by

- Using two single-width modules in tandem, sometimes mentioned as "two plus two". The height and depth of the PBAs remain the same and the pitch increases to 85.0 mm. The connector and the pinouts for interconnecting the two modules are OEM-defined and not a part of this specification.
- By designing a 4-socket solution using two-slot volumetrics. The actual functional partition, logic implementation and interconnection of those PBAs are OEM implementation specific and are not defined in this specification.

9.8.5 Double-wide DWM Midplane Connectors

The primary blade **shall** incorporate the midplane signal connectors. The connector **shall** be exactly identical to those that are specified in section 3.4.1. The way in which the signal connectors are used in designs that are created by using two identical blades is up to the OEM. Two possible options exist; this specification is not preferential:

- Use the primary blade signal connector only. Care should be taken to electrically de-activate the other signal connector.
- Use both of the signal connectors for additional bandwidth from the double-wide blade into the switch. The data transmission from the blade to the switch has to be managed by the application. This management is out of scope for this specification.

9.8.6 Equipment Environment Specifications

See Table 2.1 on page 10 of the ASHRAE "Thermal Guidelines for Data Processing Environments" ISBN 1-931862-43-5, Class 1, for blade operating temperatures, humidity and elevations.

10 Compute Blade Thermal Management

10.1 Introduction

Airflow requirements (and limitations) for the Compute Blade are determined by several factors including but not limited to: total heat dissipated, component selection, component density, component location, heat dissipation per device, acoustic targets, and practical limits of forced air cooling technology. The intent of this specification is to require a baseline air cooling solution for forward-looking designs. However, this baseline does not (and can not) comprehensively address all previously mentioned design variables. In addition to ensuring that baseline conditions are satisfied, designers and integrators are encouraged to make sure that component and/or module level thermal requirements are met given their own unique implementations.

10.2 Equipment Environment Specifications

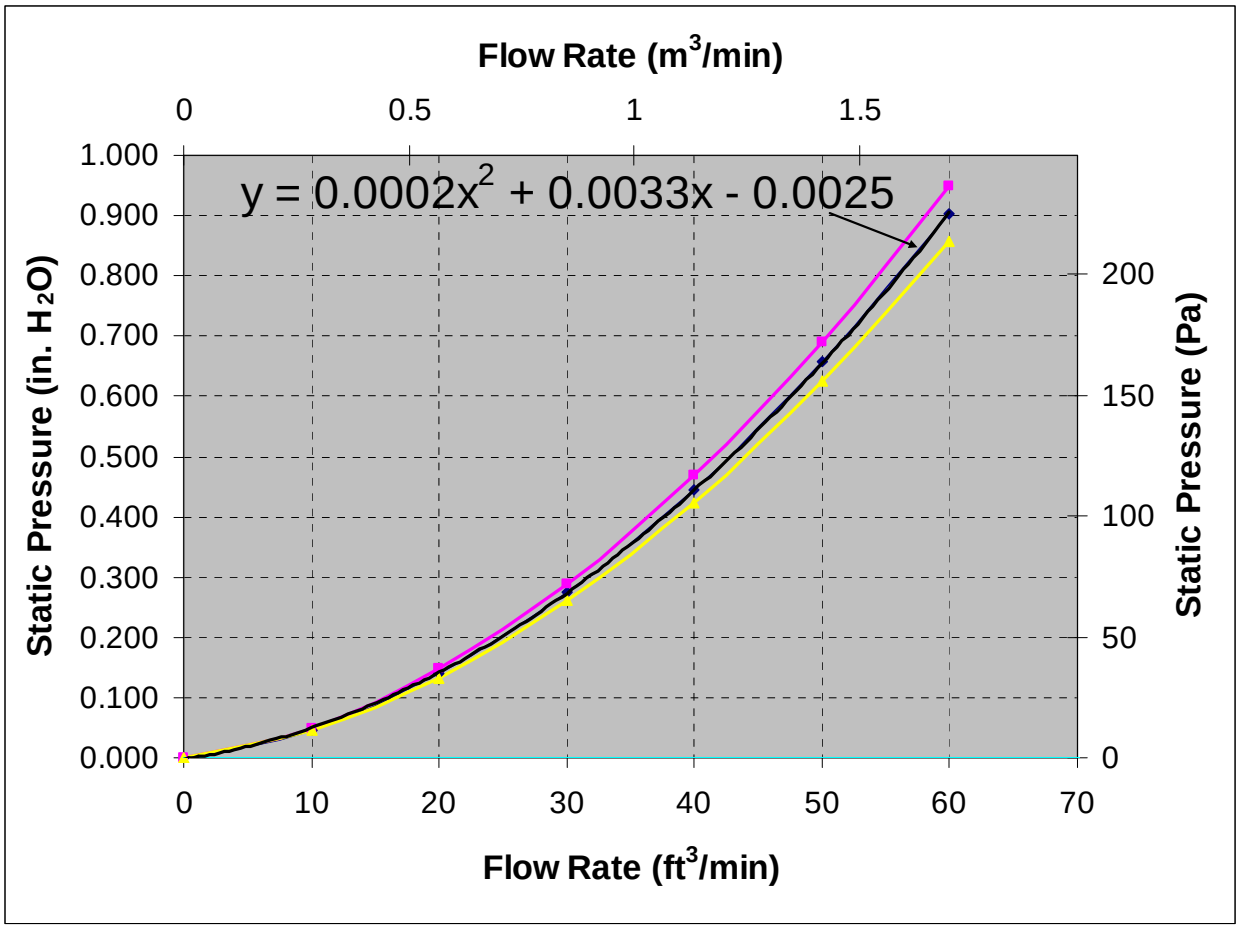
See Table 2.1 on page 10 of Reference ASHRAE "Thermal Guidelines for Data Processing Environments" ISBN 1-931862-43-5, Class 2, for blade operating temperatures, humidity and elevations.

10.3 Compute Blade Airflow/Cooling Requirements

Each Compute Blade, when installed into a system, **shall** be enabled to achieve a flow rate of 55CFM in performance mode and 45CFM with a failed fan condition. Careful design is required to ensure proper fan speed versus compute blade thermal requirements and feedback loops. Each design will require a detailed analysis and testing to verify that final cooling goals are met. The analysis **shall** include mechanical tolerances of removable components and address issues such as varying DIMM pcb thickness. [155]

While this specification does not address chassis impacts into which a compute blade will be installed, the designer will need to consider these external parameters when placing components and designing their cooling solution. In order for various vendor blades to be cooled properly in the same chassis, each blade **shall** match the airflow impedance curve as shown in Figure 10-1. Design variability is ± 5%. The impedance curve can be determined by numerical analysis or empirical testing. Final design verification testing **shall** be performed by means of an empirical test. See Figure 10-2 for direction on how to perform an impedance test, and refer to Appendix B.

Figure 10-1: Blade Airflow Rate Versus Impedance

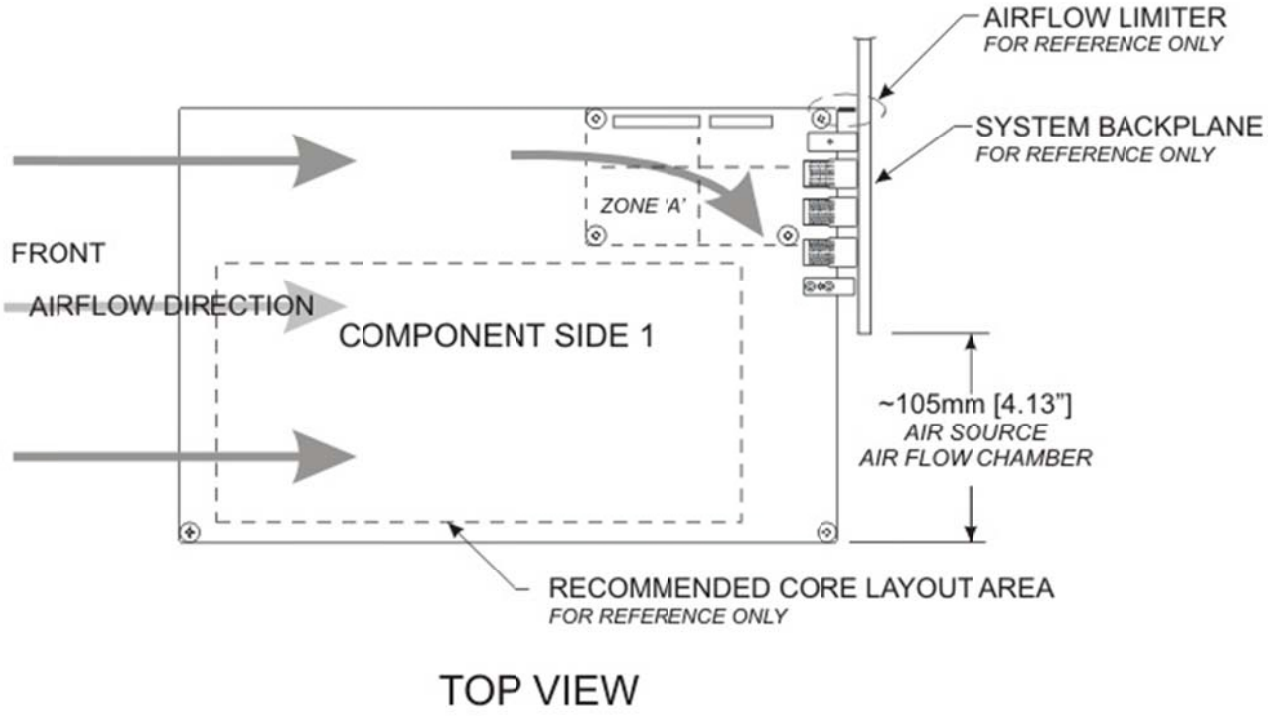


10.3.1 Air Distribution and Component Placement Considerations in a Compute Blade

Air flow direction in the system chassis will be determined by the deployment usage model and the system chassis should be designed accordingly to cool the compute blade in that environment. Air distribution within the blade will be impacted by components used: heatsinks, DIMMs, ducting, disk drives, etc., and the system integrator **shall** ensure their blade flow paths cool all components adequately. It is strongly recommended that key components (such as processors and memory) NOT be placed behind large airflow disruptors (such as hard disk drives). It is recommended that high-power devices located on the mezzanine card be located in Zone A, which is where the optimal flow rate is anticipated to be. External influences to the blade distribution could include the chassis the blade is placed into and midplane design.

The airflow direction on a compute blade **shall** be from the front to the rear of the compute blade. See Figure 10-2.

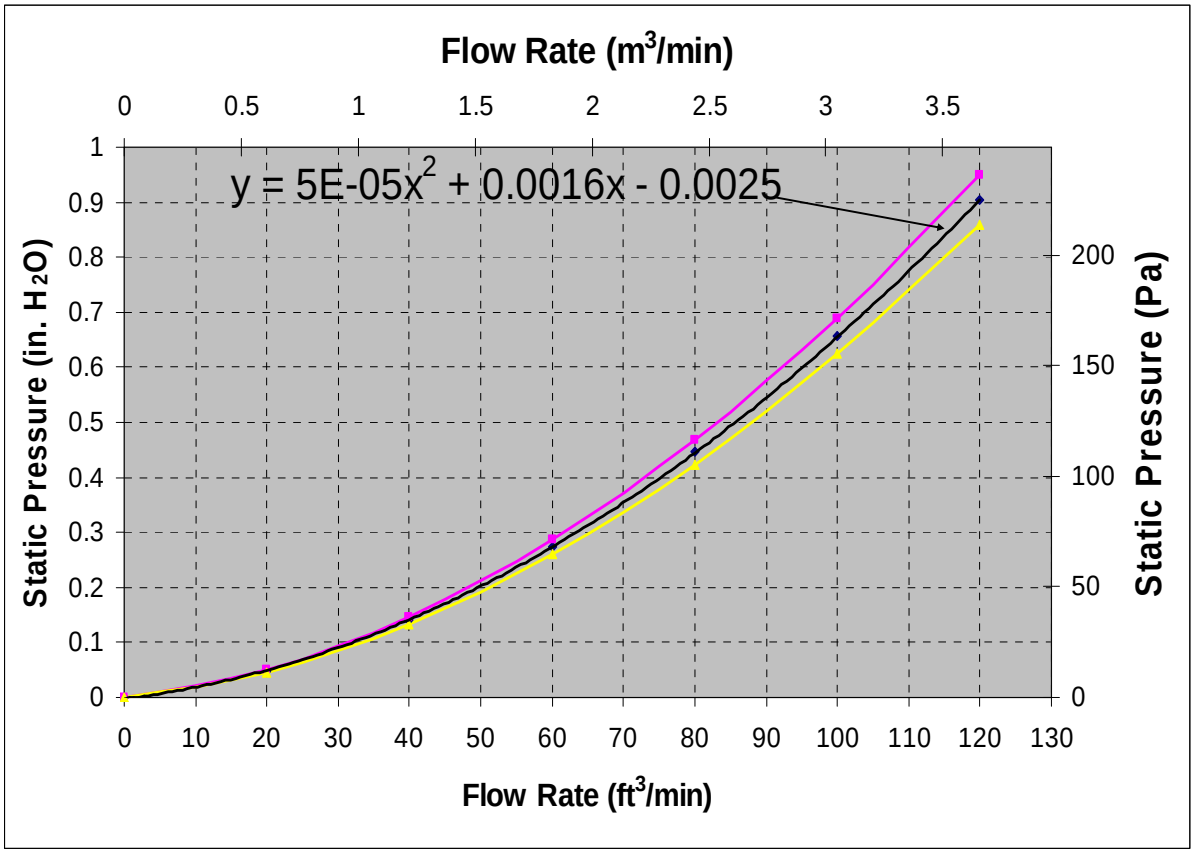
Figure 10-2: Compute Blade Airflow Distribution



10.3.2 Double-wide Compute Blade Thermal Management

Each blade, when installed into a system, **shall** be enabled to achieve a maximum flow rate of 110CFM at maximum fan speeds. Each design will require detailed analysis and testing to verify final cooling goals are met. While this specification does not address chassis impacts on where a blade will be placed, the designer will need to consider these external parameters when placing components and designing their cooling solution. In order for various vendor blades to be cooled properly in the same chassis, each blade **shall** match the airflow impedance curve as shown in Figure 10-3. Design variability is ± 5%. See Figure 10-2 for direction on how to perform the impedance test. The impedance curve can be determined by numerical analysis or empirical testing. Final design verification testing **shall** be performed by means of an empirical test.

Figure 10-3: Double-wide Blade Airflow Rate Versus Impedance



10.3.3 Equipment Environment Specifications

See Table 2.1 on page 10 of the American Society of Heating, Refrigerating, and Air Conditioning Engineers (ASHRAE) "Thermal Guidelines for Data Processing Environments" ISBN 1-931862-43-5, Class 2, for blade operating temperatures, humidity and elevations.

11 Product Regulations Compliance

The blade product **shall** meet regulatory requirements as governed by specific country regulations.

A Computation Supporting the Electrical Channel Specifications

A.1 Average Insertion Loss Slope m_a and Intercept b_a

For "N" points between frequency range f1 to f2 the average insertion loss slope and intercept are defined in Equation A-1 to Equation A-4.

Equation A-1

$$f_{avg} = \frac{1}{N}\sum_{n} f_n$$

Equation A-2

$$IL_{avg} = \frac{1}{N}\sum_{n} IL(f_n)$$

Equation A-3

$$m_A = \frac{\frac{1}{N}\sum_{n}\left(f_n - f_{avg}\right)\cdot\left(IL(f_n) - IL_{avg}\right)}{\sum\left(f_n - f_{avg}\right)^2}$$

Equation A-4

$$b_A = IL_{avg} - m_a \cdot f_{avg}$$

A.2 Insertion Loss Fit A(f)

The insertion loss fit A(F) is defined in Equation A-5.

Equation A-5

$$A(f) = m_a \cdot f + b_A$$

A.3 Insertion Loss to Crosstalk Ratio

A.3.1 Power Sum Differential Near-end Crosstalk PSNEXT(f) from n of N Aggressors NEXT(f) in dB

Equation A-6

$$NEXT_n(f) = 10 \cdot \log(\left| sdd21_{next}(f)_n \right|)$$

Equation A-7

$$PSNEXT(f) = -10 \cdot \log(\sum_n 10^{-\frac{NEXT_N(f)}{10}})$$

A.3.2 Power Sum Differential Far-end Crosstalk PSFEXT(f) from n of N Aggressors FEXT(f) in dB

Equation A-8

$$FEXT_n(f) = 10 \cdot \log(\left| sdd21_{fext}(f)_n \right|)$$

Equation A-9

$$PSFEXT(f) = -10 \cdot \log(\sum_n 10^{-\frac{FEXT_N(f)}{10}})$$

A.3.3 Power Sum Differential Crosstalk PSXT(f)

Equation A-10

$$PSXT(f) = -10 \cdot \log(\sum_{n} 10^{-\frac{PSFEXT(f)}{10}} + \sum_{n} 10^{-\frac{PSNEXT(f)}{10}})$$

A.3.4 Insertion Loss to Crosstalk Ratio ICR(f)

Equation A-11

$$ICR(f) = -IL(f)) + PSXT(f)$$

A.3.5 Average Insertion Loss to Crosstalk Ratio Log-log Slope m_{icr} and Intercept b_{icr}

Equation A-12

$$x_{avg} = \frac{1}{N} \sum_{n} \log(f_n)$$

Equation A-13

$$ICR_{avg} = \frac{1}{N} \sum_{n} ICR(f_n)$$

Equation A-14

$$m_{ICR} = \frac{\frac{1}{N} \sum_{n} (\log(f)_n - \log(f_{avg})) \cdot (IL(f_n) - IL_{avg})}{\sum (\log(f)_n - f_{avg})^2}$$

Equation A-15

$$b_{ICR} = ICR_{avg} - m_{ICR} \cdot x_{avg}$$

A.3.6 Insertion Loss to Crosstalk Ration Fit $ICR_{fit}(f)$

Equation A-16

$$ICR_{fit}(f) = m_{ICR} \cdot \log(f) + b_{ICR}$$

A.3.7 Minimum Insertion Loss to Crosstalk Ratio

Equation A-17

$$ICR_{fit}(f) \geq ICR_{\min}(f) = ICRa - ICRb \cdot \log(\frac{f}{f_{icr}})$$

B Airflow Impedance Test

The following pictures show an example of a blade airflow impedance test.

Figure B-1: Blade Airflow Impedance Test

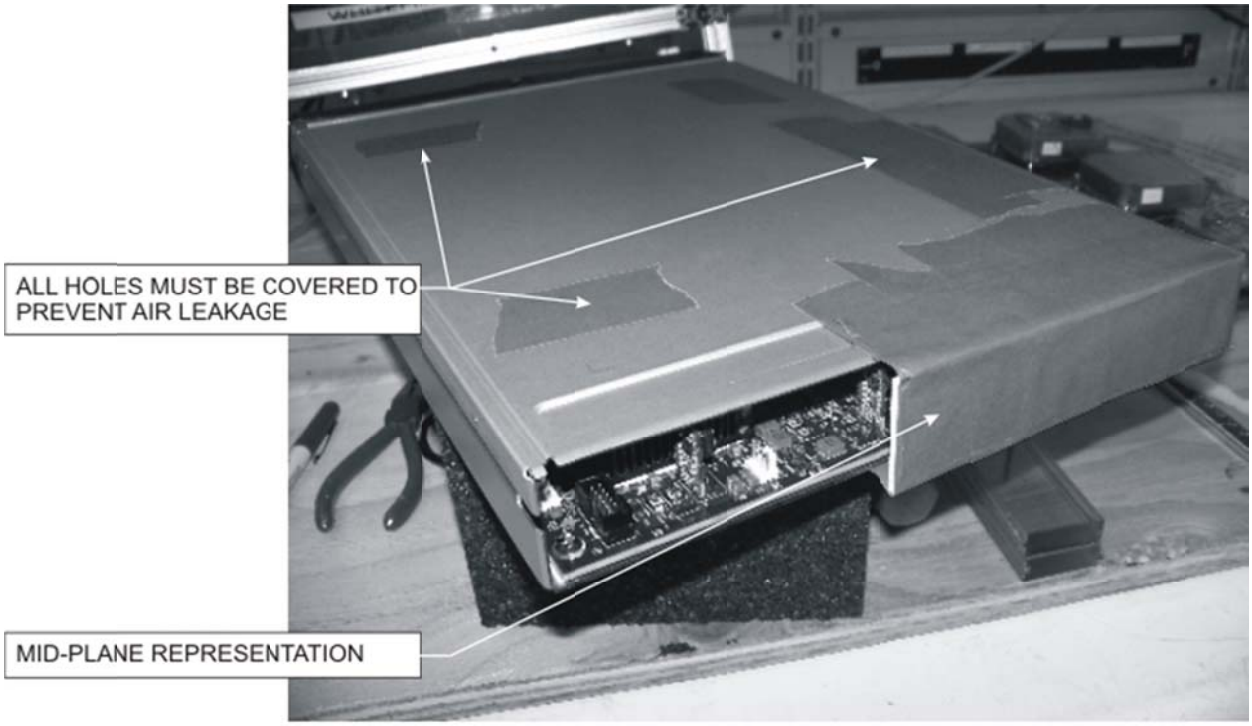


Figure B-2: Blade Airflow Impedance Test