InfiniBand Administration Package

User's Manual

Rev 1.20

InfiniBand Administration Package User's Manual

Document Number: 2130UM

Mellanox Technologies, Inc.
2900 Stender Way
Santa Clara, CA 95054
U.S.A.
www.Mellanox.com

Tel: (408) 970-3400
Fax: (408) 970-3403

Mellanox Technologies Ltd
PO Box 586 Hermon Building
Yokneam 20692
Israel

Tel: +972-4-909-7200
Fax: +972-4-959-3245

Contents

Contents 3
List of Figures 7
List of Tables 9
About this Manual 11
Chapter 1 Overview 13
1.1 Key Concepts And Terms 13
1.2 Tools Included In The IBADM Package 15
1.3 Dependencies 15
Chapter 2 IBADM Software Architecture In Brief 17
Chapter 3 Installation Of The IBADM Package 19
3.1 Prerequisites 19
3.2 Installing IBADM Software 19
3.3 Typical Configuration 19
3.4 Example Installation And Configuration 19
Chapter 4 Initialization 21
4.1 Initialization Steps 21
4.2 Initialization Example 21
Chapter 5 IBADM Configuration Files 23
5.1 ibadm.conf - IBADM Configuration File 23
5.2 ibadm.topo - IBADM Topology File 23
Chapter 6 Tools Usage 25
6.1 ibtopogen 25
6.1.1 ibtopogen Synopsis 25
6.1.2 ibtopogen Description 25
6.1.3 ibtopogen Examples 26
6.1.4 Topology Description Format 27
6.1.5 Racks File Format 27
6.1.6 Limitation 27
6.1.7 "Racks" Option Effect On Generated Topology Files 27
6.2 ibdmchk 29
6.2.1 ibdmchk Synopsis 29
6.2.2 ibdmchk Reports 30
6.2.2.1 Design Mode Report - A Good Case Example 30
6.2.2.2 Verification Mode Report - A Bad Case Example 31
6.3 ibls 33
6.3.1 ibls Synopsis 33
6.4 ibmon 35
6.4.1 ibmon Synopsis 35
6.4.2 Types of ibmon Reports 36

6.4.2.1 Topology Matching Reports 36
6.4.2.2 Performance Event (PM) Reports 37
6.4.2.3 Board Management (BM) Reports 38
6.5 ibfwmgr 39
6.5.1 ibfwmgr Synopsis 39
6.5.2 ibfwmgr Configuration And Initialization 39
6.5.2.1 Introduction 39
6.5.2.2 FW Related Configuration In /etc/ibadm.conf 40
6.5.2.3 FW Specification File 40
6.5.3 ibfwmgr Operation Modes 40
6.5.3.1 Query Mode 40
6.5.3.2 Burn Mode 41
6.5.4 Cluster FW Initialization 41
6.5.5 Cluster FW Update 42
6.5.6 FW Specification File Format 42
6.5.6.1 Example Of A FW Specification File 43
6.6 ibcon 45
6.6.1 ibcon Synopsis 45
6.7 ibcert 46
6.7.1 ibcert Synopsis 46
6.7.2 ibcert Logfile 46
6.7.3 Usage and Output Examples 46
Appendix A IBADM Custom Installation, Configuration, and Initialization 49
A.1 Custom Installation (Under Directory <dir>) Steps 49
A.2 IBADM Setup Wizard 49
A.3 Custom Configuration 49
A.3.1 An Example 49
A.4 Initialization 50
Appendix B Log Files 51
Appendix C IBNL Netlist Format 53
C.1 Overview 53
C.2 Main Concepts 53
C.3 File Format 54
C.4 Writing an IBNL System File 55
C.5 Formal Definition in YACC Syntax 57
C.6 Example IBNL for MTS14400 60
Appendix D PSID Assignment 65
D.1 PSID Field Structure 65
D.2 PSID Assignment and Integration Flow 65
Appendix E mlxburn 67
E.1 mlxburn Installation 67
E.1.1 IBADM Local Installation Software Dependencies 67
E.1.2 Installation Instructions 67

E.2 mlxburn Synopsys 67
E.3 Tool Description 68
E.3.1 Firmware Customization 68
E.4 Examples 68
E.5 Exit Return Values 69

List of Figures

Figure 1 Simple two node cluster: "Simple Cluster" 14
Figure 2 Software Components Of An IBADM System 17
Figure 3 A 16-node 2 level cluster named "Gnu" 26
Figure 4 Actual racks diagram for the "Gnu" cluster 27
Figure 5 Example of an ibfwmgr query run on a cluster 40
Figure 6 Example of a FW Specification file 44
Figure 7 IBNL File Sections 54

List of Tables

Table 1 Board Configuration Options And Supported Systems 24
Table 2 Design Mode Report - Good Case Example 30
Table 3 Verification Mode - Bad Case Example - MTS14400 and 3x MTS2400 31
Table 4 FW Specification Groups And Parameters 43
Table 5 Log Files 51
Table 6 PSID format 65
Table 7 External Software Dependencies 67

About this Manual

The InfiniBand Administration (IBADM) package described in this User's Manual aims to provide the means for performing System Administration tasks required by InfiniBand clusters.

This manual is organized in the following manner:

- Chapter 1 provides an overview of the IBADM package (page 13)
- Chapter 2 is a brief introduction to the SW architecture of the IBADM package (page 17)
- Chapter 3 provides instructions for the installation of the IBADM package (page 19)
- Chapter 4 describes initialization procedure for this IBADM package (page 21)
- Chapter 5 describes the IBADM configuration files (page 23)
- Chapter 6 describes IBADM tools usage (page 25)
- Appendix A presents IBADM Custom Installation, Configuration, and Initialization (page 49)
- Appendix B presents a summary of the log files produced by the tools (page 51)
- Appendix C presents the IBNL generic netlist format (page 53)
- Appendix D presents how to Specify a different PSID for a different firmware configuration (page 65)
- Appendix E presents the **mlxburn** tool for firmware image generation (for Mellanox devices) and burning of HCAs (page 67)

Intended Audience

The audience of this User Manual is System Administrators who have installed InfiniBand hardware and wish to monitor, maintain, and configure it.

Related Documentation

For InfiniBand related issues, please refer to the following specification:

- InfiniBand Architecture Specification Volume 1, Release 1.2

Conventions

Throughout this document, TCL shell commands are typed in the following format:

```
command
=> ...
```

where the second line is the command return.

For Example:

```
osm_opts configure
=> { -m_key -sm_key -subnet_prefix -m_key_lease_period -sweep_interval -max_wire_smps
-transaction_timeout -sm_priority -lmc -max_op_vls -reassign_lids -reassign_lfts -ignore_other_sm
-single_thread -no_multicast_option -disable_multicast -force_log_flush -subnet_timeout
-packet_life_time -head_of_queue_lifetime -local_phy_errors_threshold -overrun_errors_threshold
```

-polling_timeout -polling_retry_number -force_heavy_sweep -sweep_on_trap -max_port_profile
-port_profile_switch_nodes }

1 Overview

The InfiniBand Administration (IBADM) package described in this User's Manual aims to provide the means for performing System Administration tasks required by InfiniBand clusters.

The audience of this User Manual is System Administrators who have installed InfiniBand hardware and wish to monitor, maintain, and configure it.

A key feature of this tool-set is that it provides a uniform way to manage all the Mellanox devices and Systems.

> Note: For supported platforms, operating systems, firmware, drivers, etc., please refer to the IBADM release notes of your current revision.

The IBADM software package accesses IB devices through the IB connectivity (In-Band) and optionally through a secondary Ethernet network (Out-of-Band). Another method for communication with the devices is by attaching a USB-to-I2C adapter to the target IB system. The In-Band services are provided by running InfiniBand client software which communicates with the Subnet-Administrator (SA, which is part of OpenSM) and the IB devices. Out-of-Band control is provided by communicating with dedicated administration servers running on each of the Mellanox systems (equipped with a "management" option).

The rest of this overview chapter is organized in the following sections:

- "Key Concepts And Terms"
- "Tools Included In The IBADM Package"
- "Dependencies"

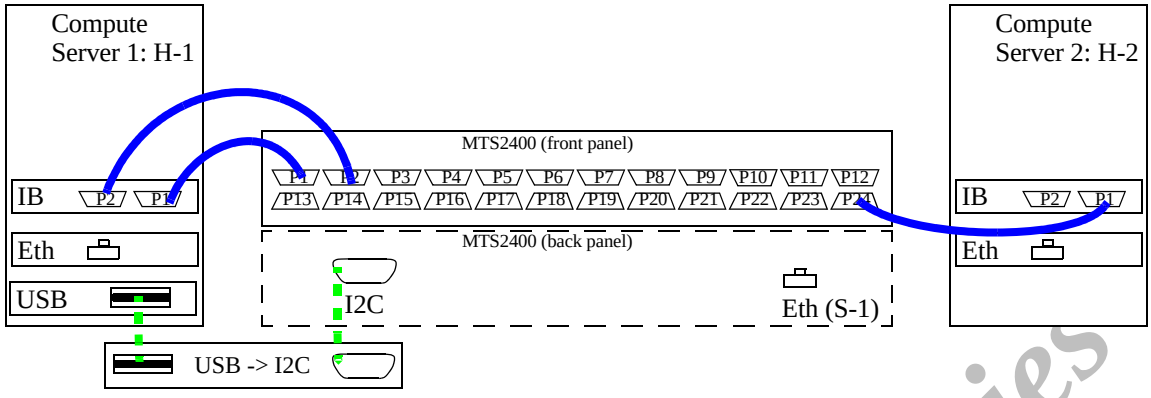
1.1 Key Concepts And Terms

Throughout this manual, there is frequent reference to various concepts and terms which are common to the general audience of System Administrators. However, there is also reference to InfiniBand-specific concepts and terms, a part of which are listed and briefly defined in the following list and throughout the document.

- **IB devices:** Integrated Circuits implementing InfiniBand compliant communication.
- **IB Fabric/Cluster/Subnet:** A set of IB devices connected by IB cables.
- **In-Band:** A term assigned to administration activities traversing the IB connectivity only.
- **Out-of-Band:** A term assigned to administration activities extending outside the IB connectivity (Ethernet or I2C).

A simple ***cluster*** example demonstrating the above terms is presented in Figure 1 (page 14). This cluster includes two compute servers (nodes) and an IB switch. The switch is an MTS2400-M, a Mellanox device managed via an Ethernet port by a resident PowerPC board. This cluster consists of three IB ***devices***. Each of the two servers' Host Channel Adapter (HCA) cards has one IB device. The third device is inside the MTS2400-M. The wide cables are 4X IB connections. All the administration tasks of the cluster can be carried out In-Band, that is through the IB network. The dashed cables show an alternative, Out-of-Band, control path of the switch via the I2C interface. The Ethernet ports on the back panel of the MTS2400 and the compute nodes can be connected to an optional secondary network providing yet another Out-of-Band control path.

Figure 1: Simple two node cluster: “Simple Cluster”



1.2 Tools Included In The IBADM Package

The IBADM package provides the following tools for administering the IB fabric:

- **ibtopogen:** A cluster design tool for fat-tree topologies
- **ibdmchk:** Checks the topology and provides the means to diagnose the IB routing
- **ibls:** Lists the available IB systems, devices and cables in the IB Subnet
- **ibmon:** Monitors all the devices and systems (for both IB and system related issues)
- **ibfwmgr:** Loads and tracks the IB devices firmware
- **ibcon:** Provides a view of some of the devices' internal configuration registers
- **ibcert:** A cluster certification flow

1.3 Dependencies

IBADM requires an OpenIB driver to be running on at least one of the nodes in the IB Fabric for various services. One of these nodes should be used to execute In-Band services. Also, one IB compliant Subnet-Administrator should be running and actively serving the cluster. For this task, it is possible to use OpenSM.

2 IBADM Software Architecture In Brief

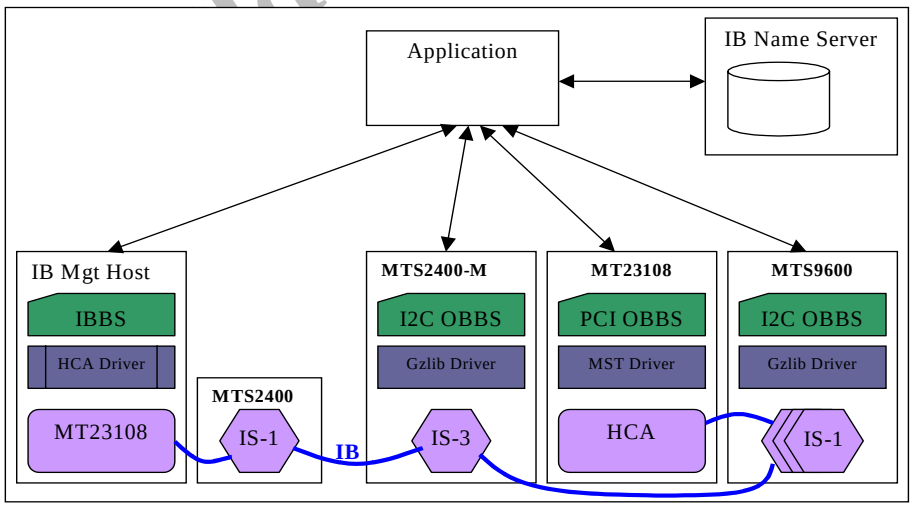
IBADM is based on multiple software agents running on every (switch or HCA) system in the IB fabric. These agents, referred to as "bus servers", wait for client socket connections and respond to a standard set of commands they receive through the socket. Examples of some commands are: list available IB devices, read/write to configuration registers of IB devices, reset devices, etc. Each bus server is classified by the type of physical bus connecting it to the IB devices. The following are the available bus types:

- **PCI Out-of-Band Bus Server (PCI OBBS):** The PCI bus server provides the interface for all the IB devices connected to the PCI bus within the system the bus server is running on. Since all device communications are handled via PCI transactions, this server is classified as Out-of-Band. By default, each compute-node (host) in the cluster runs such a bus server which serves its resident HCAs.
- **I2C Out-of-Band Bus Server (I2C OBBS):** The I2C bus server provides the interface for all the IB devices connected to the I2C bus within the system the bus server is running on. I2C bus servers are part of the embedded software provided for the Mellanox managed switch systems.
- **In-Band Bus Server (IBBS):** This server provides administration features over the IB fabric. The IBBS manages all the IB devices attached to the subnet it is connected to. It does that by contacting the subnet local Subnet Administrator (SA). It injects IB queries for performance monitoring and baseboard management using the standard IB Management datagrams (MADs). For non-standard features it uses Mellanox Vendor-Specific MADs.

IBADM applications interact with the available bus servers using the standard commands mentioned above. The following applications are provided as part of IBADM.

- **IB devices Names Server (IBNS):** This server consolidates for each device information available from the OBBSs and IBBSs. It then provides a "Names Interface" to the rest of the IBADM applications.
- **IBADM Application:** This may be a monitoring, configuration, or management application. Each application learns about the available devices by communicating with the IBNS. It then connects to the appropriate IBBS or OBBS server to access the devices.

Figure 2: Software Components Of An IBADM System



3 Installation Of The IBADM Package

3.1 Prerequisites

Before the installation of the IBADM package, it is necessary to guarantee the requirements listed here.

- One of the drivers (OpenIB/VAPI) has already been installed. The drivers can be obtained from several sources:
  - The "IB Gold Distribution" (IBGD) from http://www.mellanox.com
  - OpenIB stack available through www.openib.org
  - The VAPI driver from http://docs.mellanox.com (requires a Mellanox customer account).
- OpenSM revision 0.3.2 or later.

3.2 Installing IBADM Software

Now that all the groundwork has been done, it is possible to go ahead and install the software. This section will describe a **typical** installation while the details for how to perform a **custom** installation are provided in Appendix A,"IBADM Custom Installation, Configuration, and Initialization," on page 49. The typical installation is to place the package under the '/usr' directory. You will need to install the package on every compute node. Typical Installation Steps:

1. cd /tmp
2. tar -zxvf ibadm-<ibadm rev>.tgz
3. On a system with the IB Gold Distribution installed, use: /tmp/ibadm-<ibadm rev>/install.sh

3.3 Typical Configuration

The typical IBADM installation is configured to run all the utilities on the same machine that runs the In-Band services. The typical configuration uses In-Band as well as Out-of-Band control. While only one machine runs the In-Band services, all the compute nodes and IB switches (with management option) should run Out-of-Band services. For non-typical configurations, IBADM provides a setup wizard described in Appendix A,"IBADM Custom Installation, Configuration, and Initialization," on page 49. The following configuration steps are required to complete the standard installation:

1. Edit the file /etc/ibadm.conf modifying the name of the IBBS_HOST to the name or IP of the host that is assigned to run the In-Band services.
2. Install the modified file on all the hosts of the fabric.
3. Add all the compute node names (or IPs) as well as the names (or IPs) of the managed IB switches to the file /etc/ibadm.hosts on the machine to run IBADM tools. This provides the user with an alternative path to access the IB devices through Out-of-Band (Ethernet, PCI, I^2C) in case the In-Band connection experiences problems that prevent data transmission. (This is true for the tools: ibls, ibfwmgr, and ibcon.)

3.4 Example Installation And Configuration

The following is an installation of IBADM on the "Simple Cluster" and its configuration. In this example, it is assumed that all the systems are connected to a secondary Ethernet network. Following are the required steps.

1. Install the IBADM package on all the compute nodes. Assuming you are logged on to H-1 (with IB Gold Distribution installed), and the tar file is /tmp/ibadm-<ibadm rev>.tgz, run the following commands:
   tar xf /tmp/ibadm-<ibadm rev>.tgz
   ./ibadm-<ibadm rev>/install.sh -b openib
2. Prepare the /etc/ibadm.conf:
   sed ‘s/IBBS_HOST=.*/IBBS_HOST=H-1/’ /usr/lib/ibconf1.0/ibadm.conf > /etc/ibadm.conf
3. Distribute the file /etc/ibadm.conf to the other nodes:
   rcp /etc/ibadm.conf H-2:/etc/ibadm.conf
4. Create the file /etc/ibadm.hosts with the names/IPs of all IB systems:
   echo “H-1” > /etc/ibadm.hosts
   echo “H-2” >> /etc/ibadm.hosts
   echo “S-1” >> /etc/ibadm.hosts

   Note: When using system names rather than IP addresses, make sure the systems are accessible by name using ‘ping’. If required add them to the /etc/hosts file.

4 Initialization

Following is the sequence of operations required to bring up the IBADM in a **typical** configuration, for a **custom** configuration please see Appendix A,"IBADM Custom Installation, Configuration, and Initialization," on page 49.

4.1 Initialization Steps

For IBADM to run properly, it is required to provide a "topology file" which describes the InfiniBand connectivity of the cluster. The syntax of the file is described in the section 5.2 "ibadm.topo - IBADM Topology File" on page 23. This file allows IBADM to generate reports using the provided (readable alphanumeric) system names used in the file. Furthermore, IBADM compares the actual topology with the specified one in this file. Thus it can easily catch human errors in connecting the cluster.

For initialization follow these steps:

1. Turn on all the IB systems.
2. Start the IB HCA driver on all the hosts of the fabric.
3. Start OpenSM or any other Subnet Manager (SM) to initialize the fabric. To use OpenSM simply run: opensm.
4. Login to a machine you wish to run IBADM on.
5. Prepare a topology file describing your subnet (ibadm.topo):
   - The default location for this file is: /etc/ibadm.topo. The location can be modified by changing the IBBS_TOPO_FILE variable in /etc/ibadm.conf. (Check this file for details.)
   - Note that the host running your In-Band services should be called "H-1" in the topology file. The name can be changed by simply changing the IBBS_NAME variable in /etc/ibadm.conf.
6. Start IBADM services on all the compute hosts using the command: ibadm start
   Note: It is recommended to use the command 'ibadm status' to verify whether a server is up and running.

Now you may start using any of the IBADM tools: **ibcert**, **ibls**, **ibmon**, **ibcon**, and **ibfwmgr**.

4.2 Initialization Example

For our "Simple Cluster" example, the initialization steps are:

1. Make sure H-1, H-2 and S-1 are turned ON
2. Make sure the IB HCA driver is up on H-1 and H-2 (using "vapi status".)
3. Start OpenSM on H-1.
4. Create the topology file: /etc/ibadm.topo
5. Start IBADM services on all the hosts:
   rsh H-1 ibadm start
   rsh H-2 ibadm start

5 IBADM Configuration Files

This section of the manual describes the files that are required to be updated by the user prior to using the tools.

5.1 ibadm.conf - IBADM Configuration File

IBADM provides a set of configuration variables that controls its functionality and default values. These are placed in a file named /etc/ibadm.conf. Please read the file as it contains detailed descriptions of the variables. The following sections frequently refer to these configuration parameters.

5.2 ibadm.topo - IBADM Topology File

The topology file describes the IB connectivity and systems included in the cluster. The default file name is /etc/ibadm.topo. It serves two purposes:

1. Support for arbitrary system names to be later used in every report that IBADM generates.
2. Connectivity verification: The specified cluster topology is verified against the discovered one. Mismatch errors resulting from missing cables and/or wrong connections are reported by IBADM.

   Note: If the user changes the topology file after IBADM start, the user needs run 'ibadm restart' for the changes to take effect.

The topology file is composed of "system" sections. Each such section describes the connectivity of one system to other systems in the cluster. The first line of each section is a declaration of the system composed of a system-type, its local system-name section, and optional configuration details. The lines to follow until the next empty line describe the connections between the local system ports to the other systems. The following is a formal definition of a system section syntax. An example is listed afterwards.

```
SYSTEM-TYPE LOCAL-SYSTEM-NAME [CFG: <board Name>=R|Removed|4X|12X, [<board Name>=R|Removed|4X|12X]...]
        LOCAL-PORT-PANEL-NAME -> REM-SYS-TYPE REM-SYS-NAME REM-PORT-PANEL-NAME
        LOCAL-PORT-PANEL-NAME -> REM-SYS-TYPE REM-SYS-NAME REM-PORT-PANEL-NAME
        LOCAL-PORT-PANEL-NAME -> REM-SYS-TYPE REM-SYS-NAME REM-PORT-PANEL-NAME
        . . .

SYSTEM-TYPE LOCAL-SYSTEM-NAME [CFG: <board Name>=R|Removed|4X|12X, [<board Name>=R|Removed|4X|12X]...]
        LOCAL-PORT-PANEL-NAME -> REM-SYS-TYPE REM-SYS-NAME REM-PORT-PANEL-NAME
        LOCAL-PORT-PANEL-NAME -> REM-SYS-TYPE REM-SYS-NAME REM-PORT-PANEL-NAME
        . . .
. . .
```

Where:

- Board Name can be something like leafN or spineN (N is a number).
- SYSTEM-TYPE = REM-SYS-TYPE = MTEK43132|MTS2400-12T4|MTS2400-24|MTS2400|MTS9600|MTS14400|MTPB23108|MHX-CEXXX-T|MHXL-CEXXX-T|MTLP23108|MHEL-CFXXX-T|MHEAXX-XT
- LOCAL-SYSTEM-NAME = the name of the system described in this topology file section.

- LOCAL-PORT-PANEL-NAME = a name of the local system port. The numbers printed on the front panel are used together with L<N> for Leaf no. N or S<N> for Spine no. N.
- REM-SYS-NAME = the name of the system connected to the local port.
- REM-PORT-PANEL-NAME = a name of the remote system port. We use the numbers as printed on the front panel and L<N> for Leaf number N or S<N> for Spine number N.

Example. The following is a topology file for the “Simple Cluster”.

```
MTS2400 S-1
  P1 -> MT23108 H-1 P1
  P2 -> MT23108 H-1 P2
  P24 -> MT23108 H-2 P1
```

The optional “CFG:” section in the system declaration line describes the special customization of each board of the system. The format of the CFG field for a system that supports N leafs and M spines is:

CFG: leaf1=R,leaf2=12X,...leafN=4X, spine1=R, spine2=R, ... spineM=12X.

That is the CFG string is a set of comma-separated sub-fields. Each sub-field (if exists) describes some special configuration of a corresponding system board (starting with leaf1 and ending with the last spine).

All switch systems that have plug-in cards support the following syntax for the leaf-cfg and spine-cfg fields:

- A ‘D’ or space or empty string between the commas stands for the default configuration.
- ‘R’ or ‘Remove’ stands for “remove”. That is the board is not to be installed in the system.
- 12X stands for a 12X-port board configuration.

Further system specific specialization options for the system board are provided in the following table:

Table 1 - Board Configuration Options And Supported Systems

Option	Mnemonic	MTS9600	MTS14400
Remove the Board	R	Yes	Yes
Use 12X ports	12X	No	Also for spines
Add 4X ports	4X	No	Only for spines

Example. The following is an example of a definition-line in a topology file of the MTS9600 switch system. This switch system can have up to eight leafs and four spines. This example of the MTS9600 lacks (‘R’) leafs no.6,7 and 8, and lacks spines no. 3 and 4.

MTS9600 PartialGz1 CFG: leaf3=R,leaf5=R,leaf7=R,spine1=R

6 Tools Usage

This chapter of the manual describes the provided administration utilities, their tasks in the overall flow, their generated output, and user control.

6.1 ibtopogen

Unlike the Ethernet fabric, the IB fabric does not maintain host names in its devices. Furthermore, to efficiently manage a cluster of hosts and switches, IBADM needs a naming scheme that allows fast identification of each device and cable. To bridge this gap, IBADM uses a "topology" file which describes the fabric in terms of administrator-level names and system types, and maps the discovered IB devices attached to the fabric as a first step in any tool run.

The task of the **ibtopogen** tool described below is to generate such an InfiniBand topology file from a set of high level cluster parameters. It is quite a helpful tool: while writing a topology file for a 24-node cluster may not be a hard task, writing such a file for a 2400-node cluster could use some automation.

6.1.1 ibtopogen Synopsis

ibtopogen [options] NODESNUM N-STAGE L1RADIX [L2RADIX ...]

The parameters NODESNUM, N-STAGE, and L1RADIX are mandatory.

NODESNUM	- indicates the total number of computing hosts in the cluster
N-STAGE	- indicates the number of switch system levels
LxRADIX	- indicates the number of ports of each switch system at level x

Note: The number of LxRADIX parameters in the command line must be equal to N-STAGE. For example, if N-STAGE is 2 then there will be 2 parameters of L1RADIX, one for each level (in the order of appearance).

The command line **options** are:

-r CBB	- to select the over-subscription ratio. For example, -r 2 will use a 2:1 over-subscription. Default is 1. (CBB: Constant Bisection Bandwidth). Over subscription of 2 actually means that the number of cables connecting hosts to the first level of switches will be double the number of cables connecting the first level of switches to the second level of switches.
-p racks_file_name	- to get racks structures, and to attach rack-name prefixes to device names.
-o output_file_name	- to output the connectivity structures to a specific file. By default /etc/ibadm.topo is generated (if option is not used).
-h	- for help.
-v	- for **ibtopogen** version number.

6.1.2 ibtopogen Description

ibtopogen generates an N-STAGE fat-tree topology description based on the NODESNUM and the RADIX of the switches used in each level. This simplifies the task of generating a description that matches the formats of Infini-

Band monitoring tools. Furthermore, it provides a methodical way to connect and cable a multi-stage cluster in a readable manner.

By default **ibtopogen** assumes a full Constant Bisection Bandwidth (CBB), that is a 1:1 subscription. However, with the -r CBB command option, it is possible to choose a different over-subscription ratio. For example, -r 2 will set a 2:1 over-subscription (half CBB).

It is recommended to choose even numbers for the cluster parameters NODESNUM and LxRADIX. If not, **ibtopogen** rounds up the odd parameter(s) to the next higher integer, thus leaving some ports unused and resulting in a higher CBB than specified.

Moreover, the LxRADIX parameters should be given values equal to the number of ports of available switches. Examples from Mellanox Technologies products: LxRADIX should be 8 for MTEK43132, 24 for MTS2400, and 96 for MTS9600.

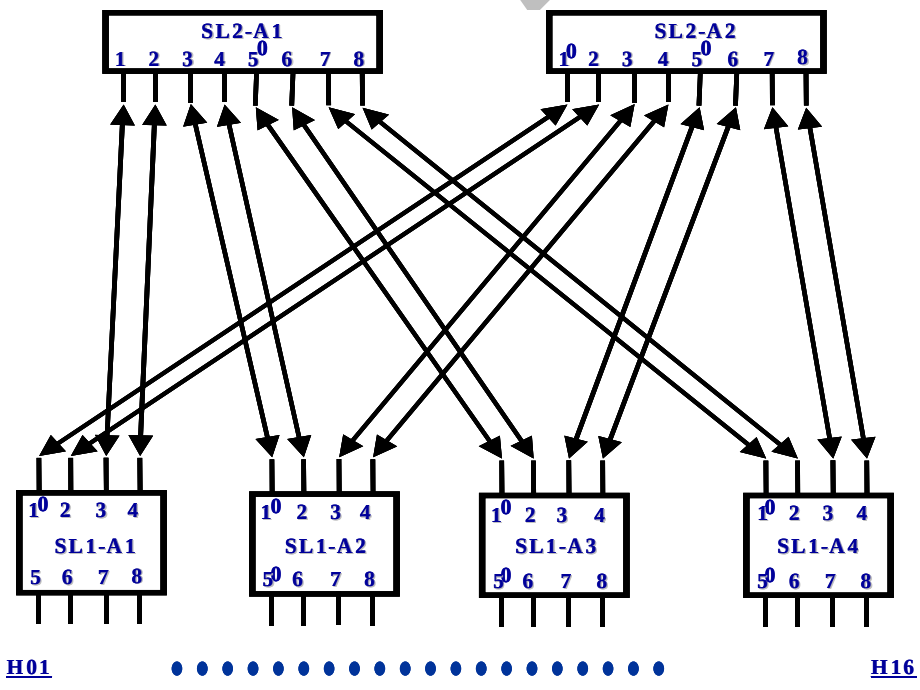
6.1.3 ibtopogen Examples

ibtopogen 288 2 24 24	288 hosts at Full CBB using two levels of 24 port switches
ibtopogen -r 2 192 2 24 96	192 hosts at Half CBB with 24-port L1 switches having 16 nodes and 8 connections to a single 96 node L2 switch.
ibtopogen -r 2 1152 2 96 96	1152 hosts at Half CBB comprised of 96-port L1 switches and 96-port spines

This cluster example, referred to as “Gnu” in this document, has the following topology:

ibtopogen 16 2 8 8	16 hosts at Full CBB with four 8-port L1 switches having 16 nodes and 16 connections to two 8-port L2 switches. The following figure illustrates this cluster.

Figure 3: A 16-node 2 level cluster named “Gnu”



6.1.4 Topology Description Format

The topology file is generated using generic names for hosts and switches, and matches the required format of the InfiniBand topology monitor: **ibmon**. Generic hostnames have the prefix 'H'; Level 1 switches start with 'SL1-' ; Level 2 switches start with 'SL2-'; and so forth.

The topology file generated includes system sections only for switch systems. Compute nodes are always included in the leaf switch connectivity.

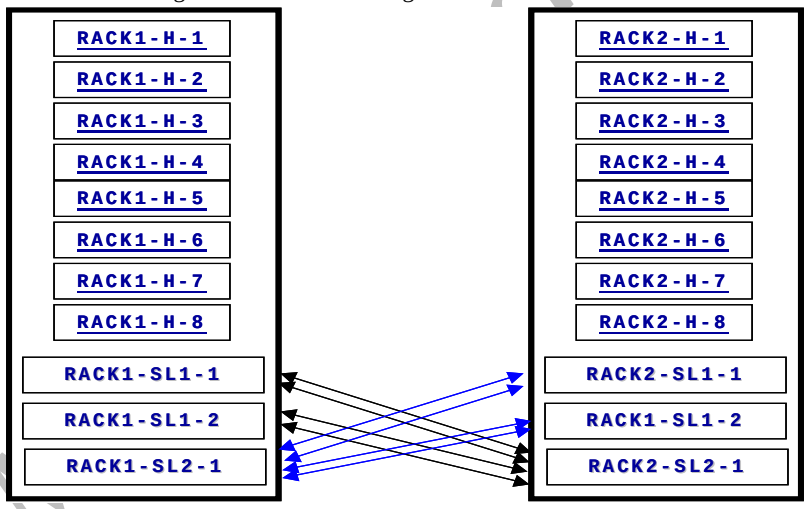
6.1.5 Racks File Format

Supplying a racks definition file to ibtopogen results in an improved topology file. **ibtopogen** uses racks definitions to assign a "rack" prefix to each system. The rack identifier eases the process of locating a failing system or cable when needed. Each line in the racks file (see below) describes a single rack by providing the number of systems it holds for each tree level. Note that the (defined) racks get populated in a sequential order.

Following is an example of a racks file (racks.txt) for the "Gnu" cluster composed of 16 hosts, 4 level-1 switches of 8 ports, and 2 level-2 switches of 8 ports. The cluster is organized as two identical racks holding: 8 hosts, 2 level-1 switches and a single level 2 switch.

```
# name   level: 0  1  2
Rack-1          8  2  1
Rack-2          8  2  1
```

Figure 4: Actual racks diagram for the "Gnu" cluster



6.1.6 Limitation

ibtopogen fills in the available slots in each rack serially. Thus it produces a non-symmetrical implementation if a "non-full" topology is used. The user should carefully plan the number of available slots in each level of each rack to obtain an optimal result.

6.1.7 "Racks" Option Effect On Generated Topology Files

Following is an example of two topology files output by **ibtopogen** for the "Gnu" cluster. The right column is the output using a rack file, the left column is without using a "rack file".

ibtopogen 16 2 8 8

```
MTEK43132 SL1-1
    P1 -> MT23108 H-1 P1
    P2 -> MT23108 H-2 P1
    P3 -> MT23108 H-3 P1
    P4 -> MT23108 H-4 P1
    P5 -> MTEK43132 SL2-1 P1
    P6 -> MTEK43132 SL2-1 P2
    P7 -> MTEK43132 SL2-2 P1
    P8 -> MTEK43132 SL2-2 P2

MTEK43132 SL1-2
    P1 -> MT23108 H-5 P1
    P2 -> MT23108 H-6 P1
    P3 -> MT23108 H-7 P1
    P4 -> MT23108 H-8 P1
    P5 -> MTEK43132 SL2-1 P3
    P6 -> MTEK43132 SL2-1 P4
    P7 -> MTEK43132 SL2-2 P3
    P8 -> MTEK43132 SL2-2 P4

MTEK43132 SL1-3
    P1 -> MT23108 H-9 P1
    P2 -> MT23108 H-10 P1
    P3 -> MT23108 H-11 P1
    P4 -> MT23108 H-12 P1
    P5 -> MTEK43132 SL2-1 P5
    P6 -> MTEK43132 SL2-1 P6
    P7 -> MTEK43132 SL2-2 P5
    P8 -> MTEK43132 SL2-2 P6

MTEK43132 SL1-4
    P1 -> MT23108 H-13 P1
    P2 -> MT23108 H-14 P1
    P3 -> MT23108 H-15 P1
    P4 -> MT23108 H-16 P1
    P5 -> MTEK43132 SL2-1 P7
    P6 -> MTEK43132 SL2-1 P8
    P7 -> MTEK43132 SL2-2 P7
    P8 -> MTEK43132 SL2-2 P8

MTEK43132 SL2-1
    P1 -> MTEK43132 SL1-1 P5
    P2 -> MTEK43132 SL1-1 P6
    P3 -> MTEK43132 SL1-2 P5
    P4 -> MTEK43132 SL1-2 P6
    P5 -> MTEK43132 SL1-3 P5
    P6 -> MTEK43132 SL1-3 P6
    P7 -> MTEK43132 SL1-4 P5
    P8 -> MTEK43132 SL1-4 P6

MTEK43132 SL2-2
    P1 -> MTEK43132 SL1-1 P7
    P2 -> MTEK43132 SL1-1 P8
    P3 -> MTEK43132 SL1-2 P7
    P4 -> MTEK43132 SL1-2 P8
    P5 -> MTEK43132 SL1-3 P7
    P6 -> MTEK43132 SL1-3 P8
    P7 -> MTEK43132 SL1-4 P7
    P8 -> MTEK43132 SL1-4 P8
```

ibtopogen -p racks.txt 16 2 8 8 -o racks_out.txt

```
MTEK43132 Rack-1-SL1-1
    P1 -> MT23108 Rack-1-H-1 P1
    P2 -> MT23108 Rack-1-H-2 P1
    P3 -> MT23108 Rack-1-H-3 P1
    P4 -> MT23108 Rack-1-H-4 P1
    P5 -> MTEK43132 Rack-1-SL2-1 P1
    P6 -> MTEK43132 Rack-1-SL2-1 P2
    P7 -> MTEK43132 Rack-2-SL2-1 P1
    P8 -> MTEK43132 Rack-2-SL2-1 P2

MTEK43132 Rack-1-SL1-2
    P1 -> MT23108 Rack-1-H-5 P1
    P2 -> MT23108 Rack-1-H-6 P1
    P3 -> MT23108 Rack-1-H-7 P1
    P4 -> MT23108 Rack-1-H-8 P1
    P5 -> MTEK43132 Rack-1-SL2-1 P3
    P6 -> MTEK43132 Rack-1-SL2-1 P4
    P7 -> MTEK43132 Rack-2-SL2-1 P3
    P8 -> MTEK43132 Rack-2-SL2-1 P4

MTEK43132 Rack-2-SL1-1
    P1 -> MT23108 Rack-2-H-1 P1
    P2 -> MT23108 Rack-2-H-2 P1
    P3 -> MT23108 Rack-2-H-3 P1
    P4 -> MT23108 Rack-2-H-4 P1
    P5 -> MTEK43132 Rack-1-SL2-1 P5
    P6 -> MTEK43132 Rack-1-SL2-1 P6
    P7 -> MTEK43132 Rack-2-SL2-1 P5
    P8 -> MTEK43132 Rack-2-SL2-1 P6

MTEK43132 Rack-2-SL1-2
    P1 -> MT23108 Rack-2-H-5 P1
    P2 -> MT23108 Rack-2-H-6 P1
    P3 -> MT23108 Rack-2-H-7 P1
    P4 -> MT23108 Rack-2-H-8 P1
    P5 -> MTEK43132 Rack-1-SL2-1 P7
    P6 -> MTEK43132 Rack-1-SL2-1 P8
    P7 -> MTEK43132 Rack-2-SL2-1 P7
    P8 -> MTEK43132 Rack-2-SL2-1 P8

MTEK43132 Rack-1-SL2-1
    P1 -> MTEK43132 Rack-1-SL1-1 P5
    P2 -> MTEK43132 Rack-1-SL1-1 P6
    P3 -> MTEK43132 Rack-1-SL1-2 P5
    P4 -> MTEK43132 Rack-1-SL1-2 P6
    P5 -> MTEK43132 Rack-2-SL1-1 P5
    P6 -> MTEK43132 Rack-2-SL1-1 P6
    P7 -> MTEK43132 Rack-2-SL1-2 P5
    P8 -> MTEK43132 Rack-2-SL1-2 P6

MTEK43132 Rack-2-SL2-1
    P1 -> MTEK43132 Rack-1-SL1-1 P7
    P2 -> MTEK43132 Rack-1-SL1-1 P8
    P3 -> MTEK43132 Rack-1-SL1-2 P7
    P4 -> MTEK43132 Rack-1-SL1-2 P8
    P5 -> MTEK43132 Rack-2-SL1-1 P7
    P6 -> MTEK43132 Rack-2-SL1-1 P8
    P7 -> MTEK43132 Rack-2-SL1-2 P7
    P8 -> MTEK43132 Rack-2-SL1-2 P8
```

6.2 ibdmchk

InfiniBand uses Static Routing tables for routing packets. The static routes are computed and assigned by the Subnet Manager (e.g., OpenSM) controlling the fabric. **ibdmchk** is a topology checker which provides the means to diagnose the IB routing during either the cluster design phase or verification (certification) phase. Thus, **ibdmchk** has two operation modes: *cluster design* and *cluster verification*.

The *cluster design* mode is intended for usage prior to cluster building, In this mode, **ibdmchk** performs basic checks on a network specified by a topology file. After simulating the SM LID assignment and routing algorithms, it reports an HCA-to-HCA paths depth histogram, a link over-subscription histogram, and any potential credit-deadlock in the resulting routing scheme.

The *cluster verification* mode is intended for usage after the cluster building. Prior to using **ibdmchk**, OpenSM must be running with the debug option '-V' or '-D 0x43'. In this mode, **ibdmchk** reports the subnet and FDB tables into the files /tmp/subnet.lst, /tmp/osm.fdbs and /tmp/osm.mcfdbs. Based on these files, the utility checks all HCA to HCA connectivity. Further analysis for potential credit deadlocks is performed and reported. In case the specified LMC value is greater than 0, **ibdmchk** reports histograms, one per a pair of ports, detailing how many systems and nodes are common among the different paths between each pair.

Each of the two modes has its own set of command line flags. They are described in the following section.

6.2.1 ibdmchk Synopsis

Cluster Design Mode:

ibdmchk [-v][-h] -t <topology file> -n <SM Node> -p <SM Port> [-e] [-l <lmc>] [-u]

Where the ***required*** arguments are:

-t|--topo <topo file>	To provide the topology file specifying the fabric
-n|--node <SM Node>	To provide the name of the Subnet Manager node (syntax: <Topo-File-System>/U1)
-p|--port <SM Port>	To provide the port number by which the SM nodes are attached to the fabric

Optional arguments are:

-v | --verbose	For verbose mode
-h | --help	To get this help message
-l |--lmc <lmc>	If lmc > 0 then 2^lmc lids will be assigned to each port
-e|--enh	To use an enhanced routing algorithm if lmc > 0 and report the resulting paths correlation histogram (using the same system/node names)
-u|--updn	To use the up/down routing algorithm instead of OpenSM's min-hop algorithm

Cluster Verification Mode:

ibdmchk [-v][-h] [-s <subnet file>] [-f <fdb file>] [-m <mc fdb file>] < -l <lmc>

Where the ***required*** argument is:

-l | --lmc <lmc>	The LMC value used while running OpenSM. If not specified, the default is 0.

Optional arguments are:

-v|--verbose	For verbose mode
-h|--help	To get this help message
-s | --subnet <file>	To provide the OpenSM subnet.lst file (Default: /tmp/subnet.lst)
-f | --fdb <file>	To provide the OpenSM dump of Unicast LFDB. Use OpenSM with '-D 0x41' to generate the dump. (Default: /tmp/osm.fdb)
-m | --mcfdb <file>	To provide the OpenSM dump of Multicast LFDB. Use OpenSM with '-D 0x41' to generate the dump. (Default: /tmp/osm.fdb)

6.2.2 ibdmchk Reports

ibdmchk provides slightly different reports for the *design* and *verification* modes. The reports provide various types of analysis. One report per mode is listed below covering a good cluster case and a bad cluster case.

The "-E-" message prefix designates an error, thus it should not be ignored by the user.

The "-I-" prefix is used to mark an "Informative" message.

6.2.2.1 Design Mode Report - A Good Case Example

Table 2 - Design Mode Report - Good Case Example

Report Example	ibdmchk Operation
IBDMCHK Cluster Design Mode: Topology File .. ibdm/Clusters/FullGnu.topo SM Node ..... H01/U1 SM Port ..... 1 LMC ..... 0 -I- Parsing topology definition:ibdm/Clusters/FullGnu.topo -I- Defined 17/22 systems/nodes -I- Assigned 22 LIDs (lmc=0) in 5 steps	Provides information about the command line parameters and initial parsing of the provided topology file. The first steps of LID assignment are run and the number of assigned lids is reported.
-I- Init Min Hops Tables in:3 steps ---------- CA to CA : MIN HOP HISTOGRAM ---------- HOPS NUM-CA-CA-PAIRS 1 64 3 192 ---------- -I- Found worst min hops:3 at node:GNU1/main/U3 to node:H08/U1	Calculates the minimal hops tables for all switches. A histogram for the minimal hops from every HCA to every HCA is provided. Note: Hop count ignores the first hop from the HCA o the nearest switch.
---------- TRACE PATH BY MIN HOPS ---------- -I- Tracing by Min Hops from lid:2 to lid:11 [0] FROM Node:GNU1/main/U3 Port:1 TO Node:GNU1/main/U2 Port:8 [1] FROM Node:GNU1/main/U2 Port:1 TO Node:GNU1/main/U4 Port:8 [2] FROM Host:GNU1 Plug:M/P8 Node:GNU1/main/U4 Port:3 TO Host:H08 Plug:P1 Node:H08/U1 Port:1	The worst path is traced from the switch nearest to the HCA to the target HCA. Each link traversed has "From" and "To" lines designating the two nodes and ports. Also note that some links have two lines: if the link goes from one system to another, the details of the system ports are provided as well.
-I- Using standard OpenSM Routing ---------- LINK SUBSCRIPTIONS HISTOGRAM ---------- NUM-LIDS COUNT 1 16 2 6 3 10 4 12 5 4	Reports what routing algorithm is being used (Up/Down if the -u option is provided), then the histogram for the number of paths assigned to a link.

Table 2 - Design Mode Report - Good Case Example (Continued)

Report Example	ibdmchk Operation
-I- Verifying all CA to CA paths ... ---------- CA to CA : LFT ROUTE HOP HISTOGRAM ----- HOPS NUM-CA-CA-PAIRS 1 48 3 192 ---------- -I- Scanned:240 CA to CA paths	After routing is performed a check is run to verify all HCA to **other** HCA routes. (The hop count histogram provided still ignores the first hop. So in this case the max number of hops is really 4.)
-I- Scanning all multicast groups for loops and connectivity...	If a multicast group gets assigned, this section describes it (see Verification example below)
-I- Automatically recognizing the tree root nodes ... -I- Recognized 2 root nodes: GNU1/main/U1 GNU1/main/U2	In order to report "Credit Loops" in the fabric, the code automatically recognizes the "roots" nodes of the tree. If it fails to find any, it falls back to DFS algorithm.
-I- Tracing all CA to CA paths for Credit Loops potential ... -I- No credit loops found in:240 CA to CA paths	Traces through all HCA to HCA paths looking for "credit loops". It reports any potential deadlock loop with a "-E-" message.
-I- Scanning all multicast groups for Credit Loops Potential ..	A similar check to the one above is performed for the multicast group routing.

6.2.2.2 Verification Mode Report - A Bad Case Example

Table 3 - Verification Mode - Bad Case Example - MTS14400 and 3x MTS2400

Example Report	ibdmchk Operation
IBDMCHK OpenSM Routing Verification Mode: FDB File = /tmp/osm.fdbs MCFDB File = /tmp/osm.mcfdbs Subnet File = /tmp/subnet.lst LMC = 0 -I- Parsing OpenSM Subnet file:/tmp/subnet.lst -I- Defined 47/47 systems/nodes -I- Parsing OpenSM FDBs file:/tmp/osm.fdbs -I- Defined 517 fdb entires for:11 switches -I- Parsing OpenSM Multicast FDBs file:/tmp/osm.mcfdbs -I- Defined 0 Multicast Fdb entires for:11 switches	Provides information about the command line parameters and initial parsing of the subnet.lst, osm.fdbs and osm.mcfdbs files.
-I- Init Min Hops Tables in:4 steps ---------- CA to CA : MIN HOP HISTOGRAM ---------- HOPS NUM-CA-CA-PAIRS 1 432 3 576 5 288 -I- Found worst min hops:5 at node:node:0002c90000000091 to node:node:0002c90000000061	Reports, per switch, the results of min-hop tables calculation. Note that the first hop from the HCA to the switch is ignored.

Table 3 - Verification Mode - Bad Case Example - MTS14400 and 3x MTS2400 (Continued)

Example Report	ibdmchk Operation
`` ------------------------- TRACE PATH BY MIN HOPS ------------------- -I- Tracing by Min Hops from lid:37 to lid:25 [0] FROM Host:system:0002c90000000091 Plug:node:0002c90000000097/P13 Node:node:0002c90000000091 Port:13 TO Host:system:0002c90000000097 Plug:node:0002c90000000097/P1 Node:node:0002c90000000097 Port:1 [1] FROM Host:system:0002c90000000097 Plug:node:0002c90000000097/P7 Node:node:0002c90000000097 Port:7 TO Host:system:0002c90000000093 Plug:node:0002c90000000097/P13 Node:node:0002c90000000093 Port:13 [2] FROM Host:system:0002c90000000093 Plug:node:0002c90000000099/P19 Node:node:0002c90000000093 Port:19 TO Host:system:0002c90000000099 Plug:node:0002c90000000099/P7 Node:node:0002c90000000099 Port:7 [3] FROM Host:system:0002c90000000099 Plug:node:0002c90000000099/P1 Node:node:0002c90000000099 Port:1 TO Host:system:0002c90000000095 Plug:node:0002c90000000099/P13 Node:node:0002c90000000095 Port:13 [4] FROM Host:system:0002c90000000095 Plug:node:0002c90000000095/P1 Node:node:0002c90000000095 Port:1 TO Host:system:0002c90000000061 Plug:P1 Node:node:0002c90000000061 Port:1 ```	Lists the trace of one of the maximal hops paths
`` -I- Verifying all CA to CA paths ... ---------------------- CA to CA : LFT ROUTE HOP HISTOGRAM ------------- HOPS NUM-CA-CA-PAIRS 1 396 3 576 5 288 --- -I- Scanned:1260 CA to CA paths ```	Checks that all HCAs are accessible from all other HCAs by traversing the route tables provided by OpenSM
-I- Scanning all multicast groups for loops and connectivity...	(In this example no multicast groups were setup)
`` -I- Automatically recognizing the tree root nodes ... -I- Recognized 6 root nodes: node:0002c9000000009a node:0002c9000000009b node:0002c9000000009c node:0002c9000000009d node:0002c9000000009e node:0002c9000000009f ```	In order to report “Credit Loops” in the fabric, the code automatically recognizes the “roots” nodes of the tree. If it fails to find any, it falls back to DFS algorithm.
`` -I- Tracing all CA to CA paths for Credit Loops potential ... -E- Potential Credit Loop on Path from:node:0002c90000000061/1 to:node:0002c90000000001/1 Going:Down from:node:0002c90000000099 to:node:0002c90000000093 Going:Up from:node:0002c90000000093 to:node:0002c90000000097 Going:Down from:node:0002c90000000097 to:node:0002c90000000091 -E- Potential Credit Loop on Path from:node:0002c90000000061/1 to:node:0002c...... Going:Down from:node:0002c90000000099 to:node:0002c90000000093 Going:Up from:node:0002c90000000093 to:node:0002c90000000097 Going:Down from:node:0002c90000000097 to:node:0002c90000000091 Going:Down from:node:0002c90000000099 to:node:0002c90000000093 Going:Up from:node:0002c90000000093 to:node:0002c90000000097 Going:Down from:node:0002c90000000097 to:node:0002c90000000091 -E- Found:72 CA to CA paths that can cause credit loops. ```	Traces through all HCA to HCA paths looking for “credit loops”. It reports any potential deadlock loop with a “-E-” message.
-I- Scanning all multicast groups for Credit Loops Potential ...	Error as above might be reported

6.3 ibls

Like the Unix command "ls", the tool **ibls** lists all the known devices of the IB Fabric. It uses system names as defined in the topology description file described in Section 6.1, "ibtopogen" and presents the requested data to the standard output. Several options of **ibls** enable the user to obtain various details on the entire fabric, on a specific system, or on a specific device.

6.3.1 ibls Synopsis

ibls [-h] [-v] [-V] [-b] [-s] [-d] [-D] [-l] [-I] [-L] [-g guid] [-n name] [-i device-name] [-e]

If no options are provided in the command line, **ibls** returns information about all the available systems, devices and links.

The command line **options** are:

-h - help.

-v - version.

-V - verbose mode (DEBUG | WARNING | INFORM).

-b - list bus servers (Out-of-Band: PCI, I^2C, Ethernet; In-Band: InfiniBand).

-s - list systems.

-d - list devices.

-D - list devices with all GUIDs information.

-l - list links.

-I - list system internal links too.

-L - list links with all GUIDs information.

-g - list devices with a matching guid or name (system node or port).

-n - list devices with a matching name only.

-i - list devices with a matching device name only (MT43132 | MT23108 | MT25208 | MT47396).

-e - list topology checker errors.

Note: The topology check (-e) option is *not* executed by default.

The following is an example of **ibls** output:

```
-I- IBLS:

-I- Systems:
SysGuid             Name         SystemType
0xffffffffffffffff  c1_SL1-1     MTS14400
0x0002c9010976a253  swlab227     MTHX-CExxx-T/MHXL-CFxxx-T
0x0002c9000100d050  c1_swlab221  MTHX-CExxx-T/MHXL-CFxxx-T
0x0002c9000100d060  swlab222     MTHX-CExxx-T/MHXL-CFxxx-T
0x0002c9010a66c293  swlab123     MHEL-CFxxx-T
0x0002c90109769a03  c1_swlab228  MTHX-CExxx-T/MHXL-CFxxx-T
0x0002c90109768ea3  swlab229     MTHX-CExxx-T/MHXL-CFxxx-T
0x0002c901093dc5d3  swlab225     MTHX-CExxx-T/MHXL-CFxxx-T
0x0002c900000000000 SL2-2        MTS2400
0xa22029083222f0ff  SL2-1        MTS9600
0x0002c90001203470  c1_swlab224  MTHX-CExxx-T/MHXL-CFxxx-T
```

```
-I- Devices:
NodeGuid            Name                DeviceType     Host
0x002c90109fb2b501  SL2-1/leaf1/U1      MT43132
0x002c90109fb2b502  SL2-1/leaf1/U2      MT43132
0x002c90109fb2b503  SL2-1/leaf1/U3      MT43132
0x002c90109fb2b504  SL2-1/leaf1/U4      MT43132
0x002c90109fb2b505  SL2-1/leaf1/U5      MT43132
0x002c90109fb2b506  SL2-1/leaf1/U6      MT43132
0x002c90109fb2b507  SL2-1/leaf1/U7      MT43132
0x0002c90000000000  SL2-2/U1            MT47396
0x0002c9010a335470  c1_SL1-1/leaf10/U1  MT47396
0x0002c9010ab8a170  c1_SL1-1/spine2/U1  MT47396
0x0002c9010b073e08  c1_SL1-1/spine2/U2  MT47396
0x0002c9010ab8a180  c1_SL1-1/spine2/U3  MT47396
0x0002c9000100d050  c1_swlab221         MT23108        swlab221
0x0002c90001203470  c1_swlab224         MT23108        swlab224
0x0002c90109769a00  c1_swlab228/U1      MT23108        swlab228
0x0002c9010a66c290  swlab123/U1         MT25208        swlab123
0x0002c9000100d060  swlab222            MT23108        swlab222
0x0002c901093dc5d0  swlab225/U1         MT23108        swlab225
0x0002c9010976a250  swlab227/U1         MT23108        swlab227
0x0002c90109768ea0  swlab229/U1         MT23108        swlab229

-I- Links:
FromPort                    ToPort
SL2-1/L1/P12             <-> c1_SL1-1/L10/P2
SL2-1/L1/P6              <-> swlab123/P1
SL2-2/P10                <-> swlab225/P1
SL2-2/P13                <-> c1_SL1-1/L10/P12
SL2-2/P15                <-> SL2-2/P18
SL2-2/P17                <-> swlab227/P1
SL2-2/P2                 <-> swlab229/P1
SL2-2/P23                <-> swlab222/P1
c1_SL1-1/L10/P1          <-> c1_swlab228/P1
c1_SL1-1/L10/P5          <-> c1_swlab221/P1
c1_SL1-1/L10/P8          <-> c1_swlab224/P1
```

6.4 ibmon

ibmon, is the InfiniBand Fabric Monitor tool. It provides vital feedback, at the system and cable levels, concerning the state, errors, data traffic, etc., of the IB communications and systems. It can also monitor voltage, temperature, and other parameters in switch systems that use Mellanox silicon devices. **ibmon** should be run continuously to monitor the "health" of the fabric.

The user is provided with a simplified summary of the events on the standard output. However, several detailed log files are preserved in a pre-defined directory. The default directory is /tmp/ibmon.

ibmon generates the following log files:

ibmon_events.log	- reports all threshold violation events.
ibmon_sweeps.log	- reports all progressing PM register values from successive sweeps.
ibmon_last_sweep.log	- log file for the last sweep results.
ibmon_bbm.log	- log file for baseboard management reports.

6.4.1 ibmon Synopsis

ibmon [-h] [-v] [-V] [-b] [-c] [-s|-t sweep-time] [-m] [-f out-dir]
[-n max-num-MAD] [-o time-out] [-x pkt-thd] [-d data-thd] [-e err-thd] [-r link-recover-thd]
[-l link-down-thd] [-i vl15-drop-thd] [-q temp-value]

The command line **options** are:

-h - for help.
-v - for ibmon version number.
-V - for verbose mode (DEBUG | WARNING | INFORM).
-b - for BM mode (reports Baseboard queries).
-c - to clear all counters first.
-s - to perform a single sweep and exit.
-t - to define the time period between sweeps in msec.
-f - to define the directory where report files are to be placed.
-m - to include reports on SMA (port zero).
-o - to define the time-out period in msec.
-n - to define the number of multi-MAD packets.

In addition, **ibmon** has the following **threshold**-parameter setting options that allow defining irregular events. (All threshold values are subject to the *InfiniBand Architecture Specification* defined limitations).

-x - to define the threshold for counter of number of packets sent or received
-d - to define the threshold for counter of number of data-octets sent or received
-r - to define the threshold for counter of link-recovery occurrences
-l - to define the threshold for counter of link-down occurrences
-i - to define the threshold for counter of VL-15 dropped packets
-e - to define the threshold for counter of total number of errors (also for all mandatory counters per the InfiniBand Architecture Specification and are not mentioned above)
-q - to define the threshold for temperature in Celsius degrees [C].

6.4.2 Types of ibmon Reports

This section provides details of the various reports produced by **ibmon** with some examples. Specifically, it presents Topology Matching reports, Performance Event reports, and Board Management reports.

6.4.2.1 Topology Matching Reports

ibmon reports system names and cable names by matching the actual IB topology to a given topology file. In order to match the topologies, ibmon requires the name of the machine running the IBBS. By default, this machine is named H-1; it is possible to define a different name by setting the IBBS_NAME configuration parameter in the /etc/ibadm.conf file. (See the file for details.)

Types of error messages. ibmon produces the following three types of error messages:

1) Link exists only in the actual topology (extracted by the Subnet-Administrator) but not in the topology file.
2) Link exists only in the topology file but not in the real topology.
3) Link found in both the topology file and the physical fabric, however the link ports do not match in both files.

Each line in the ibmon report files will start with one of the two following strings:

-I- to indicate an information line
-E- to report an error

Some error-report lines may include the string '(I)'. By this ibmon indicates that the error was found possibly on a system internal connection (link), or on any link that the **ibmon** checker was unable to locate in the topology file.

Example: The following list is the topology file presented in Section 6.1.3, "ibtopogen Examples".

```
MTEK43132 SL1-1
      P1 -> MT23108 H-1 P1
      P2 -> MT23108 H-2 P2
      P3 -> MT23108 H-3 P1
      P4 -> MT23108 H-4 P1
      P5 -> MTEK43132 SL2-1 P1
      P6 -> MTEK43132 SL2-1 P2
      P7 -> MTEK43132 SL2-2 P1
      P8 -> MTEK43132 SL2-2 P2

MTEK43132 SL1-2
      P1 -> MT23108 H-5 P1
      P2 -> MT23108 H-6 P1
      P3 -> MT23108 H-7 P1
      P4 -> MT23108 H-8 P1
      P5 -> MTEK43132 SL2-1 P3
      P6 -> MTEK43132 SL2-1 P4
      P7 -> MTEK43132 SL2-2 P3
      P8 -> MTEK43132 SL2-2 P4

MTEK43132 SL1-3
      P1 -> MT23108 H-9 P1
      P2 -> MT23108 H-10 P1
      P3 -> MT23108 H-11 P1
      P4 -> MT23108 H-12 P1
      P5 -> MTEK43132 SL2-1 P5
      P6 -> MTEK43132 SL2-1 P6
      P7 -> MTEK43132 SL2-2 P5
      P8 -> MTEK43132 SL2-2 P6

MTEK43132 SL1-4
      P1 -> MT23108 H-13 P1
      P2 -> MT23108 H-14 P1
      P3 -> MT23108 H-15 P1
      P4 -> MT23108 H-16 P1
      P5 -> MTEK43132 SL2-1 P7
      P6 -> MTEK43132 SL2-1 P8
      P7 -> MTEK43132 SL2-2 P7
      P8 -> MTEK43132 SL2-2 P8

MTEK43132 SL2-1
      P1 -> MTEK43132 SL1-1 P5
      P2 -> MTEK43132 SL1-1 P6
```

```
      P3 -> MTEK43132 SL1-2 P5
      P4 -> MTEK43132 SL1-2 P6
      P5 -> MTEK43132 SL1-3 P5
      P6 -> MTEK43132 SL1-3 P6
      P7 -> MTEK43132 SL1-4 P5
      P8 -> MTEK43132 SL1-4 P6

MTEK43132 SL2-2
      P1 -> MTEK43132 SL1-1 P7
      P2 -> MTEK43132 SL1-1 P8
      P3 -> MTEK43132 SL1-2 P7
      P4 -> MTEK43132 SL1-2 P8
      P5 -> MTEK43132 SL1-3 P7
      P6 -> MTEK43132 SL1-3 P8
      P7 -> MTEK43132 SL1-4 P7
      P8 -> MTEK43132 SL1-4 P8
```

The following report is the output of **ibmon** when run on the topology:

```
-I- ---------------------------------------------------------------------------------------------------------------------------------
-E- Found some Topology mismatches:
-E- Wronge cable connecting: SL1-1/P2 to:H-2/P2 instead  of H-2/P1
-E-
-I- ---------------------------------------------------------------------------------------------------------------------------------
```

6.4.2.2 Performance Event (PM) Reports

The *performance counters* reports are placed in three files normally located under /tmp/ibmon/. **ibmon** also prints parts of the reports to the stdout whenever the threshold of a PM counter is crossed in some IB port. Following are some details and examples.

Example of error messages printed to stdout. The following report example is of an error detected on one of the internal port counters. The line describes an external cable/port error with the system name and front panel ports listed.

```
-E- 09:44:53 Threshhold Exceeded on Cable: H-16/P1 <= SL1-A1/P2
```

Example of a last sweep log file: The file /tmp/ibmon/ibmon_last_sweep.log contains information of all performance counters from the last sweep.

The first line in the file provides the time of sweep. The second line provides the list of counters 'swept'. After that all counter values of each IB device in the cluster are listed.

```
TIME : Mon Feb 09 10:06:41 UTC 2004
# guid port symbol_error_counter link_error_recovery_counter link_down_counter port_rcv_errors port_rcv_remote
_physical_errors port_rcv_switch_relay_errors port_xmit_discard port_xmit_constraint_errors port_rcv_constrain
t_errors local_link_integrity_errors excesive_buffer_errors vl15_dropped port_xmit_data port_rcv_data port_xmi
t_pkts port_rcv_pkts
###
H-16            1  : 47 0 4 0 0 0 150 0 0 0 0 0 5614848 5612904 77984 77957
H-16            2  : 44 0 4 0 0 0 0 0 0 0 0 0 9936 10008 138 139
SL1-A1          1  : 0 0 0 0 0 0 0 0 0 0 0 0 7128 7056 99 98
SL1-A1          2  : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SL1-A1          3  : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SL1-A1          4  : 0 0 0 0 0 0 0 0 0 0 0 0 3167928 3169872 43999 44026
```

Accumulative sweeps log file. The file /tmp/ibmon/ibmon_sweeps.log accumulates information from all sweeps. Upon the initiation of a new sweep, **ibmon** extracts from ibmon_last_sweep.log the changes which occurred to PM and BM registers with respect to the previous run. These changes are appended to ibmon_sweeps.log.

Verbose error events log: /tmp/ibmon/ibmon_events.log. This file provides verbose information for every event reported to the standard output (including baseboard management). This report includes the name of each counter the threshold of which gets crossed compared to the last sweep preformed by **ibmon**. Following is an example.

```
-I- 10:06:41 H-1              1  : link_down_counter            tot:4     inc:4
-E- 10:06:41 Cable: H-1/P1 <=> SL1-A1/P4
-I- 10:06:41 H-1              1  : symbol_error_counter         tot:47    inc:47
-E- 10:06:41 Cable: H-1/P1 <=> SL1-A1/P4
-I- 10:06:41 H-1              2  : link_down_counter            tot:4     inc:4
-E- 10:06:41 Cable: 0x0002c9000100d052/P2 (I) <=> SL1-A1/P1
-I- 10:06:41 H-1              2  : symbol_error_counter         tot:44    inc:44
-E- 10:06:41 Cable: 0x0002c9000100d052/P2 (I) <=> SL1-A1/P1
```

6.4.2.3 Board Management (BM) Reports

Similar to the performance monitor reports, the *baseboard* events are reported (in brief) to the standard output and (in detail) to the file ibmon_bbm.log located under /tmp/ibmon/ .

Error messages printed to stdout. Each device gets indications from its board monitors whenever a board-malfunction occurs. Three types of board malfunctioning are reported to the standard output. They are listed here with examples.

1. Temperature Threshold Crossing:

```
-E- 10:06:42 Temperature Alarm on Device:SL1-A1
```

2. Power Supply Drop below the allowed 10% margin:

```
-E- 10:06:43 Power supply Alarm on Device:SL1-A1
```

3. Fan Malfunction:

```
-E- 10:06:48 Fan Alarm on Device:SL1-A1
```

Verbose BM messages. These are placed in the file: /tmp/ibmon/ibmon_bbm.log. This file provides verbose information for every BM event reported to the standard output. It also contains all module (system boards) and chassis (system back-plane) information of the devices. An example is listed next.

```
-I- Chassis Info Record for swlab228_i2c/main/U1
-I- Temperature 50 [C]
I- Power Measured/Expected [V]: 1.8/1.8 2.5/2.5
-I- Fans Status: ok ok ok
```

6.5 ibfwmgr

ibfwmgr is a firmware management tool over an IB cluster. It provides the means for installing and updating device firmware (FW). This tool also supports FW customization (e.g., per board type), and preserves the customization during FW updates. The tool is intended to provide cluster-wide FW management. It is capable of handling a cluster of heterogeneous systems and devices with different customizations of HCAs and switches.

To burn firmware on a host channel adapter (HCA), IBADM must be running on the local host connected to the HCA.

6.5.1 ibfwmgr Synopsis

ibfwmgr [-h] [-v] [-V] [-g <device-guid>] [-m <multiple device file>] [-r regular-expression] [-q] [-e] [-b] [-n] [-s <fw-specification-file>] [-f]

The command line **options**:

-g <device-guid> - access device by user provided guid

-m <multiple devices file> - access devices by user provided list of guids in a file (one per line). The guid may be followed by a PSID to be used by the device.

-r <regular-expression> - limit the target devices to those matching the user-provided regular expression.

Note: The -g, -r and -m options provide the scope of **ibfwmgr** operation. If none of the options is specified, the default is access to all the devices in the cluster.

-q - query the devices in the selected scope for firmware information. If this option is not specified, a firmware burn is performed.

-e - burn even if the required firmware version is not equal to the existing one.

-b - force a firmware burn even if the firmware version is up-to-date

-n - allow a non-fail-safe FW burn

-s <fw-specification-file> - use user provided firmware specification file instead of the default one (specified in /etc/ibadm.conf).

-f - non-interactive mode. No input from user is required.

-h - for help.

-v - for **ibfwmgr** version number.

-V - for verbose mode (DEBUG | WARNING | INFORM).

6.5.2 ibfwmgr Configuration And Initialization

6.5.2.1 Introduction

Different Mellanox firmware releases exist for different Mellanox (silicon) device types. When a FW release is burnt to a device, it may be customized to match a desired configuration. A configuration file (supplied by Mellanox or other system providers) is used during the burn process to customize specific FW parameters to match the desired configuration. This customization makes possible the use of the same FW release for different types of boards hosting the same silicon device.

In order to track the parameter-set with which the firmware is prepared, a parameter-set ID (PSID) field (a 16 ascii character) is integrated into the firmware image. The PSID uniquely identifies the configuration file used to generate the image. **ibfwmgr** reads the PSID from the device to preserve the current configuration during updates to the device FW.

6.5.2.2 FW Related Configuration In /etc/ibadm.conf

The following parameters in /etc/ibadm.conf are used by ibfwmgr:

FW_DIR - The default location of the directory that holds the FW specification file and usually the FW releases for all device types in the cluster.
FW_WORK_DIR - ibfwmgr uses this directory for keeping temporary files during the burning process. This directory *must* be on a shared file system, available for all hosts in the cluster.

6.5.2.3 FW Specification File

The FW Specification File (FWSF) defines the following parameters required for ibfwmgr operation:

- Location of FW files for each device type.
- Location of configuration files for each PSID.
- Default PSIDs for devices with un-initialized PSID.
- Device specific FW files.
- Device specific PSIDs.

A file path in the FWSF is considered an absolute path if it starts with a '/'. Otherwise, the path is relative to the FW_DIR (parameter) defined in the /etc/ibadm.conf. It is recommended to use relative paths to avoid changing FWSF if the FW_DIR is changed.

The default path to the FWSF is: FW_DIR/fw_spec.fws. The location can be overridden using the -s flag.

The FW specification file is described in details in section 6.5.6 "FW Specification File Format" on page 42.

6.5.3 ibfwmgr Operation Modes

ibfwmgr supports the following operation flows: query and burn.

6.5.3.1 Query Mode

When ibfwmgr runs in query mode (-q flag), it scans the cluster (or the given guid / group of guids) and provides the following information on each IB device: device type, device name, device guid, current FW version, PSID, FW status, and the available FW version (defined in the FW specification file).

If the PSID is read from the FW specification file rather than from the device, it is marked with a '*', '+', or '!' in the query output of **ibfwmgr**. The following example (Figure 5) shows the query output of **ibfwmgr** using the FW specification file listed in section Section 6.5.6. Notice the '*' for the PSID of the device H-1/U1. It is the result of the ForceGuidPsid setting in the FW specification file.

Figure 5: Example of an **ibfwmgr** query run on a cluster

```
GUID               TYPE    NAME      VER   PSID     STATUS       REQ-VER FW-FILE
0x0002c90109769a00 MT23108 H-1/U1    3.1.0 MT_001*  NEEDS_UPDATE 3.2.0   /fw/fw-23108.mlx
0x0002c90109769b00 MT23108 H-2/U1    3.1.0 NONE     NEEDS_UPDATE 3.2.0   /fw/fw-23108.mlx
0x0002c90109769c00 MT23108 H-3/U1    3.1.0 NONE     NEEDS_UPDATE 3.2.0   /fw/fw-23108.mlx
0x002c90109fb2b501 MT43132 SL1/L1/U1 5.2.0 431324XC UP_TO_DATE   5.2.0   /fw/fw-43132
0x002c90109fb2b502 MT43132 SL1/L1/U2 5.2.0 431324XC UP_TO_DATE   5.2.0   /fw/fw-43132
0x002c90109fb2b503 MT43132 SL1/L1/U3 5.2.0 431324XC UP_TO_DATE   5.2.0   /fw/fw-43132

PSID Sources Legend:
* = forced by guid
+ = forced by device type
! = using the default psid provided for the system, board and device name.
```

6.5.3.2 Burn Mode

Running **ibfwmgr** with any set of flags, not including '-q', searches in the selected scope-cluster for all the IB devices requiring an update, and updates them with the latest available FW version.

The **ibfwmgr** burn flow is as follows:

- **ibfwmgr** scans the cluster for IB devices. It reads the FW version of each device and compares it with the available FW release provided in the FW specification file.
- **ibfwmgr** prints to stdout the IB devices requiring a FW update, and prompts the user to authorize the burn process (for all the devices).
- **ibfwmgr** prepares the FW images for the different devices according to their configuration files. This stage keeps temporary files in FW_WORK_DIR (set in ibadm.conf).
- **ibfwmgr** burns the devices with the new FW images.
- During burn, **ibfwmgr** displays the status of each device on stdout. Status can be one of the following:
  - UP_TO_DATE - the device does not require a FW update.
  - BURNING - the FW of the device is currently being burnt.
  - BURN_FAILED - the FW burn process failed (see error reports in /tmp/ibfwmgr.log).
  - NEEDS_RESET - the device was successfully burnt and a reboot is required to activate the new FW.

  Note: **ibfwmgr** does NOT reset the device after burn. The user should reset the burnt devices in order for the new FW to be loaded.

  Note: Interrupting **ibfwmgr** may cause FW image corruption for the currently burnt devices.

6.5.4 Cluster FW Initialization

ibfwmgr does not burn 'dead' devices which have no FW installed. For the devices that do have a running FW, initialization consists mainly of assigning the correct PSID for each device, and installing its latest available FW.

Since the PSID is a relatively new feature of the FW, most IB devices of Mellanox boards and systems are not initialized with a valid PSID. The following procedure initializes those IB devices with a valid PSID.

1. Set up the cluster and check it using the **ibls** and **ibmon** tools.
2. Set the FW parameters in /etc/ibadm.conf.
3. Set the FW directory: Untar the FW archive from the IB Gold Distribution CD. This places the new FW files in the FW_DIR.
4. Check the FW specification file, fw_spec.fws, in FW_DIR. Make sure that the FW files for each device are taken from the correct location. Check PSID assignments.
5. Run a query: "ibfwmgr -q". This is to verify that all needed files are accessible and to obtain the devices that require a FW update.
6. Note that all Mellanox switch systems (MTS9600, MTS14400 and MTS2400) have been recognized and assigned a default PSID (marked with '!').
7. For HCAs, you need to provide the correct PSID based on the board type and configuration. If you do not have those, please contact your local FAE.

   There are a few ways to specify the exact PSID to be used for each device:

   a) Through the FWSF using the [ForceDevTypePsid] group. The same PSID is set for all the devices of a specific type. For example, to set all the MT25208 devices in a cluster to have the PSID MT_00A0000001, add to the FWSF:

```
[ForceDevTypePsid]
MT25208 = MT_00A0000001
```

   Note: This method is recommended if your cluster is homogeneous and all HCA boards are identical.

b) Through the ‘guid file’ supplied with the -m flag of **ibfwmgr**. For each device, include a line in that file with a device node guid and its desired PSID. For example:

```
0x0002c90109769a00 MT_0020000001
0x0002c90109769b00 MT_0020000004
...
```

c) Through the FWSF using the [ForceGuidPsid] group. Set the desired PSID for each device using its guid:

```
[ForceGuidPsid]
0x0002c90109769a00 = MT_0020000001
0x0002c90109769b00 = MT_0020000004
...
```

8. After setting the PSIDs rerun “ibfwmgr -q” and check that the PSID assignments are correct.
9. BURN !
   Note: To change the PSID for a device with an up-to-date FW version is up-to-date you must use the ‘-b’ flag of **ibfwmgr**.

10. After the cluster is burnt successfully it is recommended to take the following steps:
    a) Reboot the cluster to load the new FW to the devices.
    b) Query to make sure that all the devices are up-to-date with the correct PSID.
    c) Remove the force PSID statements from the FWSF and/or Guids file, re-query, and check that PSIDs are read from the devices and not from a file. (See the query mode section for details.)

A note on failsafe burning. Fail-safe burning ensures that a device can be booted even if the burn process did not complete successfully (causing firmware image corruption). Keeping two images for a device and updating them serially guaranties that at least one of the images is intact at all times. The Mellanox devices MT47396 (InfiniScale III), MT23108 (InfiniHost), and MT25208 (InfiniHost III Ex) support fail-safe burning. However, some FW versions of these devices cannot be burnt in fail-safe mode, and such an attempt causes an error (reported in the log file). For those FW versions, use **ibfwmgr** with the ‘-n’ flag to allow a non-fail-safe burn.

6.5.5 Cluster FW Update

To update cluster FW using **ibfwmgr** follow these steps:

- Create a directory under FW_DIR with the name of the release.
- Place the firmware file and the .brd files (all supplied by Mellanox) in the newly created directory.
- Under FW_DIR there is a link that points to the latest FW version. Update the link to point to the newly created directory.
- Check the fw_spec.fws in FW_DIR. Make sure the FW files are taken from the correct location.
- Burn.

6.5.6 FW Specification File Format

The FW specification file follows the format of Windows .INI files:

```
;; Comment
[Group1]
param1 = value1
param2 = value2
...
[Group2]
param1 = value1
param2 = value2
```

...

The following table describes the groups and parameters supported in the FWSF:

Table 4 - FW Specification Groups And Parameters

Group Name	Parameter	Value	Comment
DevTypeFwFiles	Mellanox device type	Path of the FW file for this device type	For MT43132 the path is a directory.
PsFiles	PSID	Path for the FW configuration file for this PSID	Supported files are of type *.brd for HCAs and *.ini for switches. For MT43132 the value is a board-type, not a file.
DefPsIdByDevRegex	A regular expression	PSID	A Device without a defined PSID may use the PSID provided in case the given regular expression matches its system name followed by its device name. The regular expression must not include spaces or the '=' character.
ForceGuidFwFiles	Node guid	Path of the FW file for this device	Allows overriding the above DevTypeFwFiles settings for a specific device.
ForceDevTypePsid	Mellanox device type	PSID	Allows forcing PSID for all devices of the given type in the cluster
ForceGuidPsid	Node guid	PSID	Allows forcing PSID for a specific device.

6.5.6.1 Example Of A FW Specification File

See Figure 6 below.

Figure 6: Example of a FW Specification file

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; FW SPECIFICATION FILE
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;
;; Define the FW files to be used by each device type
;;
[DevTypeFwFiles]

MT47396 = fw-47396/IS3FW.BIN
MT23108 = fw-23108/fw-23108-a1-rel.mlx
MT25208 = fw-25208/fw-25208-rel.mlx

;; NOTE: For MT43132 fw-release is a directory – not a file
MT43132 = fw-43132

;;
;; Map Parameter Set ID to the actual parameters set files (either INI or BRD)
;;
[PsFiles]

MT_0000000001 = fw-23108/Cougar_Mellanox_128MB.brd
MT_0020000001 = fw-23108/Cougar_Mellanox_256MB.brd
MT_0040000001 = fw-23108/Cougar_Cub_Mellanox_256MB.brd
MT_0050000001 = fw-23108/Cougar_Cub_Mellanox_512MB.brd

;; For MT43132, Value is board type, not a file.
321324XB       = MTEK32132-4X-RevB
431324XC       = MTEK43132-4X-RevC
431324XCL47    = MTEK43132-4X-RevC-Leaf-4-7

;;
;; Defaults section: Provide a default PS-ID if it is not set on the device.
;; Currently we provide default by system type and device name (board and
;; unit) as a regular expression. NOTE: spaces are not allowed.
;;
[DefPsIdByDevRegex]

; Gazelle Leafs U1-3
(MTS9600|MTEK43132|Gazelle).*/L.*/U[123]  = 431324XC
; Gazelle Leafs U4-7
(MTS9600|MTEK43132|Gazelle).*/L.*/U[4567] = 431324XCL47
; Gazelle Spines
(MTS9600|MTEK43132|Gazelle).*/S.*/U       = 431324XC

;; Force a specific PS ID for a specific guid.
;; I.E. ignore the PSID stored in the devices.
[ForceGuidPsid]

0x0002c90109769a00 = MT_001
```

6.6 ibcon

The **ibcon (ibconsole)** tool enables viewing specific device registers all around the IB fabric without the need to physically connect to the devices, whether these are in the In-Band range or Out-of-Band range. The DISPLAY environment variable should be set correctly for ibcon to work.

6.6.1 ibcon Synopsis

ibcon [-h] [-v] [-o bus type] [-i device_id] [-g guid] [-n name]

The command line **options** are:

-h - for help.

-v - for ibmon version number.

-i - list devices with a matching device type (MT43132 / MT23108 / MT47396 / MT25208...)

-o - bus type to be used: IB / I^2C / PCI / Ethernet. Default is IB.

-g - list devices with a matching guid only.

-n - list devices with a matching name only.

6.7 ibcert

The **ibcert** (IB cluster certification) tool provides a sequence of operations to be performed on a cluster in order to certify the cluster is operational and error-free. The sequence is composed of the following seven steps:

1. Running the subnet manager and verifying that no critical errors occurred (e.g., duplicate GUIDs). This step exits upon error.
2. Testing the cluster to verify that no credit loops exist in the topology. This step exits upon error.
3. Querying cluster devices for their firmware versions. This step reports errors but does not exit if they occur.
4. Verifying that the physical topology matches the one specified in the topology file provided by the user.
5. Verifying that no bad cables exist in the cluster at a basic check level (without running data in the cluster).
6. Verifying that no bad cables exist in the cluster at a higher check level (with running data in the cluster).
7. Verifying that the performance meets the minimum requirements of performance (above a pre-defined minumum bandwidth and lower than a pre-defined maximum latency).

Though all seven steps are required for a ‘healthy cluster’ certificate, the user can run **ibcert** steps in several ways: in single steps, in a consecutive subset of steps, or all steps in a row. These options can be used for debug or other purposes. See “ibcert Synopsis” below.

Note: **ibcert** requires a cluster to include at least one switch device.

6.7.1 ibcert Synopsis

ibcert [-h] [-v] [-V] [-o <num-of-step>] [-f <num-of-step>] [-t <num-of-step>]

The command line **options** are:

-h - for help.

-v - for ibmon version number.

-V - for verbose mode (DEBUG | WARNING | INFORM).

-o - perform a single step out of the **ibcert** steps above. This option overrides the ‘-f’ and ‘-t’ options.

-f - perform **ibcert** steps starting from step number specified. (If ‘-t’ is not used, the last step is no. 7)

-t - perform **ibcert** steps up to (including) the step number specified. (If ‘-f’ is not used, the first step is no. 1)

6.7.2 ibcert Logfile

ibcert generates an cluster_certification.log file where it reports all cluster certification queries and errors.

6.7.3 Usage and Output Examples

Standard Output Example. Below is an example of the usage and its **displayed** output of the **ibcert** tool.

```
[root@swlab228 root]# ibcert -f 1 -t 4
Loading Cluster Certification from: /usr/lib/ibadm/app-clustercert1.0
-I- ---------------------------------------------------------------
-I- |          Mellanox Cluster Certification Flow              |
-I- |    detailed log file:/tmp/cluster_certification.log   |
-I- ---------------------------------------------------------------
-I- | Please make sure the following conditions are met     |
```

```
-I- | before running the certification:                              |
-I- | 1. No IB applications are running.                             |
-I- | 2. IBADM is started on all the cluster hosts.                  |
-I- | 3. A list of the cluster host IPs is available.                |
-I- |                                                                |
-I- | NOTE: OpenSM will be killed and IBADM will                     |
-I- | be restarted during this flow.                                 |
-I- |                                                                |
-I- | PLEASE PRESS ENTER TO START THE FLOW.                          |
-I- ----------------------------------------------------------------

-I- 1st step: OpenSM
-I-    waiting for OpenSM to sweep ..
-I- 2nd step: Credit loop check
-I-
-I- 3rd step: firmware version report
-I-
-I- There are several devices which firmware is not up to date
-I- Please see /tmp/cluster_certification.log for further information
-I-
-I- 4th step: Topology verification report
-I-
-I- ----------------------------------------------------------------
-I- |  Cluster Certification step 4 completed successfully          |
-I- |    for further information please see log file                |
-I- ----------------------------------------------------------------
```

Logfile Output Example. Below is the logfile output by **ibcert**.

```
------------------------------------------------------------------
|   CLUSTER CERTIFICATION LOG FILE                               |
------------------------------------------------------------------

------------------------------------------------------------
|   1st step: OpenSM                                       |
------------------------------------------------------------
 Cluster certification OpenSM step passed successfully
------------------------------------------------------------
|   2nd step: Credit loop check                            |
------------------------------------------------------------
-I- Credit Loop Check Report
```

-I- No credit loops found in:6 CA to CA paths

-I- No credit loops found in:1 Multicast:0xC000 CA to CA paths

-I- No credit loops found in:1 Multicast:0xC001 CA to CA paths

Cluster certification Credit Loop step passed successfully

| 3rd step: firmware version report |

GUID TYPE NAME VER PSID STATUS REQ-VER FW-FILE

0x0002c90109769a00 MT23108 swlab228/U1 3.3.2 ??? UP_TO_DATE 3.3.2 /etc/ibfw/fw-23108/fw-23108-a1-rel.mlx

0x0002c90109768ea0 MT23108 swlab229/U1 3.3.2 MT_0000000001 UP_TO_DATE 3.3.2 /etc/ibfw/fw-23108/fw-23108-a1-rel.mlx

0x0002c90000000000 MT47396 swlab229_i2c/U1 0.4.0 MT_0060000002 UP_TO_DATE 0.4.0 /etc/ibfw/fw-47396/IS3FW.BIN

| 4th : Topology verification report |

| End cluster certification process after 4th step |

| Topology check |

Appendix A. IBADM Custom Installation, Configuration, and Initialization

A.1 Custom Installation (Under Directory <dir>) Steps

1. Make sure the environment variable PATH includes <dir>/bin
2. Make sure the environment variable LD_LIBRARY_PATH includes <dir>/lib
3. Make sure the environment variable TCLLIBPATH includes <dir>/lib
4. cd <dir>
5. tar -zxvf ibadm.<ibadm rev>.<uname –r>.tgz

A.2 IBADM Setup Wizard

The IBADM default configuration supports cluster-wide operation with a secondary fabric. However, several other "standard" modes of operation (like a single node mode) are supported too by means of an interactive Setup Wizard. The Setup Wizard can be invoked using the command 'ibadm config'. Through a series of text screens and prompts, the user is guided through the different options to select from. The user inputs are validated for correctness and the ibadm.conf file is output by the wizard. The "standard" configurations supported by the wizard are:

- Cluster-wide operation with an In-Band agent
- Cluster-wide operation with Local Host agents
- Cluster-wide operation with both Local Host agents and In-Band agents
- Single node operation with a Local Host agent
- Single node operation controlling a switch system connected through a USB to I2C adaptor

A.3 Custom Configuration

Custom configuration is intended for System Administrators who wish to modify the default configuration of the IBADM package. This is possible by means of placing a modified version of the file ibadm.conf in /etc/ibadm.conf. (The original file resides under the target installation sub-directory: "lib/ibconf1.0/".)

The ibadm.conf file provides the means to specify the IP and port numbers of the In-Band server, the IB Names server (which consolidates device data from In-Band and OutS-of-Band sources), and various other controls. For more information regarding the optional configuration parameters, check the documentation in the ibadm.conf file.

A.3.1 An Example

The following set of files illustrates how to setup a system where the In-Band server is run on a *local* host named **ibmaster**, the IB Names server on a *remote* host named **ib_cluster_adm,** and the rest of the applications on a **laptop**.

/etc/ibadm.conf

This file is to be installed on ibmaster, ib_cluster_adm and the laptop:

IBBS_HOST carries the name of a local host on which the In-Band server runs

IBBS_HOST=ibmaster

NS_HOST carries the name of the remote host on which the Names server runs

NS_HOST=ib_cluster_adm

/etc/ibadm.hosts

Required on the machine running the name server ib_cluster_adm

ibsw1 # this is an MTS9600 switch system running the embedded Out-of-Band server

ibsw2 # another one

h1 # a host system running the Out-of-Band server

h2 24443 # a host using a non-standard port number or its Out-of-Band server

A.4 Initialization

Following is the sequence of operations required to bring up IBADM.

> Note: For IBADM to run properly, the following two files must be edited by the user: ibadm.topo and ibadm.conf (see below).

1. Turn on all the IB Systems.
2. Start the IB driver on all the hosts of the fabric.
3. Start OpenSM or any other SM to initialize the fabric. To use OpenSM simply run: opensm.
4. Prepare ibadm.topo, the topology file describing your subnet (see 5.2 “ibadm.topo - IBADM Topology File” on page 23):
   - The default location for this file is: /etc/ibadm.topo. (The location can be modified by changing the IBBS_TOPO_FILE variable in /etc/ibadm.conf.)
   - Note that the host running your In-Band services should be called “H-1” in the topology file (The name can be changed by simply changing the IBBS_NAME variable in /etc/ibadm.conf)
5. Start the In-Band server machine by running: ibadm start
6. In case the IB device Names Server is not running on the same machine as the In-Band server, login to the IB Names Server machine and run: ibadm start

Appendix B. Log Files

The following table lists the names of log files, the tools that produce them, and a brief description of their contents.

Table 1 - Log Files

Log File	Output Of Tool	Contents
ibis.log	driver	IBADM lower level driver - reports events in driver
ibbs.log	server	In-Band server output - reports events on server
ibns.log	server	Name server output - reports events on server
obbs.log	server	Out-of-Band server output - reports events on server
ibmon_events.log	ibmon	Reports all threshold violation events
ibmon_sweeps.log	ibmon	Reports all progressing PM register values from successive sweeps
ibmon_last_sweep.log	ibmon	Log file for the last sweep results
ibmon_bbm.log	ibmon	Log file for baseboard management reports
ibmon.dbg.log	ibmon	Debug log file when running in debug mode
cluster_certification.log	ibcert	Reports all cluster certification queries and errors

Appendix C. IBNL Netlist Format

C.1 Overview

The IBADM topology file provides the means to describe the IB fabric using a set of predefined systems. A system definition is provided in a single file in IBNL format, which describes the internal InfiniBand connectivity of the system in terms of boards and devices.

When IBADM starts, it parses all the available system definition files before handling the topology file. The files are located under the following directory relative to the installation prefix: <prefix>/lib/ibadm/ibdm1.0/ibnl. IBADM supports any new system that is described using the IBNL file format and has a corresponding file in this directory above.

This appendix describes the IBNL file format used to define the internal IB connectivity of an arbitrary IB system. It outlines the main concepts (terms) used by the file, provides details of how to obtain such a file, formally defines the file syntax in BNF-like format (YACC readable), and concludes with a real example.

C.2 Main Concepts

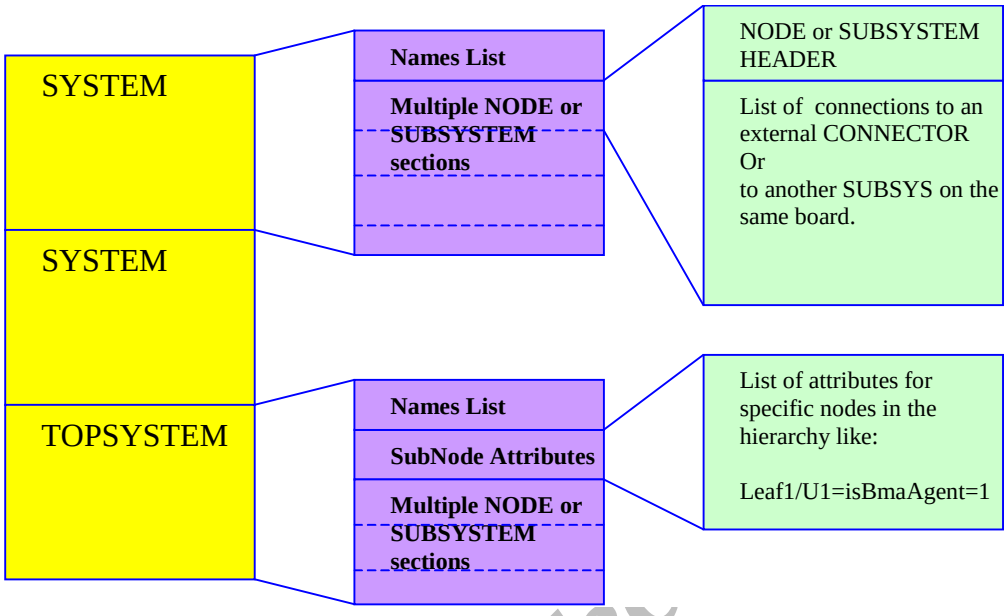
The following terms are used for defining generic system IB connectivity in the IBNL file format:

NODE	The instantiation of an IB device whether it is a switch or a channel adapter
SUB-SYSTEM	The instantiation of a board or module in a system
SYSTEM	A definition of a board or a module
TOPSYSTEM	The top-most system defined in the given file
SUB-SYSTEM MODIFIER	Many systems allow multiple variations of their sub-system components. For example, the 12X Leaf board of an MTS9600 Switch System has many variations. The differentiation is via a modifier which is a suffix to the board name. The modifier should follow the format ": <any name>". The IBNL format supports assigning multiple names to the same board.

C.3 File Format

The IBNL file is line-sensitive and requires different sections to start on a new line. The file is divided into several (optional) SYSTEM sections and one TOPSYSTEM section. Each section has one or more names. Figure 1 on page 54 shows a diagram describing the file sections with more details.

Figure 1: IBNL File Sections



Connections are defined inside the system level only, therefore they can be of two types:

1. Between any node or sub-system to another node or sub-system
2. From any node or sub-system to a connector of the system

Note that the top system may define ports that are possibly redundant and not connected to any lower level board-connector. In such cases the ports are simply omitted from the resulting system. This feature enables defining the front panel ports of a 12X/4X such that if a 12X Leaf is selected, only the 12X front panel ports are used.

Another important note is that the port width and speed provided at the lowest level have precedence over the definitions provided at upper levels in the hierarchy.

C.4 Writing an IBNL System File

The following procedure should be followed in order to provide a new system IBNL:

a. Name the file after the system name: <any sys name (no spaces)>.ibl
b. Define a SYSTEM section for each board included in the system
c. The port names of the boards are simply strings. A simple format of P<N> can be used, where N is a serial number (but it is possible to pick any name desired as long as it is unique).
d. When different flavors of the boards exist (like a 4X and a 12X option for a board), name the optional boards with a "modifier postfix". For example, if a 12X Leaf board and a 4X Leaf board is supported in a switch system, two SYSTEMs can be defined:

```
SYSTEM LEAF,LEAF:4X,LEAF:4X
…

SYSTEM LEAF:12X,LEAF:12X
…
```

Note that the instantiations of the LEAF boards in another SYSTEM or the TOPSYSTEM need not specify the postfix. They are determined only when the topology file get parsed. The "SYSTEM LEAF" with no postfix will be used by default. To continue the example above, here is how the LEAFs are instantiated in the TOPSYSTEM:

```
TOPSYSTEM MyIbSystem

LEAF leaf1
…

LEAF leaf2
…
```

The actual 4X or 12X version of the LEAF board can then be specified in the topology file CFG section to select the right combination of optional boards in the system. An example would be:

MyIbSystem N001 CFG: leaf2=12X

In this case, leaf1 will be 4X as no special modifier is defined for it (and LEAF is by default a 4X leaf). Leaf2 will be 12X as defined in the CFG section,

e. Special considerations should be made to design the TOPSYSTEM section when several optional board types exist. The TOPSYSTEM section may include multiple definitions for front panel plugs like P1-4X and P1-12X (these are simply names that should follow the silk-writing on the front or back panels). The different flavors of the boards are not required to define the same names for their ports. Thus, for example, a 12X port may have some of the top level connections unconnected and thus the front panel ports of the other flavor will be removed from the final system definition.

As an example, consider a simple board LEAF with 3 4X port flavors and one 12X port flavor. It is recommended to connect it at the top level using the following scheme:

```
SYSTEM LEAF:4X

NODE … U1
  1 -4X-> 4XP1
  2 -4X-> 4XP2
  3 -4X-> 4XP3
…..

SYSTEM LEAF:12X

NODE … U1
  1 -12X-> 12XP1
…..

TOPSYSTEM myIbSystem

SUBSYSTEM LEAF leaf1
   4XP1 -> L1/P1
   4XP2 -> L1/P2
   4XP3 -> L1/P3
   12XP1 -> L1/P1-12X
```

f. Place the file in the <prefix>/lib/ibadm/ibdm1.0/ibnl directory

g. To check the file syntax, you have two options:

1. To use the ibnlparse utility:

   Usage: ibnlparse <ib netlist file> <sys type>

   Where the system type is the top system name.

2. To parse it in context of a topology file using the utility ibdmtr (trace a direct route through the fabric):
   i. Define a topology file with your new system
   ii. Set some CFG options if applicable
   iii. Invoke: ibdmtr -t <topo file> -s <any node>/U1 -p 1 –d 0,1

C.5 Formal Definition in YACC Syntax

```
INT ::= ([1-9][0-9]*|0) ;

WIDTH ::= (4X|8x|12X) ;

SPEED ::= (2.5G|5G|10G) ;

NODETYPE ::= (SW|CA|HCA) ;

NAME ::= ([\[\]\\\*/A-Za-z0-9_.:%@~]+) ;

NL:
   LINE
  | NL LINE;

ONL:
  | NL;

ibnl: ONL systems topsystem;

systems:
  | systems system ;

sub_inst_attributes:
  | sub_inst_attributes sub_inst_attribute NL;

sub_inst_attribute:
  NAME '=' NAME '=' NAME
  | NAME '=' NAME '=' INT
  | NAME '=' NAME ;

topsystem:
   TOPSYSTEM  system_names  NL sub_inst_attributes  insts ;

system:
    SYSTEM system_names  NL insts ;

system_names:
    system_name
  | system_names ',' system_name ;

system_name:
    NAME ;

insts:
  | insts node
```

```
  | insts subsystem ;

node:
   node_header NL node_connections ;

node_header:
    NODE NODETYPE INT NAME NAME ;

node_connections:
  | node_connections node_connection NL ;

node_connection:
    node_to_node_link
  | node_to_port_link ;

node_to_node_link:
    INT '-' WIDTH '-' SPEED '-' '>' NAME
  | INT '-' WIDTH '-' '>' NAME INT
  | INT '-' SPEED '-' '>' NAME INT
  | INT '-' '>' NAME INT ;

node_to_port_link:
    INT '-' WIDTH '-' SPEED '-' '>' NAME
  | INT '-' WIDTH '-' '>' NAME
  | INT '-' SPEED '-' '>' NAME
  | INT '-' '>' NAME ;

subsystem:
    subsystem_header NL subsystem_connections ;

subsystem_header:
    SUBSYSTEM NAME NAME ;

subsystem_connections:
  | subsystem_connections subsystem_connection NL ;

subsystem_connection:
    subsystem_to_subsystem_link
  | subsystem_to_port_link ;

subsystem_to_subsystem_link:
    NAME '-' WIDTH '-' SPEED '-' '>' NAME NAME
  | NAME '-' WIDTH '-' '>' NAME NAME
  | NAME '-' SPEED '-' '>' NAME NAME
  | NAME '-' '>' NAME NAME ;

subsystem_to_port_link:
    NAME '-' WIDTH '-' SPEED '-' '>' NAME
```

```
| NAME '-' WIDTH '-' '>' NAME
| NAME '-' SPEED '-' '>' NAME
| NAME '-' '>' NAME ;
```

C.6 Example IBNL for MTS14400

The following is a partial (one Spine and 2 Leafs) IBNL for the MTS14400 Switch Platform.

```
SYSTEM LEAF,LEAF:4X,LEAF:4X

NODE SW 24 MT47396 U1
   1  -> P13
   2  -> P14
   3  -> P15
   4  -> P16
   5  -> P17
   6  -> P18
   7  -> P19
   8  -> P20
   9  -> P21
   10 -> P22
   11 -> P23
   12 -> P24
   13 -> P1
   14 -> P2
   15 -> P3
   16 -> P4
   17 -> P5
   18 -> P6
   19 -> P7
   20 -> P8
   21 -> P9
   22 -> P10
   23 -> P11
   24 -> P12

SYSTEM LEAF:12X,LEAF:12X

NODE SW 24 MT47396 U1
   1  -12X-> P13
   4  -12X-> P14
   7  -12X-> P15
   10 -12X-> P16
   13 -> P1
   14 -> P2
   15 -> P3
   16 -> P4
   17 -> P5
   18 -> P6
   19 -> P7
   20 -> P8
   21 -> P9
   22 -> P10
   23 -> P11
   24 -> P12

SYSTEM SPINE,SPINE:4X,SPINE:4X
```

NODE SW 24 MT47396 U1

21 -> P63
22 -> P30
23 -> P18
24 -> P6
1 -> P27
2 -> P15
3 -> P3
4 -> P31
5 -> P19
6 -> P7
7 -> P34
8 -> P22
9 -> P10
10 -> P43
11 -> P55
12 -> P67
13 -> P46
14 -> P58
15 -> P70
16 -> P42
17 -> P54
18 -> P66
20 -> P51
19 -> P39

NODE SW 24 MT47396 U2

10 -> P44
11 -> P56
12 -> P68
13 -> P47
14 -> P59
15 -> P71
16 -> P41
17 -> P53
18 -> P65
20 -> P50
19 -> P38
21 -> P62
22 -> P29
23 -> P17
24 -> P5
1 -> P26
2 -> P14
3 -> P2
4 -> P32
5 -> P20
6 -> P8
7 -> P35
8 -> P23
9 -> P11

NODE SW 24 MT47396 U3

```
    9 -> P12
    10 -> P45
    11 -> P57
    12 -> P69
    13 -> P48
    14 -> P60
    15 -> P72
    16 -> P40
    17 -> P52
    18 -> P64
    20 -> P49
    19 -> P37
    21 -> P61
    22 -> P28
    23 -> P16
    24 -> P4
    1 -> P25
    2 -> P13
    3 -> P1
    4 -> P33
    5 -> P21
    6 -> P9
    7 -> P36
    8 -> P24

TOPSYSTEM MTS14400,Rhino

spine1/U1=isBmaAgent

SUBSYSTEM LEAF leaf1
   P1 -> spine1 P1
   P2 -> spine1 P2
   P3 -> spine1 P3
   P4 -> spine1 P4
   P5 -> spine1 P5
   P6 -> spine1 P6
   P7 -> spine2 P7
   P8 -> spine2 P8
   P9 -> spine2 P9
   P10 -> spine2 P10
   P11 -> spine2 P11
   P12 -> spine2 P12
   P13 -> L1/P1
   P14 -> L1/P2
   P15 -> L1/P3
   P16 -> L1/P4
   P17 -> L1/P5
   P18 -> L1/P6
   P19 -> L1/P7
   P20 -> L1/P8
   P21 -> L1/P9
   P22 -> L1/P10
   P23 -> L1/P11
   P24 -> L1/P12
```

SUBSYSTEM LEAF leaf2
P1 -> spine1 P13
P2 -> spine1 P14
P3 -> spine1 P15
P4 -> spine1 P16
P5 -> spine1 P17
P6 -> spine1 P18
P7 -> spine2 P19
P8 -> spine2 P20
P9 -> spine2 P21
P10 -> spine2 P22
P11 -> spine2 P23
P12 -> spine2 P24
P13 -> L2/P1
P14 -> L2/P2
P15 -> L2/P3
P16 -> L2/P4
P17 -> L2/P5
P18 -> L2/P6
P19 -> L2/P7
P20 -> L2/P8
P21 -> L2/P9
P22 -> L2/P10
P23 -> L2/P11
P24 -> L2/P12

SUBSYSTEM SPINE spine1
P30 -> leaf3 P6
P18 -> leaf2 P6
P6 -> leaf1 P6
P27 -> leaf3 P3
P15 -> leaf2 P3
P3 -> leaf1 P3
P31 -> leaf10 P6
P19 -> leaf11 P6
P7 -> leaf12 P6
P34 -> leaf10 P3
P22 -> leaf11 P3
P10 -> leaf12 P3
P43 -> leaf9 P6
P55 -> leaf8 P6
P67 -> leaf7 P6
P46 -> leaf9 P3
P58 -> leaf8 P3
P70 -> leaf7 P3
P42 -> leaf4 P6
P54 -> leaf5 P6
P66 -> leaf6 P6
P39 -> leaf4 P3
P51 -> leaf5 P3
P63 -> leaf6 P3
P29 -> leaf3 P5
P17 -> leaf2 P5

P5 -> leaf1 P5
P26 -> leaf3 P2
P14 -> leaf2 P2
P2 -> leaf1 P2
P32 -> leaf10 P5
P20 -> leaf11 P5
P8 -> leaf12 P5
P35 -> leaf10 P2
P23 -> leaf11 P2
P11 -> leaf12 P2
P44 -> leaf9 P5
P56 -> leaf8 P5
P68 -> leaf7 P5
P47 -> leaf9 P2
P59 -> leaf8 P2
P71 -> leaf7 P2
P41 -> leaf4 P5
P53 -> leaf5 P5
P65 -> leaf6 P5
P38 -> leaf4 P2
P50 -> leaf5 P2
P62 -> leaf6 P2
P28 -> leaf3 P4
P16 -> leaf2 P4
P4 -> leaf1 P4
P25 -> leaf3 P1
P13 -> leaf2 P1
P1 -> leaf1 P1
P33 -> leaf10 P4
P21 -> leaf11 P4
P9 -> leaf12 P4
P36 -> leaf10 P1
P24 -> leaf11 P1
P12 -> leaf12 P1
P45 -> leaf9 P4
P57 -> leaf8 P4
P69 -> leaf7 P4
P48 -> leaf9 P1
P60 -> leaf8 P1
P72 -> leaf7 P1
P40 -> leaf4 P4
P52 -> leaf5 P4
P64 -> leaf6 P4
P37 -> leaf4 P1
P49 -> leaf5 P1
P61 -> leaf6 P1

Appendix D. PSID Assignment

In some cases, the user may wish to use a specific FW configuration not supplied by Mellanox. After setting the new FW parameters in a BRD/INI file, the user should assign a unique PSID (Parameter Set ID) to this new configuration. The PSID is kept as part of the FW image on the device NVMEM. The **ibfwmgr** tool uses this field to retain FW settings while updating FW versions.

This appendix explains how to assign a new PSID for a user customized FW, and how to indicate to the **ibfwmgr** tool that a new PSID exists.

> Note: Please change FW parameters with caution. A faulty setting of FW parameters may result in undefined behavior of the burnt device.

D.1 PSID Field Structure

The PSID field is a 16-ascii (byte) character string. If the assigned PSID length is less then 16 characters, the remaining characters are filled with binary 0s by the burning tool.

Table 1 provides the format of a PSID.

Table 1 - PSID format

Vendor symbol	Board Type Symbol	Board Version Symbol	Parameter Set Number	Reserved
3 characters	3 characters	3 characters	4 characters	3 characters (filled with '\0')

Example: A PSID for Mellanox's MHXL-CF128-T HCA board is MT_0030000001, where:

MT_	Mellanox vendor symbol
003	MHXL-CF128-T board symbol
000	Board version symbol
0001	Parameter Set Number

D.2 PSID Assignment and Integration Flow

To assign and integrate the new PSID to produce the new FW

1. Write the new FW configuration file (in .brd or .INI format).
2. Assign it with a PSID in the format described above. Use your own vendor symbol to assure PSID uniqueness. If you do not know your vendor symbol, please contact your local Mellanox FAE.
3. Set the PSID parameter in the new FW configuration file.
4. Place the configuration file in the "custom" directory located under the FW directory, as specified by the FW_DIR parameter in the ibadm.conf file.

5. To get **ibfwmgr** to use the new PSID and its corresponding configuration file, update the “PsFiles” section in the fw_spec.fws file under the FW directory (see “FW Specification File Format,” on page 42) as follows:

```
.
.
.

;;
;; Map Parameter Set IDs to the actual parameters set files (either INI or BRD)
;;
[PsFiles]

;  <Board Name> - <Company Name>
<NEW PSID> = <BRD/INI file location relative to the FW directory (FW_DIR)>
```

For example, if your new PSID is “VND0030000001”, and the configuration file name is “MyConf.brd”, add the following line to the file:

```
; Custom configuration for ...
VND0030000001 = custom/MyConf.brd
```

6. Use ‘ibfwmgr -m <guid-psid file>’ to burn the destination devices with the new FW configuration (and new PSID).

Appendix E. mlxburn

mlxburn is a simple lightweight tool for firmware (FW) image generation (for Mellanox devices) and burning of HCAs on a local host, and cannot be applied to a cluster. It is provided as part of the InfiniBand Administration tools package (IBADM). However, it does not depend on the IBADM services and/or agents.

E.1 mlxburn Installation

To run **mlxburn** you need to install IBADM local node burning utilities. Software dependencies and installation instructions are explained next.

E.1.1 IBADM Local Installation Software Dependencies

The local installation of IBADM (see instructions below) does not require any InfiniBand software. It only depends on the software packages listed in Table 1.

Table 1 - External Software Dependencies

Software Package	Required Version
Perl	5.6 or later
Expat	1.95 or later

E.1.2 Installation Instructions

1. cd /tmp
2. tar -zxvf ibadm-<ibadm rev>.tgz
3. /tmp/ibadm-<ibadm rev>/install.sh **--local**

E.2 mlxburn Synopsys

mlxburn <-dev mst-device | -wrimage fw-image-file> <-fw mellanox-fw-file | -image fw-image-file [-format BINARY|IMG]> [-conf parameter-set-file][-nofs][-nofs_img]

where:

-dev <mst-dev> - burns the image onto the device with the provided MST device name

-fw <mellanox-fw-file> - specifies the Mellanox firmware file to use (file extension is .mlx or .BIN)

-image <fw-image-file> - uses the given firmware image to burn

-conf <parameter-set-file> - firmware configuration file (.brd or .ini). Needed for image generation (not using the -dev flag) or if auto-detection of configuration fails

-wrimage <fw-image-file> - writes the image to the provided file name.

-nofs - When specified, the burn process will not be failsafe. A non-failsafe burn is required (on the rare occasion) when a new firmware version has modifications in the Invariant Sector

-nofs_img - when specified, the generated image will not be failsafe. If burning is also specified, it will not be failsafe either

-format <BINARY|IMG> - specifies which image format to use. Can be used only with the -wrimage flag. Default is BINARY.

-dev_type <mellanox-device-number>- **mlxburn** must know the device type in order to work properly. This options should be used if auto-detection of the device type (taken from the firmware file) fails.

-h - displays a short help text

-V <INFORM|WARNING|DEBUG>- sets the verbosity level. Default is WARNING.

-v - prints version info and exits

E.3 Tool Description

The **mlxburn** firmware update flow is composed of two separate stages: image generation and image burning. In the image generation stage a given Mellanox firmware release (in .mlx or .BIN format) is processed to generate a 'burnable' firmware image. This image is burnt to an HCA device in the second stage. The burning process retains device specific data such as GUIDs, VSD, and BSN. Also, the burn process is failsafe by default.

mlxburn runs both stages by default, but it may perform only one of the stages by means of command options. If the '-dev' option is not specified (see Section E.2, "mlxburn Synopsys"), the actual device burning is skipped. Specifying the '-image' option skips the image generation stage and loads the provided image (generated in a previous run of **mlxburn** using the '-wrimage' option).

E.3.1 Firmware Customization

A Mellanox firmware release can be customized (usually) to fit a specific board type. The customization is done by using a FW parameter-set file in the image generation stage. This file has a .brd or a .ini format. Each parameter-set file has a unique parameter-set ID (PSID), which is kept in the device flash and allows retaining device configuration during future FW updates.

During a device FW update, **mlxburn** reads the PSID from the device and uses the corresponding .brd or .ini file when generating the FW image. **mlxburn** searches for the files in the same directory of the FW release. When **mlxburn** is used to generate an image file, or when no corresponding parameter-set file is found, the user should explicitly specify which parameter-set file to use.

To produce an image file the user needs to provide the option '-wrimage <target file>'. To actually burn an HCA device, the user needs to specify the option '-dev </dev/mst/dev-file>'.

If run in burning mode, **mlxburn** auto-detects the firmware parameter-set with which the device was previously burnt. It locates and uses this parameter-set file to generate the appropriate image for the device (by merging the FW release with the specific parameter-set required).

To inhibit image generation, the '-image <pre-generated-image-file>' should be used. It instructs **mlxburn** to use the given file for burning the device.

E.4 Examples

- To update firmware on an MT23108 InfiniHost device with the Parameter-set-file auto-detected enter:

  mlxburn -fw ./fw-23108-a1-rel.mlx -dev /dev/mst/mt23108_pci_cr0

- To generate an image without burning enter:

  mlxburn -fw ./fw-23108-a1-rel.mlx -conf ./MTPB23108_128MB.brd -wrimage ./fw-23108.bin

E.5 Exit Return Values

The following exit values are returned:

0 - successful completion

>0 - an error occurred