EMERSON
Network Power

System Management Interface Based on HPI-B (Centellis 4620)

User's Guide

6806800D85A

July 2008

Contact Address

Emerson Network Power - Embedded Computing

Lilienthalstr. 15

85579 Neubiberg/Munich

Germany

Contents

List of Tables

List of Figures

About this Manual

Overview of Contents

This manual is divided into the following chapters and appendices.

Abbreviations

This document uses the following abbreviations:

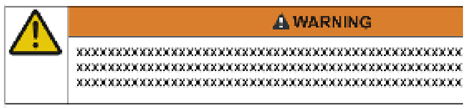
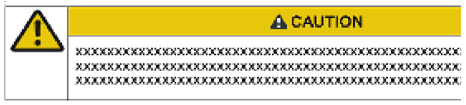
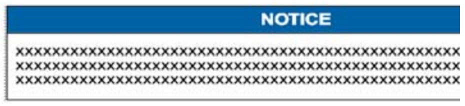
Abbreviation	Description
AMC	Advanced Mezzanine Module
ATCA	Advanced Telecom Computing Architecture
BT	Block Transfer
CGE	Carrier Grade Edition
CPIO	Copy In/Out
CPU	Central Processing Unit
ECC	Embedded Communications Computing
FRU	Field Replaceable Unit
HPI	Hardware Platform Interface
IA	Intel Architecture
ID	Identifier
IP	Internet Protocol
IPMI	Intelligent Platform Management Interface

Abbreviation	Description
LAN	Local Area Network
MVL	Montavista Linux
OEM	Original Equipment Manufacturer
PICMG	PCI Industrial Computer Manufacturers Group
RMCP	Remote Management Control Protocol
RPM	RedHat Package Manager
SAF	Service Availability Forum
SAI	Service Availability Interface
SAIM	Service Availability Interface Mapping
SAM	Shelf Management Alarm Module
SMI	Serial Management Interface
ShMC	Shelf Management Controller

Conventions

The following table describes the conventions used throughout this manual.

Notation	Description
0x00000000	Typical notation for hexadecimal numbers (digits are 0 through F), for example used for addresses and offsets
0b0000	Same for binary numbers (digits are 0 and 1)
bold	Used to emphasize a word
`Screen`	Used for on-screen output and code related elements or commands in body text
`Courier + Bold`	Used to characterize user input and to separate it from system output
Reference	Used for references and for table and figure descriptions
File > Exit	Notation for selecting a submenu
<text>	Notation for variables and keys
[text]	Notation for software buttons to click on the screen and parameter description
...	Repeated item for example node 1, node 2, ..., node 12
. . .	Omission of information from example/command that is not necessary at the time being
..	Ranges, for example: 0..4 means one of the integers 0,1,2,3, and 4 (used in registers)

Notation	Description
|	Logical OR
	Indicates a hazardous situation which, if not avoided, could result in death or serious injury
	Indicates a hazardous situation which, if not avoided, may result in minor or moderate injury
	Indicates a property damage message
	No danger encountered. Pay attention to important information

Summary of Changes

This manual has been revised and replaces all prior editions.

Part Number	Publication Date	Description
6806800D85A	July 2008	Early access version

Comments and Suggestions

We welcome and appreciate your comments on our documentation. We want to know what you think about our manuals and how we can make them better.

Mail comments to us by filling out the following online form:
http://www.emersonnetworkpowerembeddedcomputing.com/ > Contact Us > Online Form

In "Area of Interest" select "Technical Documentation". Be sure to include the title, part number, and revision of the manual and tell us how you used it.

System Management Interfaces Overview

1

1.1 Introduction

Emerson provides an SAF Hardware Platform Interface (HPI) as part of its AdvancedTCA platforms. HPI provides an industry standard interface to monitor and control highly available telecommunications system platforms. The ability to monitor and control these platforms is provided through a consistent and standard set of programmatic interfaces that are targeted for adoption by the telecom building block industry to significantly reduce product time-to-market and development costs while retaining or enhancing total system/network availability.

HPI provides the interface between the middleware software solution stack and the hardware solution stack, allowing portability of middleware software building blocks across many different hardware platforms and portability of hardware platforms across many different middleware software building blocks.

This guide describes the HPI-B implementation targeted at the Emerson AdvancedTCA Centellis 4620 platform.

1.2 Standard Compliances

The Emerson HPI-B implementation for the Centellis 4620 environment is compliant to the following standards.

Table 1-1 HPI-B Standards Supported by Emerson HPI-B Implementation

Standard	Description
SAI-HPI-B.01.02	HPI-B base specification. It abstracts hardware platform characteristics into a data model consisting of entities and resources.
SAIM-HPI-B.02.01-ATCA	HPI-B-AdvancedTCA mapping specification. It provides a vendor independent hardware platform view of an AdvancedTCA system.

Although the standards listed above are fulfilled, some specific limitations apply. For details, refer to *Limitations* on page 27.

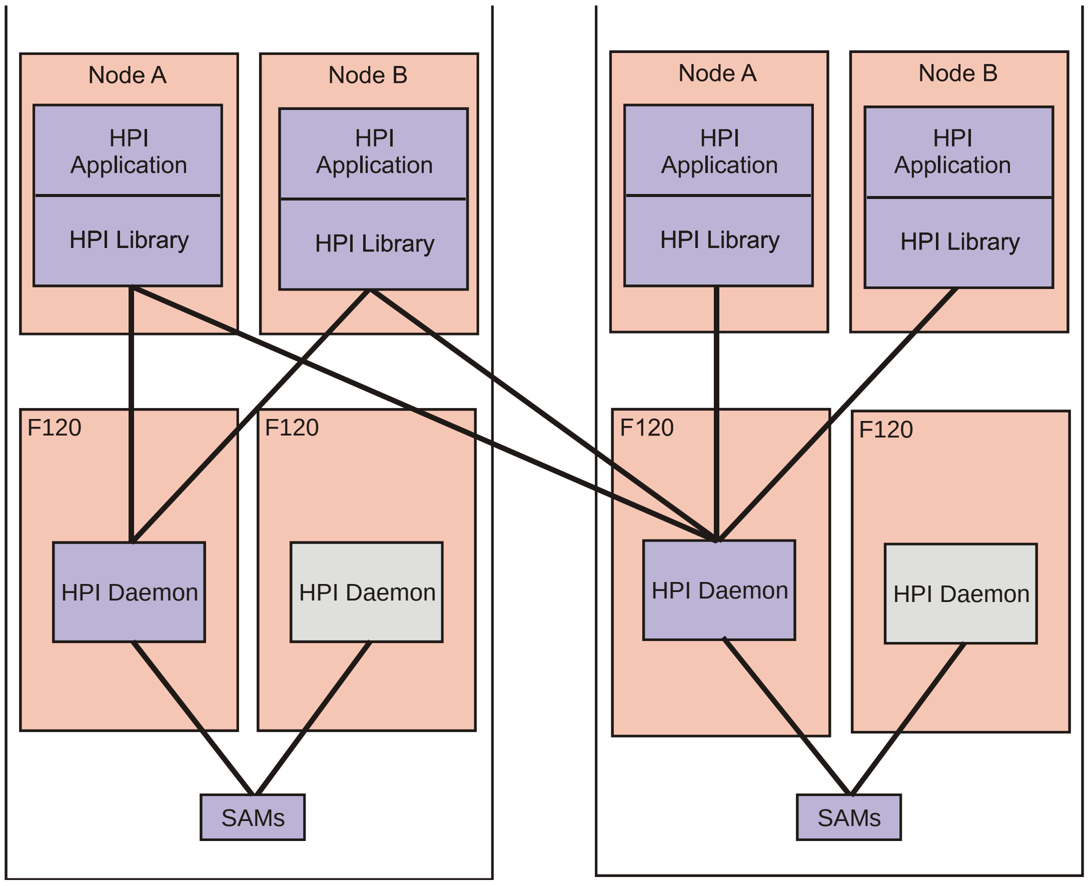
1.3 Architecture

The Emerson HPI implementation is provided in the form of a client-server architecture. In the Centellis 4620 system environment the server, called HPI daemon, runs on the ATCA-F120 switch blade, and the client, which is constituted by an HPI library and an application which links that library, runs on any node within a shelf. Internally each HPI daemon is connected to a Shelf

Management Alarm module (SAM) in the shelf. The communication between HPI daemon and SAM is realized via RMCP (Remote Management Control Protocol) which is an IP-encapsulation of IPMI commands. The communication between HPI daemon and HPI client is realized by an IP-based remote HPI communication protocol.

An HPI client may access one or more HPI daemons, and on the other hand, an HPI daemon may be accessed by one or more HPI clients. The following figure illustrates this.

Figure 1-1 Overview of HPI Usage in Systems



1.3.1 HPI Library

The HPI library is the primary user interface. It is intended to be used by applications that wish to control and monitor HPI managed components, such as ATCA shelves, blades and other FRUs. The HPI library is delivered as shared and as static library and has to be linked with an application.

There are two types of HPI libraries available:

- Single shelf library
- Multi shelf library

The single shelf library supports the communication with one HPI daemon only. Since it is only used internally and is not intended to be used by user applications, it will not be described any further in this manual. The multishelf library, on the other hand, supports the communication with one up to several HPI daemons. This becomes necessary if you wish to deploy redundancy in one HPI-B based shelf management system or if you wish to manage several shelves. The multishelf library is the library which you should build your applications on, it is the official interface to customer applications.

Details about supported combinations of CPU architecture/Linux distribution are given in Chapter 2, *Software Installation and Configuration,* on page 17.

1.3.2 HPI Daemon

The HPI daemon within an Centellis 4620 system runs on the ATCA-F120 blades and its main tasks are:

- Provide a single access point to control and monitor hardware components in a shelf
- Map information provided by the underlying Shelf Manager to HPI

1.4 High Availability

The Emerson HPI-B implementation described in this manual supports the following two redundancy options:

- Active/active
- Cold-standby

The active/active option is the default configuration. Both HPI daemons in shelf are active and run simultaneously. Your application connects to one daemon and if the connection fails it connects to the second daemon. Note that the current HPI-B implementation does not replicate any data between the two daemons, this means data consistency is not guaranteed.

Alternatively you may choose to use the cold-standby redundancy option. In this configuration your application must make sure that only one HPI daemon is active at a time. If the daemon fails, your application starts up the second, previously inactive daemon and connects to it. During start-up the HPI daemon scans the current system environment. This way it is ensured that the daemon reflects the current system configuration.

Software Installation and Configuration 2

2.1 Overview

This section describes how to set-up HPI-B daemons and clients and how to install all files needed to run client applications. Generally, all files are delivered in the form of RPMs. The content of an RPM is reflected in its naming scheme.

The following table describes the used naming scheme of the client and daemon RPMs. The placeholder "architecture" stands for the supported CPU architecture of the respective blade where the client or daemon is to run and can be any of the following:

- x86
  Intel IA-32 bit blades, such as PrAMC-7210/7211
- x86_64
  Intel IA-64-bit blades, such as the ATCA-7221
- ppc_e500v2
  PowerPC based blades, such as the ATCA-F120

Table 2-1 RPM Files for HPI-B Clients and Daemons

RPM File Name	Description
`bbs-hpib-<version>-1.<architecture>-<distribution>-<OS>.rpm`	This RPM is the HPI-B base package. It contains shared libraries to be used by HPI-B clients and daemons, as well as compiled example applications and client configuration files. This package is required both by HPI-B daemons and clients.
`bbs-hpib-daemon-<version>-1.<architecture>-<distribution>-<os>.rpm`.	This RPM contain all files which are related to the HPI-B daemon: executables, libraries and configuration file.

The files required for the HPI-B daemon come as part of the ATCA-F120 software. Depending on the particular ATCA-F120 release, the HPI-B daemon files are already preinstalled or not. Check the respective documentation of the ATCA-F120 itself and of the system where the ATCA-F120 is used.

If the HPI-B software is not preinstalled or if you want to upgrade the installed HPI-B software, then you can obtain the daemon files as an RPM file. The file can be obtained from Emerson.

2.2 Installing and Configuring the HPI-B Daemon

This section describes how to install and configure an HPI-B daemon.

2.2.1 Installing an HPI-B Daemon

As previously mentioned, it depends on the ATCA-F120 release if the HPI-B daemon files are preinstalled or not. If they are not preinstalled or if you want to upgrade existing HPI-B files, you need to obtain the desired RPM files and install them manually as follows.

Installing the HPI-B Daemon Files

In order to install/upgrade the HPI-B daemon files on an ATCA-F120 blade, proceed as follows.

1. Connect to the ATCA-F120 blade where you wish to install the HPI-B daemon files.
2. Copy the RPM file to the ATCA-F120.
3. If applicable, enter **`rpm -e <Old HPI-B daemon package name>`** to uninstall existing daemon files
4. If applicable, enter **`rpm -e <Old HPI-B client base package name>`** to uninstall an installed HPI-B client base package
5. Enter **`rpm -i <New HPI-B client base package RPM>`**
   This installs the new HPI-B client base package files.
6. Enter **`rpm -i <New HPI-B daemon RPM>`**
   This installs the new daemon files.

The following table lists all HPI-B related directories available on the ATCA-F120 after the installation.

Table 2-2 Overview of HPI-B Directories and Files on ATCA-F120

Directory	Description
`/opt/motorola/bin`	Contains HPI-B daemon binaries and compiled client example applications.
`/opt/motorola/etc/bbs-hpib`	Contains configuration files used to configure the HPI-B daemon and clients. See *Configuring an HPI Daemon* on page 18 for details on configuring the HPI daemon.
`/etc/init.d`	Daemon start/stop script
`/opt/motorola/lib`	Shared libraries

2.2.2 Configuring an HPI Daemon

At start-up, the HPI daemon reads the following configuration file: `/opt/motorola/etc/bbs-hpib/bbs-hpib.conf`.

The only entry which you must configure manually once is the IP address and port used to access the SAMs in the Centellis 4620. Both SAMs have a virtual IP address assigned to them which is to be used specifically for RMCP-based accesses. This is the IP address which you must specify in the configuration file.

The corresponding section in the configuration file is called `ipmidirect`. Within `ipmidirect`, you need to adapt the entries `addr` and `port`, where `addr` is the IP address and `port` is the port.

In a Centellis 4620 shelf the virtual IP address used to access the SAMs via RMCP is 192.168.24.11 and the port number is 623, by default. Thus, an entry in your configuration file should look as follows:

```
handler ipmidirect {
   entity_root = "{ADVANCEDTCA_CHASSIS,0}"
   name = "lan"         # RMCP
   addr = "192.168.24.11"
   port = "623"         # RMCP port

...
```

Both SAMs are operated in an active/stand-by mode. An internal redundancy mechanism ensures that the HPI-B daemon is always connected to the currently active SAM. Since a virtual IP address is used, this is transparent to the HPI-B daemons and HPI-B client applications.

2.3 Setting Up HPI Clients

This section describes how to install/configure HPI clients on node blades.

2.3.1 Installing HPI Clients

Procedure

In order to install/upgrade an HPI-B client package on a node blade, proceed as follows.

1. Connect to the node blade where you wish to install the HPI-B client package.
2. Copy the RPM file that you wish to install to the node blade. Refer to Table "RPM Files for HPI-B Clients and Daemons" on page 17 for details on available RPM files for your particular node blade.
3. If applicable, enter **`rpm -e <Old HPI-B client package name>`** to uninstall an already installed client package
4. Enter **`rpm -i <New HPI-B client RPM file name>`**
   This installs the HPI-B library package.

The following table lists the directories and their content available on the blade after installing the package.

Table 2-3 Overview of HPI-B Directories and Files on Node Blades

Directory	Description
`/opt/motorola/lib`	Contains example applications and shared libraries needed to run clients

Table 2-3 Overview of HPI-B Directories and Files on Node Blades (continued)

Directory	Description
`/opt/motorola/bin`	Contains precompiled example applications. They are controlled via the command line and can easily be identified through the prefix "hpi" in their names. Use the -h option to display supported command line parameters.
`/opt/motorola/etc/bbs-hpib`	Contains configuration files used to configure HPI client libraries. See for details.

2.3.2 Configuring HPI Clients

Before running your client, you need to configure the multishelf library on the node where the client is to run. The configuration has to be done in the following configuration file: `/opt/motorola/etc/bbs-hpib/bbs-hpibmultishelf.conf`

Most of the entries should be left as they are. They have been set to values that are appropriate for most operations. The only settings that need to be adapted are those which are related to the HPI daemons that the multishelf library wishes to access. The following table shows the expected syntax of the related entries.

Table 2-4 Multishelf Library Configuration File - HPI Daemon Entries

Entry	Description
`[Shelf<Domain Name>]`	This indicates the start of the definition of an HPI daemon. The chosen domain name appears as name of the Shelf Management Resource and is used as Domain tag. See *Shelf Management Resource* on page 32.
`Daemon=<IP address of HPI daemon>`	This is the IP address used to access an HPI daemon.
`Port=<port number>`	This is the port number. The HPI daemon uses 4743 as port.

Typically you will want to specify both HPI daemons in the shelf in the configuration file so that your application can establish a connection with the second HPI daemon in case the connection with the first HPI daemon fails.

The following table lists the IP addresses and ports of the HPI daemons in a Centellis 4620 shelf.

Table 2-5 IP Addresses/Ports of HPI Daemons in a Centellis 4620 Shelf

Location of HPI Daemon	IP Address	Port
Left ATCA-F120 in a shelf	192.168.21.1	4743
Right ATCA-F120 in a shelf	192.168.22.2	4743

Developing Applications

3

3.1 Overview

This chapter describes how to develop applications that make use of the HPI-B library.

Depending on the CPU architecture of the target system where you want to run your HPI-B application and on the operating system, different RPM files are delivered which contain include files and static libraries needed for the application development. The naming scheme used for these files is as follows: `bbs-hpib-devel-<version>-1.<architecture>-<distribution>-<os>.rpm`

In order to run your HPI-B clients, you furthermore need to install the HPI-B client base package applicable to the blade where the client is running. See *Setting Up HPI Clients* on page 19.

The HPI-B client base package contains compiled example applications which illustrate the use of HPI-B controls. For these example applications the source codes and an example make file are available as well. You may want to use the source code and the make file as a starting point for developing your own applications. For further details refer to Appendix A, *Example Applications,* on page 39.

3.2 Building the Application

If your development system is based on the same operating system/CPU architecture environment as the target system, then you can simply install the RPM files on the target system. If the development system is based on another operating system/CPU architecture environment and you consequently intend to do cross-compilation, then the RPM files should be converted to the `cpio` format and then extracted, using the standard Linux `rpm2cpio` tool.

In order to do this, you would for example enter the following at the command prompt:

1. **`cd <working directory>`**
2. **`rpm2cpio <rpm file> | cpio -id`**

After extracting the RPM or CPIO files, you obtain the following directories with the following contents.

Table 3-1 Development RPMs - Directory Structure

Directory	Content
/opt/motorola/`include/bbs-hpib`	Include files
opt/motorola//`lib` or opt/motorola//`lib64`	Static libraries

Using HPI-B

4

4.1 Overview

This chapter provides information which is necessary when writing applications that are based on the Emerson HPI-B distribution. It lists limitations with respect to the HPI-B specification and describes extensions which were added by Emerson.

4.2 Limitations

This section describes those HPI-B features which the Emerson HPI-B implementation for the Centellis 4620 environment does not support.

4.2.1 Limitations with Respect to HPI-B Base Specification

The following limitations apply with respect to the compatibility with the SAI-HPI-B.01.02 specification. Note that these limitations apply to the current and also to all future Emerson HPI-B releases for Centellis 4620 platforms. There are no plans to implement these features in the future.

- Firmware Update Management Instrument (FUMI) functionality not supported
- Diagnostic Management Instrument (DIMI) functionality not supported
- Limited `saHpiIdrAreaAdd()` call
  If the space is available, the function saHpiIdrAreaAdd() adds an OEM Inventory Area including two pre-defined fields as multi-record with a maximum size of 255 Byte. The first one is a read-only field containing the ManufacturerID (3 bytes). The second field (252 bytes) can be updated by the user. The Emerson HPI-B implementation of saHpiIdrAreaAdd() does not support the creation of other types of Inventory Areas.
- Limited saHpiIdrAreaDelete() call
  Emerson HPI-B does not allow deleting Inventory Area with saHpiIdrAreaDelete(), except OEM Multi Records not specified by the PICMG ATCA and AMC specifications.
- Unsupported saHpiParamControl() call
  Emerson HPI-B does not support saHpiParamControl().
- Unsupported resource event log
  Emerson HPI-B does not support resource event logs.
- Unsupported Unicode character set
  Emerson HPI-B does not support the Unicode character set.
- Unsupported annunciator functionality
  Emerson HPI-B does not support annunciator functionality. Our platforms do not have these features.

4.2.2 Limitations with Respect to HPI-B AdvancedTCA Mapping Specification

The following limitations apply with respect to the compatibility with the HPI-B-AdvancedTCA mapping specification SAIM-HPI-B.01.01-ATCA. Note that these limitations apply to the current and also to all future Emerson HPI-B releases. There are no plans to implement these features in the future.

- Only physical slot numbers are supported
  Emerson HPI-B only supports physical slot numbers in entity paths
- Only "shall" and "should" requirements are supported
  Emerson HPI-B only supports the "shall" and "should" requirements of the HPI-to-AdvancedTCA mapping specification SAIM-HPI-B.01.01-ATCA.

4.3 Working with the Multishelf Library

The multishelf HPI library allows your application to connect to and manage several shelves at the same time. For this purpose the multishelf library provides several HPI controls which allow the application to manage the connection to shelves and also to dynamically add and remove shelves to the HPI environment. This section provides all the information that you need to know in order to use these HPI controls and work with the HPI multishelf library.

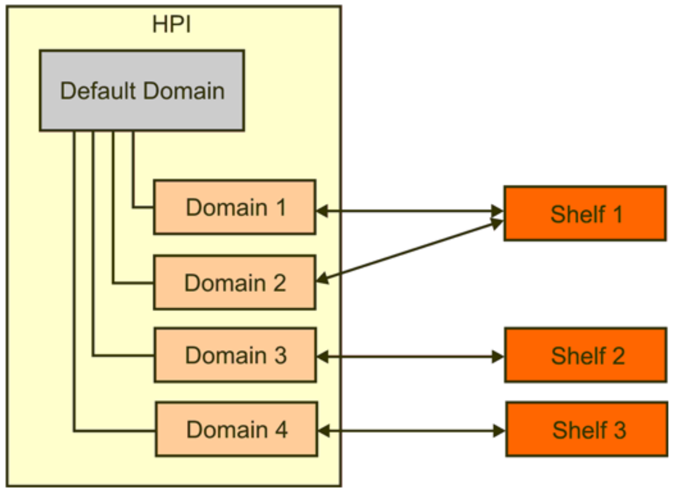
4.3.1 Overview

HPI uses the concept of domains. Generally, a domain represents one shelf. Furthermore there is a default domain. It acts as a container for all other domains and does itself not represent actual hardware.

It is possible for multiple domains to represent the same physical shelf. This is for example the case in typical AdvancedTCA systems which often provide two shelf managers with an HPI daemon running on each of them.

The following figure illustrates an example configuration with four domains and three shelves.

Figure 4-1 Multishelf Library - Representation of Shelves as Domains



Any FRUs available in a shelf are represented as HPI resources together with Resource Data Records (RDRs) corresponding to that FRU. Whenever a FRU is added to or removed from a shelf, the corresponding HPI resource/RDR is added/removed from the HPI domain.

4.3.2 Accessing HPI Domains

In order to access an HPI domain, you must open a session via the HPI call `saHpiSessionOpen()` and provide as first parameter the domain ID of the corresponding HPI domain. How to obtain the domain ID is described later within this section about the multishelf HPI library.

If you want to access several shelves, then you need to open several sessions simultaneously, one session for each HPI domain which represents a shelf. It is also possible to open several sessions for one HPI domain/shelf only as well.

When the connection to a shelf is lost, all running HPI calls which access the corresponding domain return immediately with the error code `SA_HPI_ERR_NO_RESPONSE`. All open sessions for the affected domain are automatically closed by the multishelf library. In the meantime, the library tries to regain access to the shelf. As soon as the connection is reestablished, the domain is recreated and the application can open another session and access the domain again. Whenever a domain is created or removed, an HPI event from the HPI Communication State sensor is generated in the default domain (see *Connection State Sensor* on page 30 and *HPI Domain Events* on page 31).

HPI events are handled domain wide. This means that HPI events from a shelf or FRUs in that shelf are only visible and can only be received within the session that corresponds to that domain.

4.3.3 How Domains and Shelves are Represented

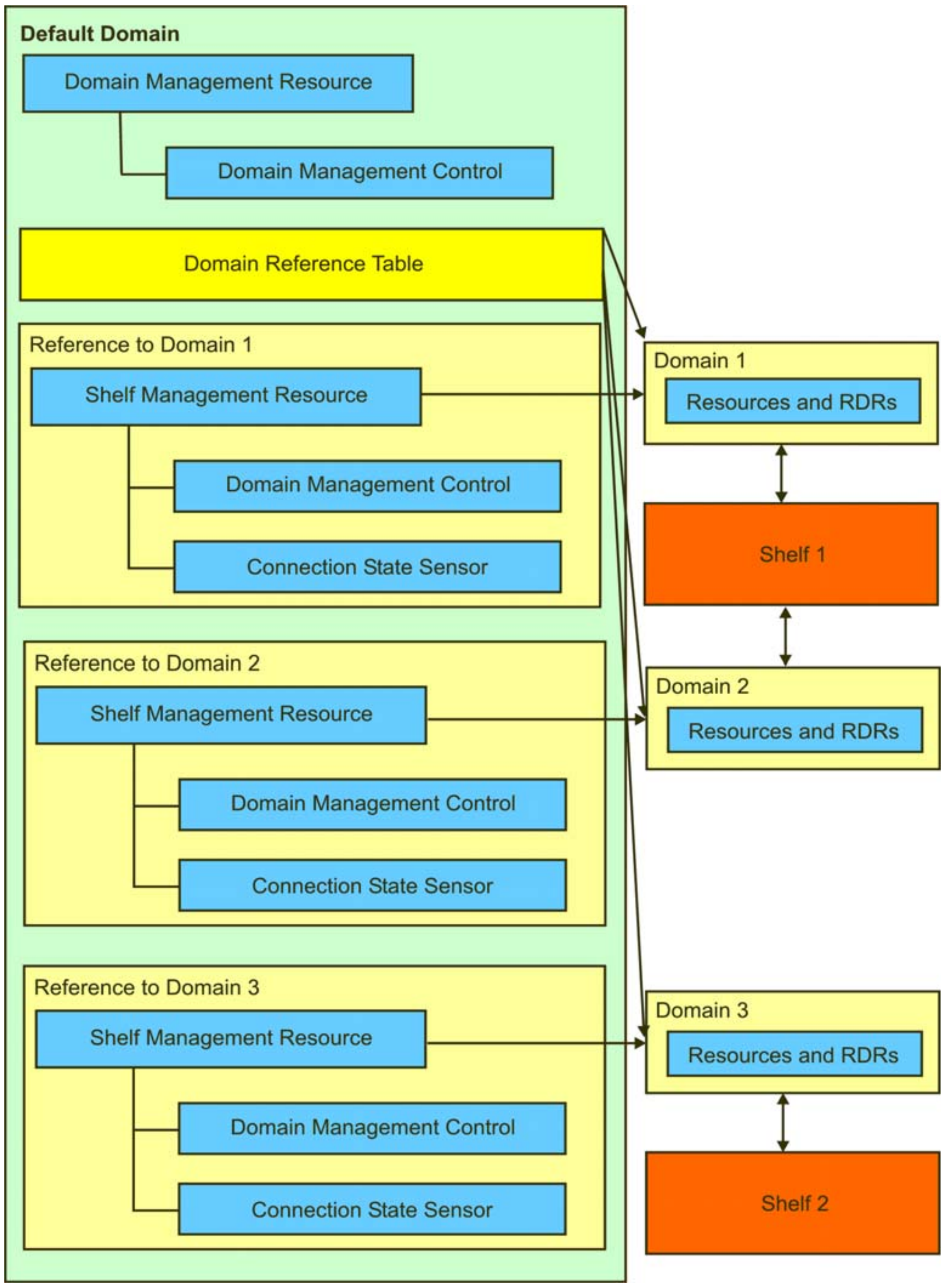
As previously mentioned, each HPI implementation has at least the default domain. It has the ID 0 assigned to it.

Starting with HPI-B, the default domain contains a Domain Reference Table, which contains references to all related domains and may be used by applications for discovery of available domains in the current configuration. For more information about the Domain Reference Table, refer to the HPI-B specification document of the SAI-HPI-B.02.01 standard.

In the Emerson HPI-B implementation, the default domain furthermore contains multiple HPI resources which handle connected shelves and their corresponding HPI domains. These HPI resources were defined and added by Emerson and are called Domain Management Resource and Shelf Management Resource.

The Domain Management Resource contains one HPI control and allows applications to add/remove HPI domains/shelves to the HPI environment. The Shelf Management Resource contains one HPI control and one HPI sensor and acts as reference to connected daemons. There is one Shelf Management Resource for each connected daemon. The following figure shows an example configuration with the HPI resources, controls and sensors which are related to the handling of multiple shelves/domains in it.

Figure 4-2 HPI Multishelf Library - Overview of Related HPI Resources and Controls



In the following, the Domain Management Resource and the Shelf Management resource will be described in detail. A description of typical usage examples/scenarios will be given after that.

4.3.3.1 Domain Management Resource

The Domain Management Resource acts as container for the Domain Management Control and is defined as follows.

Table 4-1 Definition of Domain Management Resource

SaHpiRptEntryT	Value
EntryId	Assigned by HPI
ResourceId	Assigned by HPI
ResourceInfo	0 for all values
ResourceEntity	{RACK,0} this can be changed with the multishelf library configuration file
ResourceCapabilities	SAHPI_CAPABILITY_RESOURCE | SAHPI_CAPABILITY_RDR | SAHPI_CAPABILITY_CONTROL
ResourceSeverity	SAHPI_MAJOR
DomainId	0
IdString	Domain management

This Domain Management control is only writable, not readable, and allows the application to dynamically add and remove domains.

Adding/removing a domain using this HPI control has the same effect as adding/removing a shelf by adding/removing an entry in the multishelf library configuration file. Therefore, whenever you use this HPI control to add/remove a domain, the software automatically updates the configuration file as well.

The RDR and the HPI control are defined as follows.

Table 4-2 Domain Management Control RDR

SaHpiRdrT	Value
RecordId	Assigned by HPI
RdrType	SAHPI_CTRL_RDR
Entity	The same entity like Domain Management Resource
RdrTypeUnion	Define in Table 4-3.
IdString	MOTHPI_CTRL_NAME_DOMAIN_MANAGEMENT

Table 4-3 Domain Management Control

SaHpiCtrlRecT	Value
Num	MOTHPI_CTRL_NUM_DOMAIN_MANAGEMENT
Ignore	SAHPI_FALSE
OutputType	SAHPI_CTRL_OEM
Type	SAHPI_CTRL_TYPE_OEM
TypeUnion - Oem -MId	MOTHPI_MANUFACTURER_ID_MOTOROLA
TypeUnion - Oem -ConfigData	0
TypeUnion - Oem - Default - MId	0
TypeUnion - Oem - Default - BodyLength	0
TypeUnion - Oem - Default - Body	0
Oem	0

Table 4-4 Domain Management Control State

SaHpiCtrlStateT	Value
Type	SAHPI_CTRL_TYPE_OEM
StateUnion - Oem - MId	MOTHPI_MANUFACTURER_ID_MOTOROLA
StateUnion - Oem - BodyLength	Depends on the length of the resource name
StateUnion - Oem - Body	Sequence of n bytes, named [0] ... [n], with the following definitions: [0] - Command 0 = Get state 1 = Create domain (for set state) 2 = Remove domain (for set state) [1] - [4] - IP address (little endian order) [5][6] - port (little endian order) [7] ... [n] - Domain name as null-terminated string.

4.3.3.2 Shelf Management Resource

The shelf management resource represents one daemon. For each configured daemon, the default domain creates one shelf management resource. The shelf management resource is defined as follows.

Table 4-5 Shelf Management Resource

SaHPIRptEntryT	Value
EntryId	Assigned by HPI
ResourceId	Assigned by HPI
resourceInfo	0 for all values
ResourceEntity	{SYS_MNGMT_SOFTWARE domain Id}

Table 4-5 Shelf Management Resource (continued)

SaHPIRptEntryT	Value
ResourceCapabilities	SAHPI_CAPABILITY_RESOURCE | SAHPI_CAPABILITY_RDR | SAHPI_CAPABILITY_CONTROL | SAHPI_CAPABILITY_SENSOR When the shelf is reachable: SAHPI_CAPABILITY_DOMAIN
ResourceSeverity	SAHPI_MAJOR
DomainId	Domain Id when the shelf is reachable
IdString	Domain name

The shelf management resource contains the following two RDRs:

- Domain management control
  Read-only HPI control which allows the application to obtain the IP address, port, domain name and other connection parameters of a particular shelf
- Connection status sensor
  Contains information about the current status of the connection to a shelf

The definitions of both HPI controls are given in the following.

4.3.3.2.1 Domain Management Control

This HPI control is only readable. It has the same structure and definition as the HPI controls defined in Table "Domain Management Control" on page 29, Table "Domain Management Control State" on page 29 and Table "Domain Management Control RDR" on page 28.

4.3.3.2.2 Connection State Sensor

This HPI sensor represents the status of the connection to a shelf. When the connection is interrupted, an HPI event of type SAHPI_ES_OFF_LINE is generated. When the connection is reestablished, an HPI event of type SAHPI_ES_ON_LINE is generated. An application can access a shelf only when the state of this sensor is SAHPI_ES_ON_LINE. The definition of this sensor is given in the following tables.

Table 4-6 Connection State RDR

SaHpiRdrT	Value
RecordId	Assigned by HPI
RdrType	SAHPI_SENSOR_RDR
Entity	The same entity as domain management resource
RdrTypeUnion	MOTHPI_SENSOR_NAME_DOMAIN_CONNEC TION

Table 4-7 Domain Connection Sensor

SaHpiSensorRecT	Value
Num	MOTHPI_SENSOR_NUM_DOMAIN_CONNECTION
Type	SAHPI_CHASSIS
Category	SAHPI_EC_AVAILABILITY
EventCtrl	SAHPI_SEC_PER_EVENT
Events	SAHPI_ES_ON_LINE | SAHPI_ES_OFF_LINE
Ignore	SAHPI_FALSE
DataFormat - ReadingFormats	SAHPI_SRF_EVENT_STATE
DataFormat - IsNumeric	SAHPI_FALSE
DataFormat - IsThreshold	SAHPI_FALSE
Oem	0

Table 4-8 Domain Connection Sensor Reading

SaHpiSensorReadingT	Value
ValuesPresent	SAHPI_SRF_EVENT_STATE
EventStatus - SensorStatus	SAHPI_SENSTAT_EVENTS_ENABLED | SAHPI_SENSTAT_SCAN_ENABLED
AssertEvents	SAHPI_ES_ON_LINE for a connection to a shelf SAHPI_ES_OFF_LINE when there is no connection to a shelf

4.3.3.3 HPI Domain Events

Depending on the connection status, the Domain Connection sensor can throw events. The following table provides details.

Table 4-9 Domain Connection Sensor Events

Event	Description
SAHPI_ES_OFF_LINE	No connection
SAHPI_ES_ON_LINE	Connection is established. Only now is the domain of the shelf accessible.

4.3.4 Discovering Shelves and Domains

As previously mentioned, one way to discover available domains is to use the Domain Reference Table. For further details, refer to the SAI-HPI-B.01.02 specification document.

Alternatively, you can use the Shelf Management Resources defined in the default domain. Any shelves that were added to the HPI environment are represented by one Shelf Management Resource with the entry ResourceCapability set to SAHPI_CAPABILITY_DOMAIN. So in order to discover shelves/domains, you simply need to parse the default domain for HPI resources with the entry ResourceCapability set to SAHPI_CAPABILITY_DOMAIN. Then you can obtain the respective domain ID by reading the entry DomainId and open a session to the shelf using the obtained domain ID.

4.3.5 Adding and Removing Shelves and Domains

All shelves which are used by the HPI multishelf library are listed in the multishelf library configuration file. There are two ways of adding and removing shelves:

- Manually by editing the multishelf configuration file. Changes become effective after the next restart in this case.
- Dynamically by using the HPI domain management control

For a description of how to manually edit the multishelf library configuration file, refer to *Configuring HPI Clients* on page 20.

A description of how to dynamically add and remove shelves, is given in the following two subsections.

4.3.5.1 Adding Shelves and Domains

In order to dynamically add a shelf and create a domain, you must invoke the HPI function `saHpiControlStateSet()` and provide the following parameters.

Parameter	Value
SessionId	Handle to session context.
ResourceId	Resource ID of the addressed resource.
CtrlNum	Number of the control for which the state is being set.
CtrlState	Pointer to control state as described in Table "Domain Management Control State" on page 29. In this control state, fill the field body as follows:, [0] : 1 (stands for "create domain") [1] - [4] : IP address of shelf you want to add (little endian order) [5] - [6] : port of shelf (little endian order) [7] - [n] : Domain name

After calling saHpiControlStateSet(), a new resource with the domain name you supplied as parameter is created. Additionally, a hot swap event of type ACTIVE for the created resource is sent. The newly created shelf is also added to the HPI multishelf configuration file as a new entry.

All shelves must have different IP addresses.

4.3.5.2 Removing Shelves and Domains

In order to dynamically remove a shelf/domain, you must invoke the HPI function `saHpiControlStateSet()` and provide the following parameters.

Parameter	Value
SessionId	Handle to session context.
ResourceId	Resource ID of the addressed resource.
CtrlNum	Number of the control for which the state is being set.
CtrlState	Pointer to control state as described in Table "Domain Management Control State" on page 29. In this control state, fill the field body as follows:, [0] : 2 (stands for "remove domain") [1] - [4] : IP address of shelf you want to add (little endian order) [5] - [6] : port of shelf (little endian order) [7] - [n] : Domain name

If the domain/shelf specified in the call exists, the domain/shelf is removed, an HPI domain removed event is generated and any sessions that may be open to this domain are closed. Furthermore the domain shelf resource in the HPI domain is removed and a hot swap even of type "NOT_PRESENT" is generated.

4.4 Emerson Extensions

The following describes features which are not specified in the HPI-B specifications, but were added by Emerson.

4.4.1 HPI Controls for Domain and Shelf Management

This refers to the Domain Management Resource and Shelf Management Resource which were previously explained. Both are Emerson-specific extensions. See *Working with the Multishelf Library* on page 24.

4.4.2 IPMI System Boot Options Support

A FRU may have a payload which is capable of booting an operating system (OS). Usually, a boot firmware, such as BIOS or U-Boot, is started after the payload is powered up or reset. Via the System Boot Options Control you can set some options for the boot firmware. The boot firmware will read these settings from the IPMC.

In order to set or get the system boot options, you need to use the HPI Boot Option control.

This control maps the IPMI commands `Set System Boot Options` and `Get System Boot Options` to HPI.

The Boot Option Control is only available for AdvancedTCA front blades and the shelf manager if the respective IPMC supports the `Set System Boot Option` IPMI command. Refer to the Intelligent Platform Management Interface Specification v2.0, section 28.12 Set System Boot Options Command and 28.13 Get System Boot Options Command, for further details. Furthermore refer to the respective IPMI Programmer's Reference manuals of the respective blades.

Table 4-10 Boot Option RDR

SaHpiRdrT	Value
RecordId	Assigned by HPI
RdrType	SAHPI_CTRL_RDR
Entity	The same entity as resource
RdrTypeUnion	Defined in next table
IdString	MOTHPI_CTRL_NAME_BOOT_OPTION

Table 4-11 Boot Option Control

SaHpiCtrlRecT	Value
Num	MOTHPI_CTRL_NUM_BOOT_OPTION
Ignore	SAHPI_FALSE
OutputType	SAHPI_CTRL_OEM
Type	SAHPI_CTRL_TYPE_OEM
TypeUnion.Oem.Mid	MOTHPI_MANUFACTURER_ID_MOTOROLA
Oem	0

Table 4-12 Boot Option State

SaHpiCtrlStateT	Value
Type	SAHPI_CTRL_TYPE_OEM
StateUnion.Oem.Mid	MOTHPI_MANUFACTURER_ID_MOTOROLA
StateUnion.Oem.BodyLength	**Get operations:** When input parameter: 3 When output parameter: Total length of the response data - 2 **Set operations:** Total length of the request data
StateUnion.Oem.Body	**Get operations:** When input parameter: Byte 0: Parameter selector Byte 1: Set selector Byte 2: Block selector When output parameter: Response data from the GetSystemBootOptions IPMI command without the first two bytes **Set operations:** Request data for the IPMI SetSystemBootOptionss command

Example: on most Emerson blades, the BIOS software is stored twice on a flash device, BIOS bank 1 and BIOS bank 2. The bank to boot from can be selected with the System Boot Options Control.

You have to select parameter 96 with the Boot Option Select Control and then set the BIOS 1 (0) or BIOS 2 (1) with the Boot Option Control.

The following example shows how to set BIOS 2 for a resource with the ID 120 using an example program which is delivered together with the HP-B development package:

```
hpibootoptions -r 120 96 1
```

The example program hpibootoptions, which is available in the base RPM package, shows how an option can be set or got.

4.4.3 POST Type Control

This HPI control allows you to set/get the Power-On Self Test (POST) type of the blade. Two POST types are configurable: long POST and short POST. Refer to the respective hardware user manual of the blade for details about both POST types. The definition of the respective HPI controls is given in the following tables.

Table 4-13 POST Type HPI Control RDR

SaHpiRdrT	Value	Notes
RecordId	Assigned by HPI implementation	Unique identifier for the RDR
RdrType	SAHPI_CTRL_RDR	
RdrTypeUnion.CtrlRec.Num	MOTHPI_CTRL_NUM_POSTTYPE	
RdrTypeUnion.CtrlRec.OutputType	SAHPI_CTRL_OEM	Indicates that this control does not correlate to any of the given control output types, and describes a generic control output.
RdrTypeUnion.CtrlRec.Type	SAHPI_CTRL_TYPE_DISCRETE	
RdrTypeUnion.CtrlRec.TyoeUnion.Discrete.Default	0x0	
RdrTypeUnion.CtrlRec.Oem	Assigned by HPI implementation	
IdString	MOTHPI_CTRL_NAME_POSTTYPE	

Table 4-14 POST Type HPI Control Sate Values

SaGPISateT	Value	Notes
Type	SAHPI_CTRL_TYPE_DISCRETE	
StateUnion.Discrete	Bits 7..0: 0: Short POST 1: Long POST Bits 15..8: CPU complex number Bits 31..16: Reserved	The POST type value is specified in the least significant byte. The CPU number (or SET Selector byte) is normally zero. However, for blades that support more than one CPU complex, the processor complex is identified with this field.

4.4.4 HPI Logging Support

The actions taken by the HPI daemon are written to a log file. The path and name of the log file is defined in the configuration file `bbs-hpib.conf`.

To control what kind of information is written to the log file, the log control can be used.

Refer to the header file `MotorolaHpi.h` for used defines.

Table 4-15 Log RDR

SaHpiRdrT	Value
RecordId	Assigned by HPI
RdrType	SAHPI_CTRL_RDR
Entity	The same entity as logical shelf resource
RdrTypeUnion	Defined in next table
IdString	MOTHPI_CTRL_NAME_LOG found in MotorolaHpi.h

Table 4-16 Log Control

SaHpiCtrlRecT	Value
Num	MOTHPI_CTRL_NUM_LOG
Ignore	SAHPI_FALSE
OutputType	SAHPI_CTRL_OEM
Type	SAHPI_CTRL_TYPE_OEM
TypeUnion.Oem.MId	MOTHPI_MANUFACTURER_ID_MOTOROLA (2x)
TypeUnion.Oem.ConfigData	
TypeUnion.Oem.Default.MId	MOTHPI_MANUFACTURER_ID_MOTOROLA (2x)
TypeUnion.Oem.Default.BodyLength	0
TypeUnion.Oem.Default.Body	0
Oem	0

Table 4-17 Log Control State

SaHpiCtrlStateT	Value
Type	SAHPI_CTRL_TYPE_OEM
StateUnion.Oem.MId	MOTHPI_MANUFACTURER_ID_MOTOROLA
StateUnion.Oem.BodyLength	40 + length of log file name
StateUnion.Oem.Body	`MOTHPI_LOG_ERROR_FACILITIES_OFFSET` - facilities for that error logging is enabled `MOTHPI_LOG_WARNING_FACILITIES_OFFSET` - facilities for that warning logging is enabled `MOTHPI_LOG_INFO_FACILITIES_OFFSET` - facilities for that info logging is enabled `MOTHPI_LOG_DEBUG_FACILITIES_OFFSET` - facilities for that debug logging is enabled The logging facilities are defined in the file log_utils.h, enum oh_log_fac. `MOTHPI_LOG_CURRENT_IDX_OFFSET` - index of currently used log file `[MOTHPI_LOG_OFFSET_PROPERTIES]` - output locationdLogStdout - stdoutdLogStderr - stderrdLogFile - log file `[MOTHPI_LOG_NUM_FILES_OFFSET]` - number of logfile created `[MOTHPI_LOG_MAX_FILE_SIZE_OFFSET]` - maximum logfile size before creating a new one. This is a 32 bit field in MSB byte order. Use GetUInt32 in byte_utils.h to get host byte order. `[MOTHPI_LOG_FILENAME_OFFSET]` - log file name.

Example Applications

A

A.1 Overview

The HPI-B client base package contains precompiled example applications. They are invoked via the command line and can be configured via command line parameters. Each example application illustrates a certain feature of HPI-B and makes use of the respective HPI-B function calls.

After extracting the HPI-B client RPMs, the example applications can be found in the following directory: `/opt/motorola/bin`. You can obtain information about the command usage by invoking the application from the command line and providing `-h` as parameter.

The HPI-B example programs are provided "as is" without any warranty of any kind, either express or implied. The entire risk as to the quality, operability and execution of the programs is with you. Should the programs prove to be faulty or incorrect, you assume the cost of all necessary servicing, repair or correction. In no event Emerson will be liable to you for any damages, any lost profits or other special, incidental or consequential damages arising out of the use or inability to use the programs.

Emerson reserves the right to revise or remove the programs in subsequent releases without obligation of Emerson to notify any person of such revision or changes.

A.2 Example Application Source Files

In order to ease application development and help you to get familiar with the HPI-B API usage, Emerson provides the source files of the example applications and an example make file. These files are available as different RPM files, depending on the operating system and CPU architecture. The naming scheme used for the RPMs is: `bbs-hpib-clientsrc-<version>-1.<CPU architecture>-<distribution>-<os>.rpm`

After installing the RPMs, the source files and the example make file are located in the following directory: `/opt/motorola/src/bbs-hpib/clients`.

A.3 List of Supported Example Applications

The following is an automatically generated output (based on the files contained in /opt/motorola/bin and the output obtained via the -h parameter) that describes all HPI example applications which were available when this manual was written and the usage of these applications. Note that in the meantime further example applications may have been added or the functionality of existing HPI example applications may have been changed slightly. The current usage and fucntionality can always be obtained by invoking the example application with the -h parameter.

```
#
# Lists the usage of all supported HPI-B example programs.
#
# Copyright (c) 2007 by Motorola GmbH
# Copyright (c) 2008, Emerson Network Power - Embedded Computing GmbH
#

#------------------------------#
# hpiautotimer
#------------------------------#

Usage: hpiautotimer [OPTION]...
HPI example application to manage the timeout values of the auto insert timer
and auto extract timer.

Options:
  -d DOMAIN_ID        use domain with id DOMAIN_ID
  -D                  walk recursivly through DRT
  -r RESOURCE_ID      use resource with id RESOURCE_ID
  -i INSERT_TIMEOUT   set auto insert timeout value in msec to INSERT_TIMEOUT
  -e EXTRACT_TIMEOUT  set auto extract timeout value in msec to EXTRACT_TIMEOUT
  -V                  print version information and exit
  -h                  display this help and exit

#------------------------------#
# hpibootbanks
#------------------------------#

Usage: hpibootbanks [OPTION]...
HPI example application to switch the boot bank of Motorola/Emerson specific
dual-flash-bank boards.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -b BANK_NUM     set boot bank number to BANK_NUM [0/1]
  -V              print version information and exit
  -h              display this help and exit

#------------------------------#
# hpibootoptions
#------------------------------#

Usage: hpibootoptions [OPTION]... [PARAM# PARAMS]
HPI example application to manage the system boot options defined in IPMI v2.0 .

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -c              turn on console redirection
                  (Not applicable on some boards)
  -o              turn off console redirection
                  (Not applicable on some boards)
  -x              perform FRU cold-reset
  -V              print version information and exit
  -h              display this help and exit

Examples:
```

```
Set the system boot option 1 (service partition) to value 2 for resource 4 on
domain 0:
  hpibootoptions -d 0 -r 4 1 2
Turn on console redirection and perform a cold-reset for resource 34 on domain
0:
  hpibootoptions -d 0 -r 34 -c -x

#------------------------------#
# hpichassisstatus
#------------------------------#

Usage: hpichassisstatus [OPTION]...
HPI example application to display the control state of the chassis status
control.

Options:
  -d DOMAIN_ID        use domain with id DOMAIN_ID
  -D                  walk recursivly through DRT
  -r RESOURCE_ID      use resource with id RESOURCE_ID
  -V                  print version information and exit
  -h                  display this help and exit

#------------------------------#
# hpicooling
#------------------------------#

Usage: hpicooling [OPTION]...
HPI example application to control the cooling mode.
Note: Only applicable if the HPI daemon runs in Shelf Manager mode.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -D              walk recursivly through DRT
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -m mode         set cooling mode [1=AUTO, 2=MANUAL]
  -V              print version information and exit
  -h              display this help and exit

#------------------------------#
# hpidomain
#------------------------------#

Usage: hpidomain [OPTION]...
HPI example application to display all domains found.

Options:
  -V   print version information and exit
  -h   display this help and exit

#------------------------------#
# hpidomainel
#------------------------------#

Usage: hpidomainel [OPTION]...
HPI example application to display the domain event log.

Options:
  -d DOMAIN_ID  use domain with id DOMAIN_ID
  -c            clear the event log
  -A            display everything
  -t            display RDR with the event log
  -p            display RPT with the event log
  -x            display debug messages
  -V            print version information and exit
  -h            display this help and exit

#-------------------------------#
# hpidomainself
#-------------------------------#

Usage: hpidomainself [OPTION]...
HPI example application to print the domain ID where this program is running on.
Note: Requires multishelf library.

Options:
  -V   print version information and exit
  -h   display this help and exit

#-------------------------------#
# hpifailedextract
#-------------------------------#

Usage: hpifailedextract [OPTION]...
HPI example to remove a failed resource using the failed extract control.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -r RESOURCE_ID  use resource with id RESOURCE_ID to extract
  -a              display all resources (default: list only failed resources)
  -V              print version information and exit
  -h              display this help and exit

#-------------------------------#
# hpifan
#-------------------------------#

Usage: hpifan [OPTION]...
HPI example application to control the cooling mode.
Note: Only applicable if the HPI daemon runs in Shelf Manager mode.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -D              walk recursivly through DRT
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -s level        set cooling level (only in MANUAL mode)
  -V              print version information and exit
  -h              display this help and exit

#-------------------------------#
# hpifruactivation
#-------------------------------#

Usage: hpifruactivation [OPTION]...
HPI example application to manage the FRU activation mode.

Options:
  -d DOMAIN_ID     use domain with id DOMAIN_ID
  -D               walk recursivly through DRT
  -r RESOURCE_ID   use resource with id RESOURCE_ID
  -a {0|1}         disable/enable ShM activation
  -w DELAY         set delay before next power on to DELAY * 1/10 sec
  -V               print version information and exit
  -h               display this help and exit

#-------------------------------#
# hpifruipmcreset
#-------------------------------#

Usage: hpifruipmcreset [OPTION]...
HPI example application to set FRU IPMC Reset Control actions.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -c              IPMC cold reset
  -w              IPMC warm reset
  -V              print version information and exit
  -h              display this help and exit

#-------------------------------#
# hpifruresetdiag
#-------------------------------#

Usage: hpifruresetdiag [OPTION]...
HPI example application to set FRU Reset and Diagnostic Control actions.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -s VALUE        set control value VALUE
                  [1=Graceful Reboot, 2=Diagnostic Interrupt]
  -V              print version information and exit
  -h              display this help and exit

#-------------------------------#
# hpifumi
#-------------------------------#

Usage: hpifumi [OPTION]...
HPI example application to manage the firmware upgrade of FRUs using FUMI.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -f FUMI_NUM     use fumi with number FUMI_NUM
  -b BANK_NUM     use bank with number BANK_NUM
  -s URI          calls saHpiFumiSourceSet URI
  -u              calls saHpiFumiInstallStart
  -v              calls saHpiFumiSourceInfoValidateStart
  -i              calls saHpiFumiSourceInfoGet
  -t              calls saHpiFumiTargetInfoGet
  -a              calls saHpiFumiActivate
  -g              calls saHpiFumiUpgradeStatusGet
  -y              calls saHpiFumiTargetVerifyStart
  -c              calls saHpiFumiUpgradeCancel
  -V              print version information and exit
  -h              display this help and exit
```

```
#------------------------------#
# hpiha
#------------------------------#

Usage: hpiha [OPTION]...
HPI example application to show the HA state and to initiate a switch-over.

Options:
  -d <domain ID>  use domain with ID <domain ID>
  -m              initiate a Shelf Manager switch-over
  -s              initiate a HPI Daemon switch-over
  -V              print version information and exit
  -h              display this help and exit


#------------------------------#
# hpihotswap
#------------------------------#

Usage: hpihotswap [OPTION]...
HPI example application to invoke hotswap actions.
Note: If calling without options the application switches to interactive mode.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -s              show hotswap resources
  -a RESOURCE_ID  activate resource
  -t RESOURCE_ID  deactivate resource
  -i RESOURCE_ID  insert resource
  -e RESOURCE_ID  extract resource
  -p RESOURCE_ID  get powerstate of resource
  -u RESOURCE_ID  power up resource
  -o RESOURCE_ID  power down resource
  -c RESOURCE_ID  cold-reset resource
  -V              print version information and exit
  -h              display this help and exit

#------------------------------#
# hpiidh
#------------------------------#

Usage: hpiidh [OPTION]...
HPI example application to manage inventory data.

Options:
  -d DOMAIN_ID                    use domain with id DOMAIN_ID
  -e ENTITY_PATH                  use only RDRs with entity path ENTITY_PATH
  -p                              display all inventory data
  -z                              display OEM areas in hex format
  -m "AREA_ID FIELD_ID STRING"    write string STRING to OEM field specified
  -b "FILE AREA_ID FIELD_ID"      write content of file FILE to OEM field
specified
  -y "FILE AREA_ID FIELD_ID"      write OEM field specified to file FILE
  -a STRING                    create new OEM area and field and write string
STRING to it
  -l AREA_ID                      delete OEM area
  -u                              display User Info Areas
  -x                              display User Info Areas in hex format
  -s STRING                         write string STRING to User Info Area
  -f FILE                          write content of file FILE to User Info Area
  -w FILE                           write User Info Area to file FILE
  -V                                print version information and exit
  -h                                display this help and exit

#-------------------------------#
# hpiipmb0
#-------------------------------#

Usage: hpiipmb0 [OPTION]...
HPI example application to set the IPMB-A or IPMB-B state control.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -D              walk recursivly through the DRT
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -a {0|1}        isolate/join IPMB-A
  -b {0|1}        isolate/join IPMB-B
  -l LINK_NUM     use link number LINK_NUM when isolating the IPMB
  -V              print version information and exit
  -h              print this help and exit

#-------------------------------#
# hpiipmi
#-------------------------------#

Usage: hpiipmi [OPTION]... LUN NETFN CMD [DATA]
HPI example application to send native IPMI commands using the Motorola/Emerson
specific IPMI control.
LUN, NETFN, CMD and DATA will be interpreted as hexadecimal values

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -V              print version information and exit
  -h              display this help and exit

Example:
Send GetDeviceId command (Lun=0, Netfn=6, Cmd=1) to resource 43 on domain 1:
  hpiipmi -d 1 -r 43 0 6 1

#-------------------------------#
# hpiled
#-------------------------------#

Usage: hpiled [OPTION]...
HPI example application to control the LEDs.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -n CONTROL_ID   use control with id CONTROL_ID for set operations
  -a              set control mode to auto
  -1 RATE         set led on duration rate in 1/100sec [0-255]
  -0 RATE         set led off duration rate in 1/100sec [0-255]
  -t RATE         led test
  -V              print version information and exit
  -h              display this help and exit
```

```
#------------------------------#
# hpilink
#------------------------------#

Usage: hpilink [OPTION]...
HPI example application to display E-Keying link states and optionally listen
for link state events.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -e              listen for E-Keying link state events
  -V              print version information and exit
  -h              display this help and exit

#------------------------------#
# hpilist
#------------------------------#

Usage: hpilist [OPTION]...
HPI example application to list all RPT/RDR entries and to get all events.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -D              walk recursivly through DRT
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -s              output short information (don't display RDR data)
  -e              listen for events
  -n              no discover (useful for just getting events)
  -i              interactive mode (for development use only)
  -V              print version information and exit
  -h              display this help and exit

#------------------------------#
# hpilog
#------------------------------#

Usage: hpilog [OPTION]...
HPI example application to control the logger.

Options:
  -d <domain ID>  use domain with ID <domain ID>
  -c <ctrl num>   use control with number <cntrl num>
  -p <properties> set log properties to <properties>
  -q <properties> reset log properties <properties>
  -r <severities> <facilities>
                  reset log <severities> for <facilities>
  -s <severities> <facilities>
                  set log <severities> for <facilities>
  -V              print version information and exit
  -h              display this help and exit

Supported properties: stdout|stderr|file|simplefile|syslog|prefix|threadid|
Supported severities: debug|info|warning|error|all
Supported facilities:
other|connection|transport|session|plugin|remote|daemon|client|core|ha|hpica
ll|resource|sensor|control|inventory|watchdog|dimi|fumi|hotswap|sel|ipmi|ipm
icon|ipmidump|ipmimcthread|ipmidiscover|ipmimc|ipmisdr|ipmiservice|ipmicooli
```

```
ng|ipmipower|ipmiptpekeying|ipmisub|softwareupgrade|redundancy|script|shfruv
alidation|sdrrepository|rmcp|deassert|solmgmt|marshal|all


#-------------------------------#
# hpiposttype
#-------------------------------#

Usage: hpiposttype [OPTION]...
HPI example application to control POST type.

Options:
  -d DOMAIN_ID      use domain with id DOMAIN_ID
  -D                walk recursivly through DRT
  -r RESOURCE_ID    use resource with id RESOURCE_ID
  -t POSTTYPE       set POST Type value
  -c CPU_NUM        set CPU number [default=0]
  -V                print version information and exit
  -h                display this help and exit

#-------------------------------#
# hpipoweronsequence
#-------------------------------#

Usage: hpipoweronsequence [OPTION]...
HPI example application to manage the power on sequence of FRUs during initial
startup.
Options:
  -d DOMAIN_ID     use domain with id DOMAIN_ID
  -D               walk recursivly through DRT
  -r RESOURCE_ID   use resource with id RESOURCE_ID
  -p POSITION      use POSITION as power on position
  -s SLOT_RES_ID   set SLOT_RES_ID for specific position
                   (Requires '-r' and '-p' option)
  -c               commit power on sequence to FRU Info
  -V               print version information and exit
  -h               display this help and exit

#-------------------------------#
# hpireset
#-------------------------------#

Usage: hpireset [OPTION]...
HPI example application to reset a FRU resource.

Options:
  -d DOMAIN_ID     use domain with id DOMAIN_ID
  -r RESOURCE_ID   use resource with id RESOURCE_ID
  -c               cold-reset resource (Requires '-r' option)
  -w               warm-reset resource (Requires '-r' option)
  -V               print version information and exit
  -h               display this help and exit

#-------------------------------#
# hpiresourceself
#-------------------------------#

Usage: hpiresourceself [OPTION]...
HPI example application to print the resource ID where this program is running
```

```
on.

Options:
  -d DOMAIN_ID  use domain with id DOMAIN_ID
  -V            print version information and exit
  -h            display this help and exit

#-------------------------------#
# hpirestartdaemon
#-------------------------------#

Usage: hpirestartdaemon [OPTION]...
HPI example application to restart the HPI daemon.

Options:
  -d DOMAIN_ID  use domain with id DOMAIN_ID
  -V            print version information and exit
  -h            display this help and exit

#-------------------------------#
# hpishaddr
#-------------------------------#

Usage: hpishaddr [OPTION]...
HPI example application to display and set the shelf address.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -b HEX_STRING   set shelf address using binary hex string HEX_STRING
  -V              print version information and exit
  -h              display this help and exit

#-------------------------------#
# hpishelf
#-------------------------------#

Usage: hpishelf [OPTION]...
HPI example application to manage connections to domains.
Note: Requires multishelf library.

Options:
  -c DOMAIN_NAME  create domain with name DOMAIN_NAME
  -l DOMAIN_NAME  delete domain with name DOMAIN_NAME
  -i IP_ADDR      use IP address IP_ADDR to connect to domain
  -p PORT         use port PORT to connect to domain
  -e              listen for events
  -V              print version information and exit
  -h              display this help and exit

Examples:
Add domain "Gandalf" with IP address 192.168.111.86:
  hpishelf -c Gandalf -i 192.168.111.86
Delete domain "Gandalf":
  hpishelf -l Gandalf

#-------------------------------#
# hpiship
#-------------------------------#
```

```
Usage: hpiship [OPTION]...
HPI example application to display and set the Shelf Manager IP address.

Options:
  -d DOMAIN_ID  use domain with id DOMAIN_ID
  -n CTRL_NUM   use control with number CTRL_NUM
  -i IP_ADDR    set IP address IP_ADDR
  -m NETMASK    set netmask NETMASK
  -g GW_ADDR    set default gateway address GW_ADDR
  -V            print version information and exit
  -h            display this help and exit

#------------------------------#
# hpisol
#------------------------------#

Usage: hpisol [OPTION]... [PARAM# PARAMS]
HPI example application to manage IPMI v2.0 SOL (Serial over LAN) settings.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -V              print version information and exit
  -h              display this help and exit

Examples:
Show SOL configuration for domain 1:
  hpisol -d 1
Set SOL param 1 (SOL Enable) to value 1 for resource 4 on domain 1:
  hpisol -d 1 -r 4 1 1

#------------------------------#
# hpitelcoalarm
#------------------------------#

Usage: hpitelcoalarm [OPTION]...
HPI example application to control telco alarms.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -i VALUE        set minor alarm value
  -a VALUE        set major alarm value
  -c VALUE        set critical alarm value
  -V              print version information and exit
  -h              display this help and exit

#------------------------------#
# hpitop
#------------------------------#

Usage: hpitop [OPTION]...
HPI example application to display system topology.

Options:
  -d DOMAIN_ID    use domain with id DOMAIN_ID
  -r RESOURCE_ID  use resource with id RESOURCE_ID
  -A              display everything
  -p              display RPTs
  -s              display sensors
  -c              display controls
  -w              display watchdogs
  -i              display inventories
  -a              display annunciators
  -x              display debug messages
  -V              print version information and exit
  -h              display this help and exit

#------------------------------#
# hpiversion
#------------------------------#

Usage: hpiversion [OPTION]...
HPI example application to display the version of the different HPI components.
Options:
  -d DOMAIN_ID  use domain with id DOMAIN_ID
  -i ITEM       print one of the following version item:
  -V            print version information and exit
  -h            display this help and exit
                HPI
                HPI-ATCA-MAPPING
                CLIENT
                CLIENT_PROTOCOL
                MULTISHELF
                MULTISHELF-PROTOCOL
                DAEMON
                DAEMON-PROTOCOL
                DAEMON-HA-PROTOCOL
```

Related Documentation

B

B.1 Emerson Network Power - Embedded Computing Documents

The Emerson Network Power - Embedded Computing publications listed below are referenced in this manual. You can obtain electronic copies of Emerson Network Power - Embedded Computing publications by contacting your local Emerson sales office. For documentation of final released (GA) products, you can also visit the following website: www.emersonnetworkpower.com/embeddedcomputing > Resource Center > Technical Documentation Search. This site provides the most up-to-date copies of Emerson Network Power - Embedded Computing product documentation.

Table B-1 Emerson Network Power - Embedded Computing Publications

Document Title and Source	Publication Number
Centellis 4620 Release Document Collection	6806800G48

B.2 Related Specifications

For additional information, refer to the following table for related specifications. As an additional help, a source for the listed document is provided. Please note that, while these sources have been verified, the information is subject to change without notice.

Table B-2 Related Specifications

Organization	Document Title
Intel http://www.developer.intel.com/design/servers/ipm	Platform Management FRU Information Storage Definition v1.0 IPMI Specification v2.0
PICMG http://www.picmg.org/v2internal/specifications.htm	PICMG 3.0 Revision 2.0 Advanced TCA Base Specification
Service Availability Forum http://www.saforum.org	SAI-HPI-B02.01 Service Availability Forum Hardware Platform Interface specification SAI-HPI-B01.01.01-ATCA Service Availability Forum HPI-to-AdvancedTCA Mapping specification